MEDWIN PUBLISHERS
Committed to Create Value for Researchers

Open Access Journal of Data Science & Artificial Intelligence
ISSN: 2996-671X

Server Migration with Multipath-QUIC

Nakano Y, Akashi S* and Matsuzawa T

Department of Information Sciences, Tokyo University of Science, Japan

***Corresponding author:** Shigeo Akashi, Department of Information Sciences, Tokyo University of Science, Japan, Tel: 8013579982; Email: akashi@is.noda.tus.ac.jp / akashi.sub@gmail.com

Research Article
Volume 3 Issue 1
Received Date: March 04, 2025
Published Date: April 03, 2025
DOI: 10.23880/oajda-16000161

Abstract

With the development of IoT and 5G technologies, edge computing is expected to be utilized and HTTP is one of the core network technologies. A new application protocol called HTTP/3, which uses QUIC (Quick UDP Internet Connections) as its transport protocol, was introduced in 2022. QUIC has a connection migration function that allows communication to continue even if the client's IP or port number changes, but it does not consider changes in the server's IP or port number. In this study, we focus on Multipath-QUIC, which is a multipath extension of QUIC, and propose a method of server migration that maintains connection information while significantly reducing packet loss and service impact time. In our experiments, we compared the proposed method with QUIC, which requires another download during migration, assuming an environment in which a server migrates in the middle of a download from a server hosting file. Experimental results showed that the proposed method outperforms QUIC by continuing communication on one path until the server is migrated, and resuming communication on the other path after detecting the completion of the migration of the destination server. The results of this research can contribute to the development of communication in diversified edge computing environments, because communication is performed even in environments where mobility requirements are imposed on servers.

Keywords: IoT; 5G Technologies; QUIC; Server Migration

Abbreviations

HTTP: Hypertext Transfer Protocol; QUIC: Quick UDP Internet Connections; TCP: Transmission Control Protocol; SCTP: Stream Control Transmission Protocol; DCCP: Datagram Congestion Control Protocol; TCP: Transmission Control Protocol.

Introduction

The new application protocol HTTP/3 [1] standardized in 2022 adopts QUIC (Quick UDP Internet Connections) [2-4] as a transport protocol, which was standardized by IETF in 2021 and 2022, respectively. QUIC is equipped with Connection Migration, which is a connection migration function that allows communication to continue even if the client's IP address or port number changes during communication. In addition, recent communication terminals, such as smartphones, can be used while switching between Wi-Fi and cellular lines. Against this background, extended specifications for multipath support have been proposed for transport protocols, such as TCP (Transmission Control Protocol) [5], SCTP (Stream Control Transmission Protocol) [6], and DCCP (Datagram Congestion Control Protocol) [7]. Multipath transport protocols can benefit from effective use of bandwidth, fault tolerance, and QoS by communicating over multiple communication paths. Multipath- QUIC [8], an extension of QUIC, is also under discussion for standardization in the QUIC Working Group of the IETF.

QUIC supports client connection migration and does not consider server migration, i.e., changes in IP addresses and port numbers on the server side. Therefore, it is difficult to continue communication in an environment where mobility requirements occur not only on the client side as well as on the server side.

In this study, we focus on the flexibility of Multipath-QUIC to communicate using multiple paths. We evaluated the usefulness of Multipath-QUIC by modifying it to ensure that servers can continue to communicate even when IP addresses and port numbers change owing to migration.

Related work

QUIC

QUIC is a connection-oriented general purpose transport protocol. QUIC is built on top of the connectionless-oriented transport protocol UDP (User Datagram Protocol). Furthermore, it performs retransmission control, congestion control, and encryption in its own layer. QUIC has the streaming features adopted in HTTP/2, and solves the application layer level HoL blocking in HTTP/1.1 and the transport layer HoL blocking in HTTP/2. In addition to IP addresses and port numbers, QUIC manages connections using a unique ID called Connection ID. This allows QUIC to flexibly respond to changes in IP addresses and port numbers. Therefore, compared to TCP, a general- purpose connection-oriented transport protocol, QUIC is more resistant to changes in the communication environment and packet loss.

Server Migration in Multipath-TCP

Multipath-QUIC is an extension of QUIC that aims to achieve effective bandwidth utilization, fault tolerance, and QoS improvement. Currently, there are several proposed specifications of Multipath-QUIC, including by Yanmei L, et al. [8], Coninck D, et al. [9], Huitema C [10] and Liu Y, et al. [11]. While the jointly proposed specification summarizes the core features of Multipath-QUIC, it does not summarize the features, such as scheduling, setup and teardown of additional paths, address discovery. Therefore, the specification of Multipath-QUIC in this paper is based on the one proposed by Concinck D, et al [9]. Franck L, et al. [12] proposed a mechanism for server migration for Multipath-TCP-enabled VMs without disconnecting the connection by employing transparent connection through tunneling and the mechanism of adding sub-flows (paths) in Multipath-TCP. Figure 1 shows the migration flow.

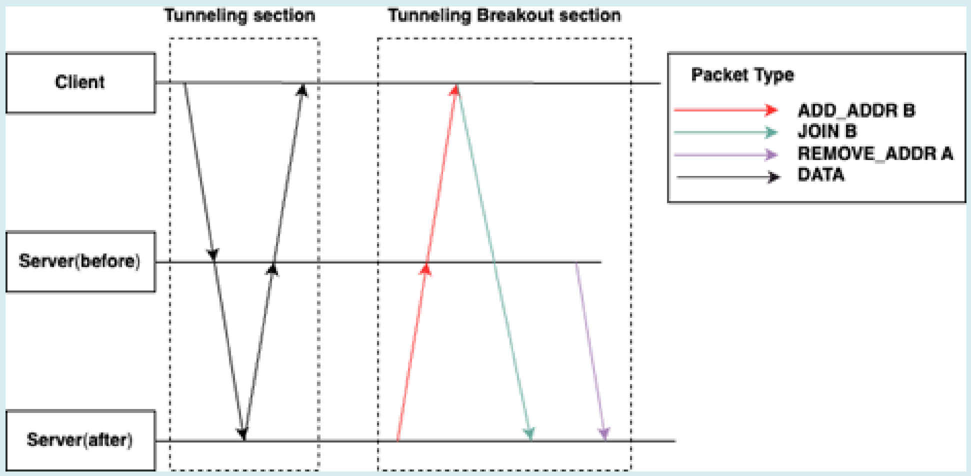

Figure 1: Migration with Multipath-TCP and Tunneling.

In the tunneling section, the post-transition server and the pre-transition server exist simultaneously. However, the pre-tunneling server passes requests from clients to the post-tunneling server and responses from the post- tunneling server to clients. The tunneling breaker sends an ADD ADDR packet to the client, which contains the address information of the new server. The client sends a JOIN packet without going through the server before the transition, and multi-path communication is enabled by employing path A through the server before the transition and path B without the server after the transition. Subsequently, path A is discarded by the REMOVE ADDR packet and path B is promoted to the main flow. Finally, the migration is completed by terminating the server before the migration.

Server Migration in QUIC

Carlo P, et al. [13] proposed an extension to QUIC to support migration of server-side connection when containers migrate between servers. They proposed three strategies for

QUIC to switch communication destinations during server migration: the Proactive-Explicit strategy, Reactive-Explicit strategy, and Pool of Address strategy. In this section, we discuss the Proactive-Explicit and Reactive-Explicit strategies.

Proactive-Explicit Strategy

The Proactive-Explicit strategy is a method in which the client actively switches the destination when it detects a migration requirement (Figure 2). When a migration requirement occurs, the server creates a new QUIC frame that can contain the destination address as a SERVER MIGRATION frame (hereinafter referred to as a MIGRATION frame) and sends it to the client. If the client responds to this frame with an acknowledgement, it immediately switches the destination address.

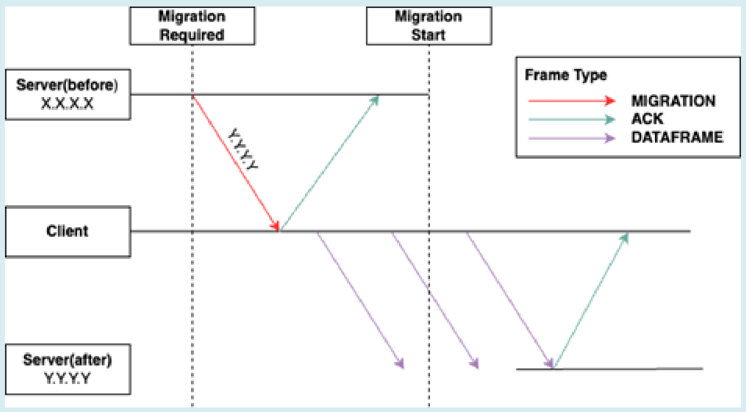

Figure 2: Proactive-Explicit Strategy.

Reactive-Explicit Strategy

The Reactive-Explicit strategy is a method in which communication continues even after the client detects the migration requirement by the MIGRATION frame, and the communication destination is switched when the server before migration stops responding (Figure 3).

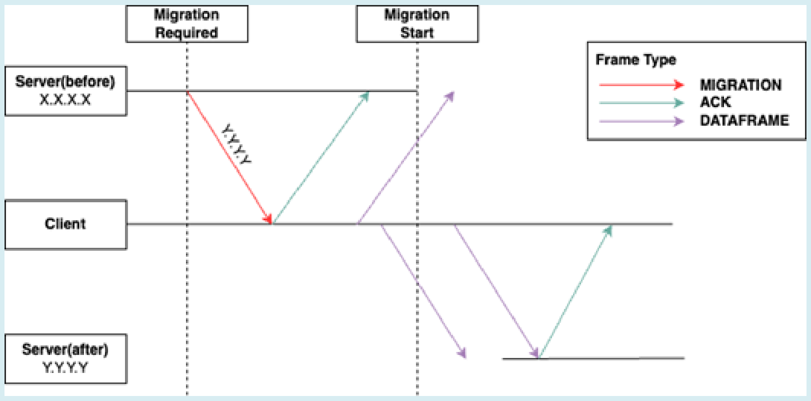

Figure 3: Reactive-Explicit Strategy.

Objectives of our Study

In this study, we combine the Multipath-TCP migration method of Franck L, et al. [12] and the QUIC migration method by Carlo et al. described in related work to realize server migration with Multipath-QUIC. The innovative part of the work by Franck et al. is that when the pre-migration server and the post-migration server exist simultaneously, the connection migration to the post- migration server is accomplished through the addition of a Multipath-TCP path, and the communication with the pre-migration server is terminated by the termination of the path. The innovative part of the work by Carlo et al. is that clients can continue communication and detect the completion of migration even when they do not know when the server migration starts and ends. We apply these techniques to functions related to connection transition and communication continuity at the time of transition requirement detection.

Proposal Method

Figure 4 presents a schematic diagram of the proposed method.

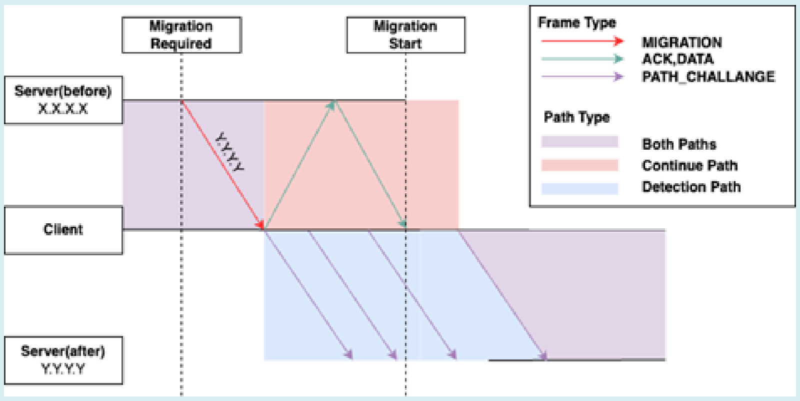

Figure 4: Schematic diagram of the proposed method.

Notification of migration requirements is made by means of a MIGRATION frame containing information on the IP address to be migrated. Upon receiving this frame, the client detects that the server will be migrated soon and terminates one of the active paths that received the MIGRATION frame. The newly created paths are used as the active detection paths, and the other paths are used as the continuation paths. The active detection path repeats the process of adding paths in Multipath-QUIC until a response is received from the migrated server. This allows simultaneous detection of the availability of the destination server and add sub-flows in Multipath-TCP, as in the Proactive- Explicit strategy. Other paths continue to communicate with the original server as in the Reactive-Explicit strategy. These paths are terminated when the server ceases to respond, or when the detection path successfully completes the process of adding a path to the destination server. Thereafter, the newly generated paths are diverted as additional active detection paths, and the process of adding paths to the destination server is performed. The reason for later detection of the availability of this path in addition to the availability detection path of the destination server is to cover the congestion control characteristics of QUIC. In case of packet loss, QUIC doubles the delay for retransmission after packet loss. Therefore, in a situation where the migration is not completed immediately, the interval between packet transmissions of the active detection path gradually increases, and the server may not be detected immediately after the server migration is completed. However, Multipath-QUIC is designed to perform congestion control on a path-by-path basis; thus, the transmission delay of the newly created path that can be created by terminating the path that continues communication is short. Therefore, using this path for operation may cause earlier detection.

Experimental Environment

The experimental network is built by Mininet on a single virtual machine. In the network, latency and bandwidth are controlled by the TC subsystem. The performance of the virtual machine is listed in Table 1.

OS Version	Ubuntu 18.04.6 LTS
Kernel Version	4.15.0-213-generic
CPU Model	QEMU Virtual CPU version 2.5+ memory
Swap	1GB

Table 1: Mininet Virtual Environment OS Version Ubuntu 18.04.6 LTS.

The network topology created by Mininet is shown in Figure 5. Note that the default network interface used by the client is Client-eth0. Therefore, Client-eth1 is never used in QUIC.

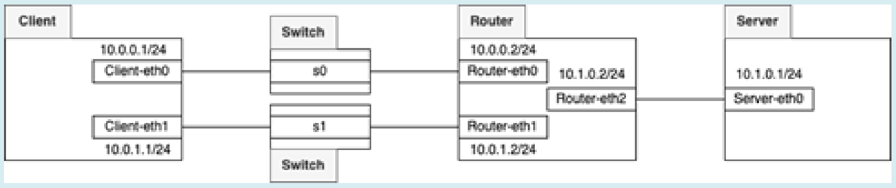

Figure 5: Network topology by Mininet.

In addition, at the interface between the switch and the router, we limit the bandwidth by qdisc and the rate by netem. Table 2 lists the experimental limits and their values. This allows each network interface to limit bandwidth as rate, qdisc buffer size as burst, packet processing delay as latency, packet loss rate as loss, and network delay as delay.

Interface	Rate(mbit)	Burst(kb)	Latency(ms)	Loss (%)	Delay(ms)
s0	100	400	1	0	25
s1	50	200	1	0	25
Router-eth0	100	400	1	0	25
Router-eth1	50	200	1	0	25

Table 2: qdisc and netem restrictions.

The transport protocols and their corresponding congestion control (CC), server, and client implementations are listed in Table 3.

Protocol	CC	HTTP Server	HTTP Client
QUIC	CUBIC	quic-go(v0.5.0)/h2-quic	quic-go(v0.5.0)/h2-quic
Multipath-QUIC	OLIA	quic-go(v0.5.0)/h2-quic	quic-go(v0.5.0)/h2-quic

Table 3: Client and server environments Protocol CC HTTP Server HTTP Client.

The HTTP server hosts a 100MB pseudo-random number file at https://{Host Address}/random. The HTTP client sends a GET request to such a server. Meanwhile, it does not verify the certificate and saves the obtained file in the current directory. Server migration is performed by CRIU (Checkpoint/Restore in Userspace). The migration method to be adopted is Pre-Copy, and the interface of the server to be restored is Server- eth1 newly created by Mininet, and its IP address is 10.1.0.3/24.

Preliminary Experiment

The purpose of the preliminary experiment is to confirm the validity of the network environment used in the experiment and the performance difference of each protocol. We measured the round-trip time, throughput, and turnaround time in an environment where server migration does not occur. The number of measurements was 20.

Figure 6 shows the round-trip time results for each interface. The average round-trip time for Client-eth0 was 53.33 ms, maximum was 54.94 ms, and minimum was 51.50 ms. The average round-trip time for Client-eth1 was 54.16 ms, maximum was 57.01 ms, and minimum was 52.13 ms.

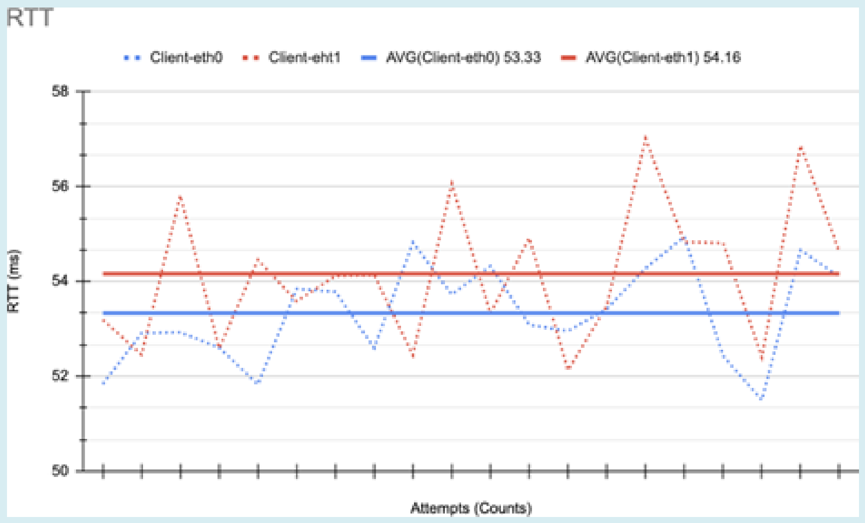

Figure 6: Round trip time results for each interface.

Figure 7 shows the turnaround time for QUIC. The average turnaround time for QUIC was 64.82 seconds, maximum was 68.88 seconds, and minimum was 60.25 seconds.

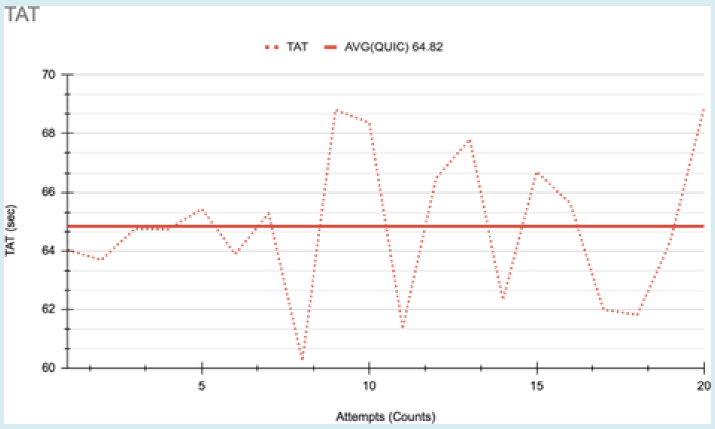

Figure 7: Turnaround time for QUIC with no server migration.

Figure 8 shows the turnaround time for Multipath- QUIC. The turnaround time of Multipath-QUIC was 48.78 seconds on average, with a maximum value of 52.90 seconds and a minimum value of 44.01 seconds.

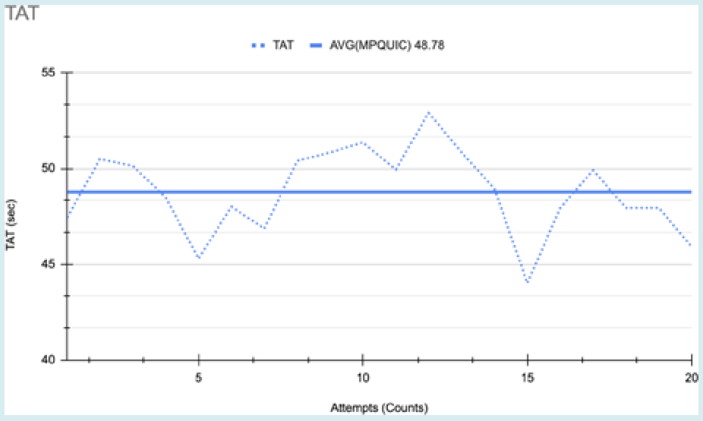

Figure 8: Turnaround time for Multipath-QUIC with no server migration

Finally, QUIC throughput is shown in Figure 9 and Multipath-QUIC throughput in Figure 10.

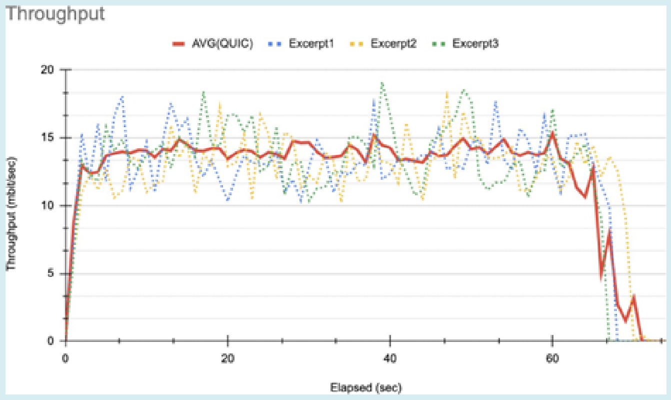

Figure 9: QUIC throughput with no server migration.

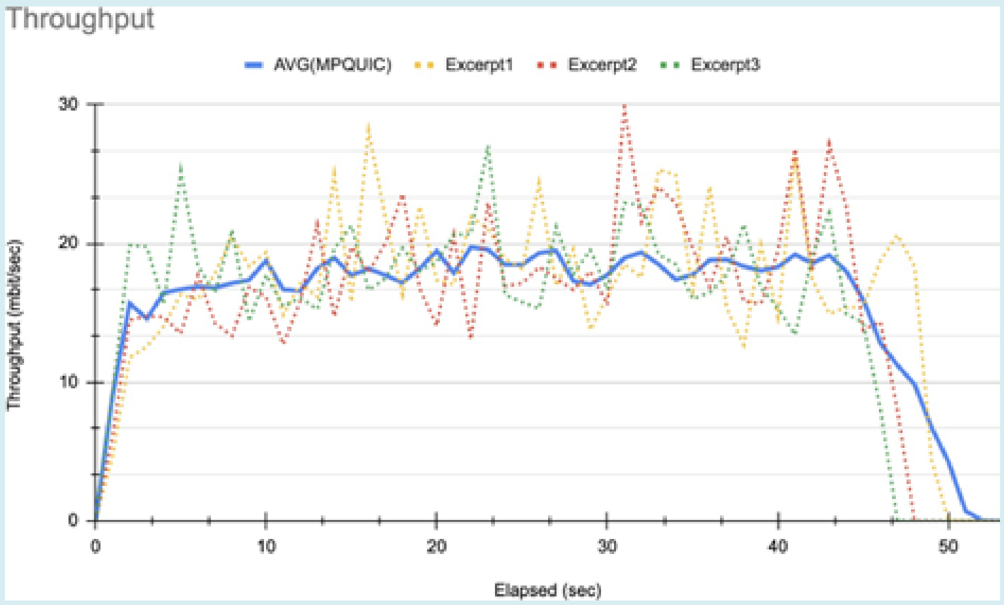

Figure 10: Multipath-QUIC throughput with no server migration.

The round-trip time is at the level of approximately 50ms for both Client-eth0 and Client-eth1. This is because the delay of 25ms set for the router and the delay of 25 ms set for the switch are properly reflected. The fact that the network topology shown in Figure 5 was established by the communication between the two interfaces.

The average turnaround times for QUIC and Multipath-QUIC are 64.8 and 48.78 seconds, respectively, with Multipath-QUIC having a lower turnaround time. This is because Multipath-QUIC effectively utilized paths with a bandwidth of 100 mbit using Client-eth0~Router-eth0 and 50 mbit using Client-eth1~Router-eth1. This is evident in the throughput experiments. The average throughput of QUIC is approximately 14 mbit/sec when the connection is stable, whereas that of Multipath-QUIC is approximately 18 mbit/sec. From the above results, it is confirmed that Multipath-QUIC can achieve higher performance than QUIC by employing multiple interfaces together.

Experiment and Discussion

Overview

In experiments, we performed migration on a server and checked the performance difference between QUIC and the proposed method. In this experiment, the turnaround time and throughput per unit time used in the preliminary experiments are used as evaluation items. The turnaround time cannot be measured because QUIC does not support server address changes. Therefore, when a client program fails, the address of the destination server is provided and the program is executed again. Simulations are performed according to the scenarios in Table 4. Scenario No. 1 is generated only for Multipath-QUIC with the proposed method.

No.	Timing	Action
1	5 sec after the start of handshake	MIGRATION frame, including the destination address 10.1.0.3/24, is sent to the client.
2	10 sec after the start of handshake	Suspend (CRIU) and dump the server process.
3	10 sec after the end of No.2	Restore (CRIU) server process at 10.1.0.3/24. (IP address is changed)

Table 4: Experimental scenario.

Results and Discussion

Figure 11 shows the turnaround time results for QUIC. The average turnaround time for QUIC was 109.28 seconds, maximum was 114.16 seconds, and minimum was 105.77 seconds.

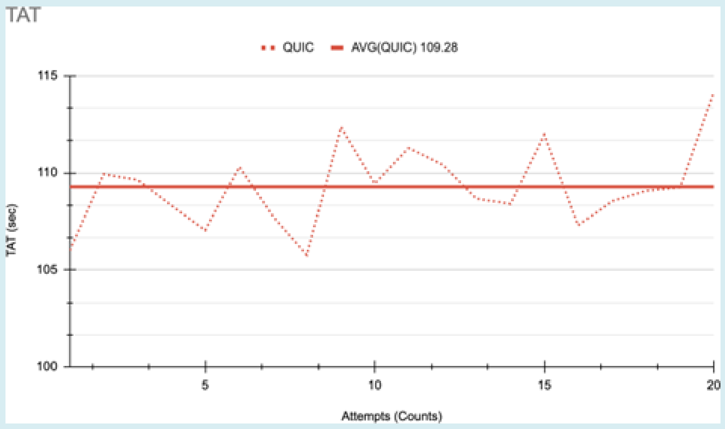

Figure 11: Turnaround time for QUIC with server migration.

Figure 12 shows turnaround time results for Multipath-QUIC. The turnaround time of Multipath- QUIC was 62.72 seconds on average, with a maximum value of 67.93 seconds and a minimum value of 57.29 seconds.

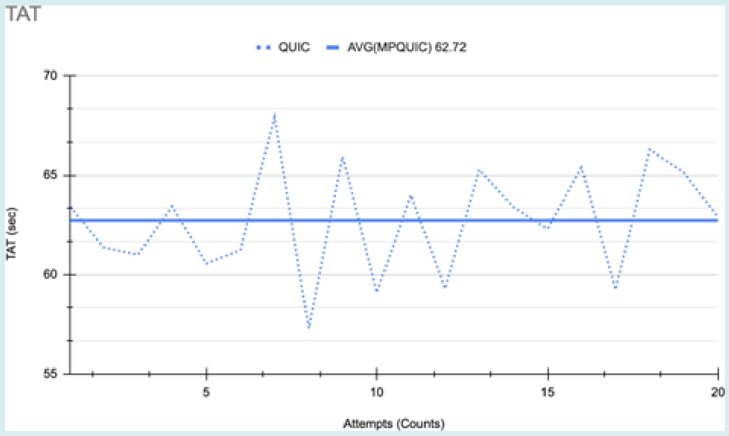

Figure 12: Turnaround time for Multipath-QUIC with server migration

The throughput of QUIC is shown in Figure 13 and that of Multipath-QUIC in Figure 14.

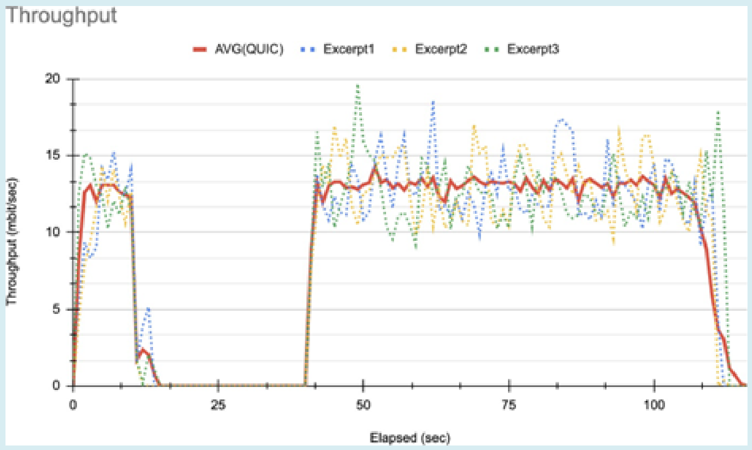

Figure 13: QUIC throughput with server migration.

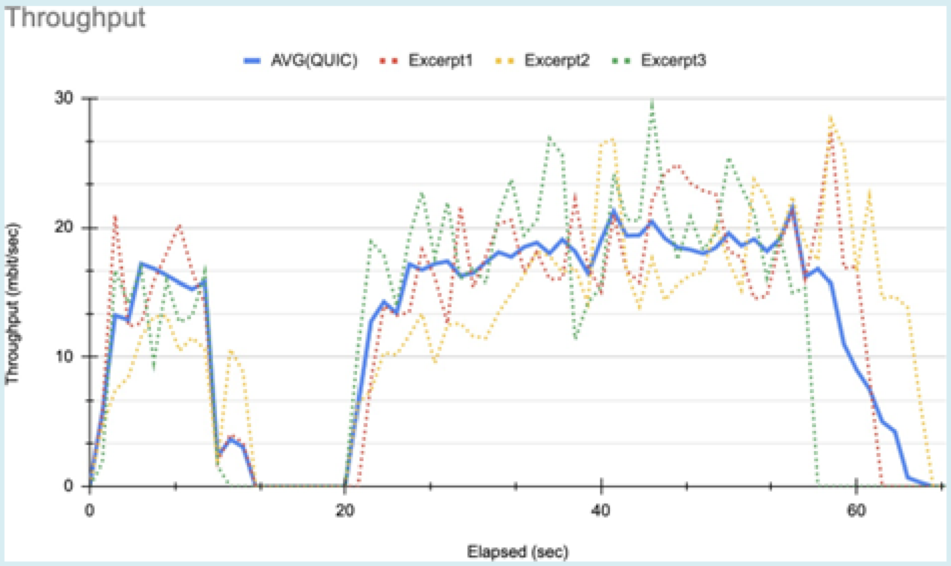

Figure 14: Multipath-QUIC throughput with server migration.

The average turnaround time was 109.28 seconds for QUIC and 62.72 seconds for Multipath-QUIC, showing a significant increase from the values measured in the preliminary experiments owing to the migration effect. However, Multipath-QUIC showed a lower rate of increase than QUIC. The throughput shows the degree to which the network is affected by the migration. The throughput of QUIC drops significantly from the 10-second point, subsequently recovers to the original level after approximately 30 seconds. The throughput of Multipath-QUIC also drops from the 10-second point, but gradually recovers around the 20-second point, when the server actually completes the migration, and recovers to the level measured in the preliminary experiment. The significant increase in QUIC roundtrips is related to connection state destruction and timeouts. When a QUIC client fails to communicate, it starts the connection over again from the handshake. Therefore, when a migration occurs, all data and connection states obtained during the communication up to the migration are discarded, and the time until the migration is lost. The completion time of the server migration is approximately 20 seconds after the start of the handshake, whereas the reconnection of the communication in QUIC occurs at 40 seconds later. This is owing to the idle timeout in QUIC.

The idle timeout is negotiated by handshaking and typically 30 seconds for the quic-go used in this experiment. During this time, the other party's response is checked several times by the probe timeout, and if no response is received, the connection is closed. Therefore, the QUIC client was considered to have failed 30 seconds after the server started the transition, which may have led to the delay in reconnection.

Multipath-QUIC improves on these points. Multipath-QUIC detects servers at approximately 20 seconds, although there is some variation. This is because the migration detection path worked properly. Subsequently, the throughput gradually increases, which is considered to be owing to the resumption of multipath communication to the migrated servers.

However, the average time-around time is lower than in the preliminary experiment, even considering the downtime of approximately 10 seconds required for migration. The reason for this is that the client uses one path as the migration detection path at 5 seconds after the client receives the MIGRATION frame; thus, in effect, the client communicates with the source server as a single path and performance is degraded. In addition, because a path is added to the destination server for each path, its time consuming to resume multi-path communication.

Future Works

First, we consider the timing of server migration. In this experiment, we measured the performance when the server is terminated after the migration requirement occurs and the migration is completed after a certain period. However, with the recent container and VM migration technologies, it is possible to complete the migration without leaving the original server, as in the study by Franck et al. In addition, if the data size of the server to be migrated is sufficiently large, the time between the server termination and completion of migration may exceed the idle timeout period in QUIC. Experiments should be conducted to observe if the proposed method can successfully migrate connections under these circumstances.

Second, we consider the MIGRATION frame. In our experiments, the server recognizes the destination address, stores it in the MIGRATION frame, and sends it to the client, thereby saving the client from the process of address discovery. However, if the server does not recognize the destination address, the client should find the destination address of the server by DNS or other means and reconnect to it, which is not considered in the proposed method. Therefore, it is necessary to find a method to continue communication with the server side even if the communication with the server side is suddenly interrupted.

Finally, we mention the specification of QUIC. The version of quic-go used in this experiment is 0.5.0. However, version 0.21.0 of quic-go meets the RFC-9000 specification, and has been improved in various ways. Therefore, applying the Multipath-QUIC extension to the latest version of quic-go may improve its performance.

Conclusion

In this study, we describe how to migrate a connection when a requirement to change the IP address or port number of a server occurs, while maintaining the connection, and propose a migration method using Multipath-QUIC. In the experiments, we measured the performance difference between the transport protocol QUIC, which requires reestablishment of connections, and Multipath-QUIC, which implements the proposed method, in a scenario where the migrated server is running after a certain period of downtime following the termination of the server before the migration. The results show that the proposed method can communicate more efficiently than QUIC while maintaining connections even during server migration.

References

1. Bishop M (2022) HTTP/3.
2. Jana I, Martin T (2021) QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000.
3. Martin T, Sean T (2021) Using TLS to Secure QUIC. RFC 9001.
4. Jana I, Ian S (2021) QUIC Loss Detection and Congestion Control. RFC 9002.
5. Alan F, Costin R, Mark JH, Olivier B, Christoph P (2020) TCP Extensions for Multipath Operation with Multiple Addresses. RFC 8684.
6. Paul D, Martin B, Thomas D, Nasif E, Jana I, et al. (2024) Load Sharing for the Stream Control Transmission Protocol (SCTP). Internet-Draft draft-tuexen-tsvwg-sctp-multipath-28, Internet Engineering Task Force, Work in Progress.
7. Markus A, Anna B, Andreas K, Veselin R, Stephen J (2025) DCCP Extensions for Multipath Operation with Multiple Addresses. Internet-Draft draft-ietf- tsvwg-multipath-dccp-20, Internet Engineering Task Force, Work in Progress.
8. Yanmei L, Yunfei M, Quentin D, Olivier B, Christian H, et al. (2025) Multipath Extension for QUIC. Internet-Draft draft-ietf-quic-multipath-12, Internet Engineering Task Force, Work in Progress.
9. Quentin D, Olivier B (2021) Multipath Extensions for QUIC (MP-QUIC). Internet-Draft draft-deconinck-quic-multipath-07, Internet Engineering Task Force, Work in Progress.
10. Christian H (2021) QUIC Multipath Negotiation Option. Internet-Draft draft-huitema-quic- mpath-option-01, Internet Engineering Task Force, Work in Progress.
11. Yanmei L, Yunfei M, Christian H, Qing A, Zhenyu L (2021) Multipath Extension for QUIC. Internet-Draft draft-liu-multipath-quic-04, Internet Engineering Task Force, September, Work in Progress.
12. Franck L, Erich N (2019) experiences implementing live vm migration over the wan with multi-path tcp. In IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp: 1090-1098.
13. Carlo P, Luca C, Antonio V, Enzo M (2022) server-side quic connection migration to support microservice deployment at the edge. Pervasive and Mobile Computing 83: 101580.