D-MPQUIC: Optimizing Loss Detection in High RTT Variation Networks

Min-Ki Kim
School of Electronic and Electrical Engineering
Kyungpook National University
Daegu, Korea
gms03158@knu.ac.kr

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

***Abstract*— With the development of the Internet, the number of users on mobile devices is increasing. However, mobile networks have a poor performance because of high round-trip time (RTT) variations. Much research has been conducted to overcome this, including active research on transport layer protocols. Google proposed a new transport protocol called QUIC and demonstrated that QUIC outperforms TCP in the real world. Furthermore, research on the multipath extension of QUIC (MPQUIC) is also being actively conducted. However, MPQUIC has poor performance in networks with high RTT variations caused by the weakness of the loss detection algorithm. In this paper, we improved the performance of MPQUIC by modifying MPQUIC's time-based loss detection algorithm. We confirmed that the download completion time decreased by 55.5% compared with the original MPQUIC.**

Keywords*— *QUIC, Multipath QUIC, Transport Protocol, RTT Variations

I. Introduction

The Internet is constantly evolving. In recent years, not only wired network technologies, but also wireless network technologies have advanced and brought convenience to daily life. Accordingly, the usage rate of mobile devices is increasing. According to a CISCO report [1], more than 70% of the world's population will use mobile networks by 2023. However, the high round-trip time (RTT) variation caused by high-speed movement, obstacles, signal interference by other wireless media, etc. degrades the performance of mobile networks. Much research is being conducted to solve these problems, for which the development of transport layer protocols is essential.

QUIC [2] is a transport protocol proposed by Google in 2012 to provide fast Internet services and adopted as a standard in 2021. According to a paper [3] published by Google, QUIC has been applied to many applications such as Chrome and YouTube, and accounted for more than 7% of global traffic as of 2017. In the same paper, Google stated that QUIC reduces search latency by 8% on desktop and 3.6% on mobile compared with TCP. Furthermore, research on the multipath extension of QUIC (MPQUIC) started in 2017 [4]. As of 2023, several drafts of MPQUIC have been proposed [5–8].

However, MPQUIC is still an early-stage protocol and is not mature enough to completely replace TCP and MPTCP. Particularly, MPQUIC performs poorly in mobile networks with high RTT variations. In environments where RTT is easily changed, frequent packet reordering occurs because the packet's transmission time is not constant. For reliable data transmission, the endpoint waits until the reordered packet arrives or considers it a loss and retransmits the packet. Compared with MPTCP, MPQUIC cannot cope effectively with packet reordering [9], resulting in lower performance in RTT variable networks such as Wi-Fi and LTE.

In this paper, we proposed Dynamic-MPQUIC (D-MPQUIC) to improve the performance of MPQUIC in environments with high RTT variations. We also compared and evaluated the performance of the original MPQUIC.

The remainder of this paper is structured as follows. Section II covers the background of the loss detection method of MPQUIC and related research on QUIC in dynamic networks. Section III provides a description of the D-MPQUIC. In Section IV, we set up various environments using an emulator to measure and assess the performance of our proposed method. Finally, in Section V, we summarize and conclude the paper.

II. Background

A. QUIC and MPQUIC

QUIC is a transport layer protocol that improves HTTP performance and enables fast connection setup and secure communication. Accordingly, the QUIC has several features that differentiate it from TCP. Explaining all of those characteristics is beyond the scope of this paper, so we briefly explain only the characteristics necessary to understand this paper.

- Congestion Control: QUIC can use the congestion control algorithm used in TCP. The congestion control algorithm is divided into a slow start phase and a congestion avoidance phase. In the slow start phase, the congestion window starts from a Maximum Segment Size (MSS) of 2; when the sender receives an acknowledgment (ACK) frame, the congestion window is increased by the number of bytes of the acknowledged packet. Furthermore, when a packet loss or RTT increase is detected, QUIC ends the slow start phase and starts the congestion avoidance phase. This enables the endpoint to reach the maximum available bandwidth quickly. In the congestion avoidance phase, QUIC uses NewReno [10] by default, but other congestion control algorithms such as CUBIC [11] and BBR [12] can be used in context.

- Loss Detection: QUIC, like TCP, detects loss based on the acknowledgment of the packet. There are two

available methods. The first method uses the packet reordering threshold. If the sender receives ACKs for subsequent packets while the previous packet is not acknowledged, the unacknowledged packet is declared as lost and retransmitted. In this case, there is a tolerance for packet reordering, which is called the packet reordering threshold. In general, the threshold value is 3. A timer called Probe Time Out (PTO) is set separately to detect tail loss. When an ACK frame does not arrive within the timeout, the sender doubles the timer considering link congestion and transmits two probe packets. If the ACK of the tail packet does not arrive until the ACKs of the two probe packets arrive, the packet is considered lost and retransmitted. The PTO is determined as follows:

$$\text{SRTT} + \max(4 \times RTT_{VAR}, 1\ ms) + 25\ ms \quad (1)$$

The second method uses a time threshold. In this case, the sender waits for an ACK for a certain period after sending a packet. If the ACK does not arrive over time, the packet is considered lost and retransmitted. The sender does not trigger the PTO timer when the time-based loss detection timer is running to avoid confusion with the PTO timer. The time threshold is determined as follows:

$$\max\left(\frac{9}{8} \times \max(SRTT, RTT_{latest}), 1\ ms\right) \quad (2)$$

Today's mobile devices have multiple network interfaces to enable Wi-Fi and cellular networks. The IETF's Multipath TCP Working Group developed Multipath TCP (MPTCP) to enable the simultaneous use of multiple network interfaces of an endpoint [13]. The use of multipath protocols such as MPTCP offers several benefits, including high throughput proportional to the number of subflows in use and robustness of connectivity. Similarly, research on multipath extensions for QUIC has been conducted since 2017 [4], and as a result, several drafts of MPQUIC have been proposed [5–8]. One of the factors that determines the performance of the multipath transmission protocol is the packet scheduler. MPQUIC can use the packet scheduler used in MPTCP [21-23], and the packet schedulers for MPQUIC [14,15, 23-25] are also being studied continuously.

B. Related Work

According to Arash *et al.* [16], in most cases in desktop environments with a constant RTT, QUIC outperformed TCP in terms of page load time. However, when loading a page with a large number of small objects, TCP outperformed QUIC. Even in environments with a jitter of approximately 10 ms, TCP outperformed QUIC in most scenarios considered by the author.

Jawad *et al.* [17] evaluated the performance of QUIC in terms of RTT variations and MAC layer frame aggregation in Wi-Fi. According to the experimental results, QUIC does not use the frame aggregation of the MAC layer, so its performance is poor over Wi-Fi. Accordingly, the researchers proposed Bursty QUIC. Bursty QUIC does not use packet pacing and uses ACK_DECIMATION mode, which delays ACK until it receives up to 10 packets, to exploit MAC layer frame aggregation. Through these changes, an approximately 30% throughput improvement was obtained over Wi-Fi networks.

A subsequent study [18] by the same author proposed BQUIC which is increased the time threshold used for loss detection to improve QUIC's performance in RTT-changing networks. BQUIC improved throughput by 7% to 28%. Furthermore, the authors confirmed that increasing the time threshold does not adversely affect QUIC performance in static networks.

In our previous study [27], we proposed D-QUIC to improve the performance of QUIC in networks with variable RTT. D-QUIC adds RTT variation to QUIC's loss detection threshold, reducing file download time by up to 52%.

Research has also been conducted to improve MPQUIC's performance in dynamic networks using a scheduler. Hongjia *et al.* proposed a machine-learning-based packet scheduler called Peekaboo [15] to adapt to dynamic environments and send packets on the appropriate path. Performance improved by more than 30% compared with the conventional scheduler.

III. DYNAMIC-MPQUIC

In this section, we modify the loss detection scheme of MPQUIC based on the previous studies introduced in Section II. MPQUIC is slower than MPTCP in dynamic networks with high RTT variations for two reasons. First, because of RTT variations exceeding the threshold, the slow start ends early and maximum available bandwidth cannot be utilized. Second, packet reordering occurs. Both packet reordering and time thresholds have some margin, so they are appropriate in environments where packet reordering occurs less, such as a wired network. However, in mobile networks where RTT variations are severe because of handover, obstacles, and signal interference, frequent reordering occurs that exceeds the threshold. Therefore, even if the actual packet loss does not occur, the loss is falsely determined, and spurious retransmission occurs.

A previous study [18] solved this problem by increasing the time threshold, adding fixed margin until a loss is detected. However, this approach is not optimal because it increases the time required to detect a loss in environments where many actual losses occur. Furthermore, RTT variations above the threshold can cause the same problems as described earlier.

In this paper, to solve the problem of MPQUIC, we apply the loss detection scheme proposed in [27] to MPQUIC. We call it D-MPQUIC. Specifically, we set the default loss detection mode of MPQUIC to time-based loss detection and changed the time threshold as follows.

$$max\left(\frac{9}{8} \times \max(SRTT, RTT_{latest}) + 4 \times RTT_{VAR}, 1\ ms\right) \quad (3)$$

The initial RTT_{VAR} is (RTT_{latest} / 2) and then as follows:

$$0.75 \times RTT_{VAR} + 0.25 \times |SRTT - RTT_{latest}| \quad (4)$$

By adding RTT_{VAR} to the time threshold, the D-MPQUIC can reflect RTT variations in the loss detection. Consequently, even if packet reordering that exceeds the time threshold of

TABLE I. NETWORK PARAMETERS

Access Link Bandwidth [Mbps]	100
Access Link Delay [ms]	10
Bottleneck Link Bandwidth [Mbps]	25, 50, 75 100
Bottleneck Link Delay [ms]	10, 30, 50, 100
Bottleneck Link Loss Rate [%]	0.1
Bottleneck Link Jitter	10% of RTT

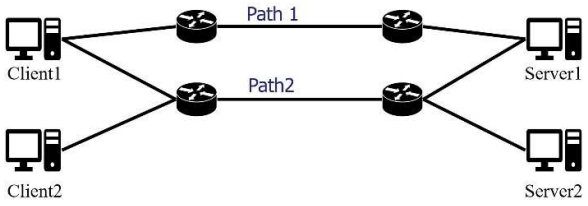

Fig. 1. Network topology used in experiments.

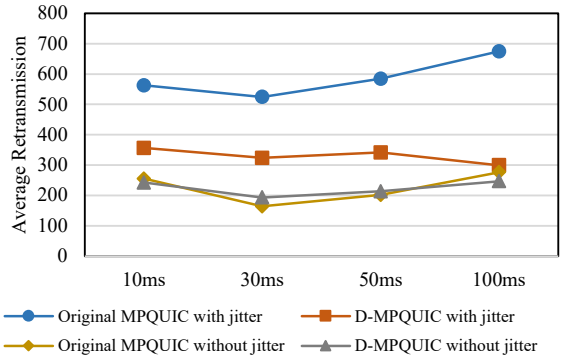

Fig. 2. Average number of Retransmission packets of the original and D-MPQUIC (Path 2 with 50 Mbps bandwidth and 10, 30, 50, 100 ms Delay).

the original MPQUIC occurs, the frequency of determining it as lost is reduced, thus reducing spurious retransmissions. T his modification can appropriately adjust the threshold for "environments where large RTT variations beyond threshold occur" or "environments where RTT variations are small but have many actual losses," which have been a problem when using a small threshold. Therefore, it has the advantage that it can be easily applied to a more diverse environment. Furthermore, in stable scenarios with minimal RTT variations, such as wired networks, it operates at 9/8*max(SRTT, RTT_{latest}) like the original MPQUIC. Therefore, it does not negatively affect scenarios where the original MPQUIC performed effectively.

IV. EVALUATION

A. Experiment Setup

We verified the validity and effectiveness of the modification by applying the change to the MPQUIC implementation [19] based on quic-go. We conducted experiments in Mininet and used token bucket filter and netem to set bandwidth, RTT, loss rate and jitter. The topology used in the experiment consists of server and client connected to two paths, and another server and client sharing one of the paths as a bottleneck link to implement background traffic, as shown in

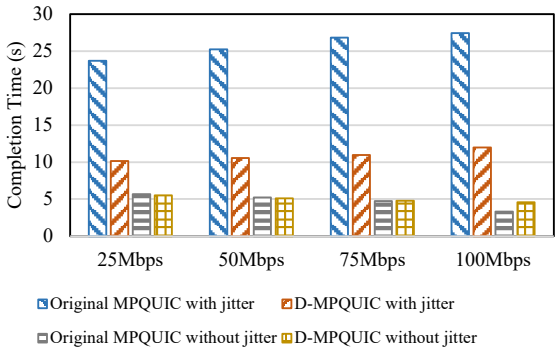

Fig. 3. Completion time of the original and D-MPQUIC (Path 2 with 30 ms Delay and 25, 50, 75, 100 Mbps bandwidth).

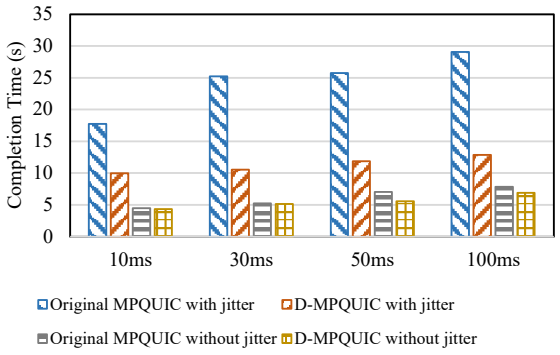

Fig. 4. Completion time of the original and D-MPQUIC (Path 2 with 50 Mbps bandwidth and 10, 30, 50, 100 ms RTT).

Figure 1. The network parameters used in experiments are presented in Table 1. The network parameters of the access link were fixed at 100 Mbps of bandwidth and 10 ms of delay, and the first path was also fixed at 50 Mbps of bandwidth and 30 ms of delay. In order to implement homogeneous and heterogeneous paths, the second path set the bandwidth from 25 Mbps to 100 Mbps and the delay from 10 ms to 100 ms. In scenarios where jitter was considered, the jitter was set to 10% of the bottleneck link's RTT.

The experiment measured the number of retransmitted packets and the completion time required for a multipath host to download a 10 MB file and repeated it 10 times in each scenario. The congestion control algorithm used OLIA [20] in experiments. The packet scheduler used Minimum RTT (MinRTT), the default scheduler of Linux.

B. Experiment Results

Figure 2 presents the mean number of retransmitted packets for the original MPQUIC and D-MPQUIC in different scenarios where a path 2 has a bandwidth of 50 Mbps and delays of 10, 30, 50, and 100 ms. In environments with jitter, it can be observed that the original MPQUIC consistently has a higher average number of retransmitted packets compared to other experiments. As the jitter increases, the frequency of packet delays exceeding the time threshold also increases, resulting in an increase in the number of retransmitted packets. In contrast, D-MPQUIC reduces the number of retransmitted packets by a minimum of 47% to a maximum of 66% compared to the original MPQUIC. As the

jitter increases, there is a trend of decreasing retransmitted packets, which is due to larger jitter allowing more margin for loss detection threshold. When comparing environments with and without jitter, it can be seen that D-MPQUIC has a higher average number of retransmitted packets due to its adaptive speed, but up to a maximum of 8% reduction in the difference of retransmitted packets can be achieved when the path delay is 100 ms. When comparing the number of retransmitted packets between original MPQUIC and D-MPQUIC in environments without jitter, the results of the two protocols are similar, confirming that the two protocols behave similarly in an environment without jitter.

Figure 3 shows the completion time of the original MPQUIC and D-MPQUIC in different scenarios where a path 2 has a delay of 30 ms and bandwidths of 25, 50, 75 and 100 Mbps. In scenarios with jitter, because the time threshold of the original MPQUIC does not reflect RTT variations, all packets that require more than 1.125 SRTT to receive an ACK are considered lost and retransmitted. These spurious retransmissions lead to poor performance: the completion times were 23.7 s, 25.2 s, 26.8 s and 27.4 s in each scenario. In networks with jitter, it can also be observed that as the bandwidth increases, download completion time is getting longer. This is because more packets are delivered at a time through unstable networks, resulting in more frequent Head-of-Line Blocking. In contrast, D-MPQUIC can reflect jitter in the time threshold for loss detection, so the spurious retransmission is reduced compared with the original MPQUIC, resulting in a faster completion time. The completion times of the D-MPQUIC were 10.1 s, 10.5 s, 10.9 s and 11.9 s in each scenario. These results suggest that the modification also performs effectively for the multipath protocol. Without jitter, the performance of original MPQUIC and D-MPQUIC appears similar. This confirms that the modification does not adversely affect the scenario where the original MPQUIC performed effectively.

Figure 4 shows the completion time of the original MPQUIC and D-MPQUIC for scenarios using a path 2 with 50 Mbps bandwidth and various Delays. As with the previous experiments, it can be observed that D-MPQUIC performs better than the original MPQUIC in environments with jitter. In this experiment, D-MPQUIC was able to reduce the average download completion time by a minimum of 37% and up to a maximum of 56% compared to the original MPQUIC. This shows that the D-MPQUIC works well on various jitter and is more effective in environments with large jitter. In environments without jitter, the performance of D-MPQUIC is comparable to or better than the original MPQUIC. This indicates that D-MPQUIC can prevent packet drops due to latency differences between the two paths in environments without jitter. In these experiments, we used MinRTT as a packet scheduler, but if a scheduler such as Peekaboo [15] that considers a dynamic network is used together with D-MPQUIC, we can expect a significant performance improvement.

However, even with the modified loss detection scheme, the performance is still not optimal in the presence of jitter, as seen in the comparison between cases with and without jitter. This is due to the possibility of spurious losses being detected if there are sudden RTT variations that exceed the adaptation speed, despite the time threshold reflecting RTT variations.

V. Conclusion

In this paper, we proposed the D-MPQUIC, which modifies the time threshold used for time-based loss detection of MPQUIC to improve the performance of MPQUIC in networks that experience significant variations of RTT, such as mobile communication networks. D-MPQUIC effectively responds to RTT variations, reducing the average download completion time by 55.5% compared to original MPQUIC. However, the performance was not as good as in the case without jitter, even after applying RTT variation to the loss detection threshold. In future works, the authors plan to address issues caused by sudden RTT variations and evaluate the performance in various environments using D-MPQUIC with other schedulers.

Acknowledgment

This research was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by Ministry of Education (No. NRF-2018R1A6A1A03025109) and by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2023R1A2C1003928).

References

[1] Cisco Annual Internet Report. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, March 2020.

[2] J. Iyengar, and M.Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. Available: https://www.rfc-editor.org/rfc/rfc9000.html

[3] A. Langley *et al*, "The QUIC transport protocol: Design and Internet-scale deployment," SIGCOMM: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pp. 183–196, 2017.

[4] Q. D. Coninck, and O. Bonaventure, "Multipath QUIC: Design and Evaluation," CoNEXT '17, pp. 160–166, 2017.

[5] Y. Liu, Y. Ma, C. Huitema, Q. An, and Z. Li, "Multipath Extension for QUIC," IETF Internet Draft, draft-liu-multipath-quic-04, September 2021. Available: https://datatracker.ietf.org/doc/draft-liu-multipath-quic/04/

[6] Q. D. Coninck, and O. Bonaventure, "Multipath Extension for QUIC (MP-QUIC)," IETF Internet Draft, draft-deconinck-quic-multipath-07, May 2021. Available: https://datatracker.ietf.org/doc/draft-deconinck-quic-multipath/07/

[7] Y. Liu, Y. Ma, O. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Multipath Extension for QUIC," IETF Internet Draft, draft-lmbdhk-quic-multipath-00, January 2022. Available: https://datatracker.ietf.org/doc/draft-lmbdhk-quic-multipath/00/

[8] Y. Liu, Y. Ma, O. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Multipath Extension for QUIC," IETF Internet Draft, draft-ietf-quic-multipath-02, July 2022. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/02/

[9] J. Iyengar, and I. Swett, "QUIC Loss Detection and Congestion Control," RFC 9002, May 2021. Available: https://www.rfc-editor.org/rfc/rfc9002.html

[10] S. Floyd, and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 6582, April 2012. Available: https://datatracker.ietf.org/doc/html/rfc6582

[11] S. Ha, I.Rhee, and L. Xu, "CUBIC: A New TCP-friendly High-speed TCP Variant," SIGOPS Oper. Syst. Rev., vol. 42, no. 5, pp. 64–74, Jul. 2008.

[12] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-Based Congestion Control," Commun. ACM, vol. 60, no. 2, pp. 58–66, 2017.

[13] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and S. Barre, "TCP extensions for multipath operation with multiple addresses," RFC 6824, January 2013. Available: https://datatracker.ietf.org/doc/html/rfc6824

[14] X. Shi, L. Wang, F. Zhang, B. Zhou, and Z. Liu, “PStream: Priority-Based Stream Scheduling for Heterogeneous Paths in Multipath-QUIC,” 29th International Conference on Computer Communications and Networks (ICCCN), pp. 1–8, 2020.

[15] H. Wu, Ö. Alay, A. Brunstrom, S. Ferlin, and G. Caso, “Peekaboo: Learning-Based Multipath Scheduling for Dynamic Heterogeneous Environments,” in IEEE Journal on Selected Areas in Communications, vol. 38, no. 10, pp. 2295–2310, Oct. 2020.

[16] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, “Taking a Long Look at QUIC,” Internet Measurement Conference (IMC), pp. 290–303, November 2017.

[17] J. Manzoor, L. Cerdà-Alabern, R. Sadre, and I. Drago, “Improving Performance of QUIC in WiFi,” IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6, 2019.

[18] J. Manzoor, L. Cerdà-Alabern, R. Sadre, and I. Drago, “On the Performance of QUIC over Wireless Mesh Networks,” Journal of Network and Systems Management 28, pp. 1872–1901, 2020.

[19] qdeconinck/mp-quic, https://github.com/qdeconinck/mp-quic, September 2017.

[20] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, “Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP,” IETF Draft draft-khalili-mptcp-congestion-control-04 (work in progress), Feb. 2013. Available: https://datatracker.ietf.org/doc/html/draft-khalili-mptcp-congestion-control

[21] C. Paasch, S. Ferlin, Ö. Alay, and O. Bonaventure, “Experimental Evaluation of Multipath TCP Schedulers,” in Proc. ACM SIGCOMM Workshop Capacity Sharing Workshop, pp. 27-32. 2014.

[22] 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, pp. 147–159, 2017.

[23] S. Ferlin, Ö. Alay, O. Mehani and R. Boreli, “BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks,” 2016 IFIP Networking Conference (IFIP Networking) and Workshops, Vienna, Austria, pp. 431-439, 2016.

[24] H. Wu, G. Caso, S. Ferlin, Ö. Alay and A. Brunstrom, “Multipath Scheduling for 5G Networks: Evaluation and Outlook,” in IEEE Communications Magazine, vol. 59, no. 4, pp. 44-50, April 2021.

[25] H. Wu, Ö. Alay, A. Brunstrom, G. Caso and S. Ferlin, “FALCON: Fast and Accurate Multipath Scheduling using Offline and Online Learning,” arXiv preprint arXiv:2201.08969 (2022).

[26] M. M. Roselló, “Multi-path Scheduling with Deep Reinforcement Learning,” 2019 European Conference on Networks and Communications (EuCNC), Valencia, Spain, pp. 400-405, 2019.

[27] M.-K. Kim and Y.-Z. Cho, “Improved Loss Detection Scheme for QUIC in Networks with High RTT Variations,” Proceedings of Symposium of the Korean Institute of communications and Information Sciences, pp. 272-273, 2022.