Intercommunication Method Between Local Edge Computing Devices Using QUIC-Based CYPHONIC

Shota Horisaki
Graduate School of Science and Technology
Meijo University
Nagoya, Japan
shota.horisaki@ucl.meijo-u.ac.jp

Kazushige Matama
Graduate School of Business Administration and Computer Science, Aichi Institute of Technology
Toyota, Japan
matama@pluslab.org

Ren Goto
Graduate School of Business Administration and Computer Science, Aichi Institute of Technology
Toyota, Japan
r0719en@pluslab.org

Katsuhiro Naito
Faculty of Information Science
Aichi Institute of Technology
Toyota, Japan
naito@pluslab.org

Hidekazu Suzuki
Faculty of Information Engineering
Meijo University
Nagoya, Japan
hsuzuki@meijo-u.ac.jp

***Abstract*—Internet of Things (IoT) devices are installed in various locations to realize smart cities, smart factories, smart homes, etc., and are connected to the Internet for data communication. Most current IoT systems are built on a client-server model using servers built on the cloud. On the other hand, edge computing services are increasing and there are still demands to manage sensitive data in a local environment, and a local edge computing is emerging as a new concept to address these demands. Next-generation IoT services are expected to increase the importance of interoperability between different IoT systems, and connectivity to the local edge computing environment will be required. The authors have proposed QUIC-based CYPHONIC (CYber PHysical Overlay Network over Internet Communication) as a communication architecture that simultaneously achieves communication connectivity and mobility transparency in a mixed IPv4/IPv6 environment. This paper shows that QUIC-based CYPHONIC can be used to realize end-to-end encrypted communication between local edge devices located in different local networks over the actual Internet environment with practically acceptable performance.**

***Index Terms*—Overlay Network, QUIC, End-to-End Connectivity, Local Edge Computing**

I. Introduction

Cyber Physical Systems (CPS) have attracted attention as an approach to solving various problems in the real world (physical space) in which we humans live [1]. CPS collects a variety of data in physical space using sensing and computing technologies and stores it in server space. This big data is then analyzed and converted into knowledge using large-scale data processing technologies, etc., and the information and value created through this process is feedbacked to the physical space, which is expected to revitalize industry and solve social problems.

In order to realize CPS, elemental technologies such as Internet of Things (IoT), network technology, database, and artificial intelligence are necessary. In particular, the use of the cloud is indispensable as a platform that constitutes the infrastructure in cyberspace where big data is stored and processed by AI. IoT devices installed in physical space send sensing data and so on to servers built on the cloud via the Internet. In addition, the servers on the cloud feedback notifications and device control commands to the IoT devices that exist in the physical space. In today's Internet environment, however, IPv4 and IPv6 are mixed, and IoT devices connected to a network environment where IPv4 must be used are located in a private LAN. On the communication path between such IoT devices and the cloud, there is always a router with Network Address Translation (NAT) [2] functionality. Although NAT is widely used as a life-extending technology for IPv4, the end-to-end connectivity that is the principle of the Internet is lost due to the NAT traversal problem. Therefore, it is not possible to initiate communication from cyberspace to physical space. However, many IoT systems today address this issue by implementing an arbitrary NAT traversal solution [3]–[7] in their services, or by initiating communication from the IoT device to the cloud, i.e., uplink communication, and using the connections established during the uplink communication to achieve downlink communication. This is a well-known software architecture based on the client-server model.

On the other hand, there are some IoT systems in which complete dependence on the cloud would be inconvenient. When using the cloud, there will be a delay of several seconds in exchanging data between the IoT device and the cloud server. For example, this delay can be fatal to services and systems such as autonomous vehicles, which need to immediately respond to their surroundings in order to drive safely. In addition, since the cloud operates on a pay-as-you-go business model, the number of IoT devices, the size of

the data generated from them, and the amount of computing resources used by the cloud will increase due to the expansion of services, thereby increasing the running cost of services enormously. Therefore, edge computing [8], [9], which utilizes computing resources distributed at the edge of IoT devices and networks, is becoming increasingly important. However, in IoT systems based on edge computing, it must be considered that the server functions do not necessarily reside in the public network, but in a private network, i.e., under a NAT router.

The authors have proposed CYber PHysical Overlay Network over Internet Communication (CYPHONIC) that provides both end-to-end encrypted communication between IoT devices and the capability to continue communication even when the IP address changes due to the movement of IoT devices [10]–[12]. QUIC-based CYPHONIC is a new method that solves the problem of conventional CYPHONIC, which is the inability to communicate with IoT devices under corporate networks. In this paper, we show that the proposed overlay network construction technique with QUIC [13]–[16] can be used to achieve end-to-end interconnection between edge computing devices located in different local networks, and that the communication performance can be achieved without any practical problems.

II. Related Works

A. Peer-to-Peer Secure Update Framework

Herry *et al.* have proposed a Peer-to-Peer (P2P) type secure update framework for the purpose of managing IoT devices and securely distributing software updates [17]. To address situations where IoT devices are located behind NATs or firewalls, or connected to mobile networks with partial or intermittent network connectivity, the framework performs UDP hole punching using Session traversal utilities for NAT (STUN) [3]. This allows a STUN server located on the Internet to obtain address information bound to a NAT router on the communication path to the IoT device, i.e., the NAT's global IP address and opened UDP port number. The STUN server notifies the IoT device of the binding address and port number information. The IoT device uses this information to deliver messages to peers under the NAT using the gossip protocol [18], thus constructing a peer-to-peer overlay network.

This framework is intended to deliver update files to IoT devices, but it can be applied to other purposes as well. However, the scheme uses BitTorrent [19] for file distribution, which makes it difficult to apply to IoT applications that are not suited for this scheme. Also, there is no mention of whether the framework works in mixed IPv4 and IPv6 environments.

B. CYPHONIC

Yoshikawa *et al.* have proposed CYPHONIC that provides end-to-end encrypted communication on an overlay network to applications by assigning a fixed virtual IP address to each node [10], [11]. Fig. 1 shows an overview of CYPHONIC. CYPHONIC consists of a cloud service and CYPHONIC nodes that implement CYPHONIC. The CYPHONIC cloud service consists of the following three types of services.

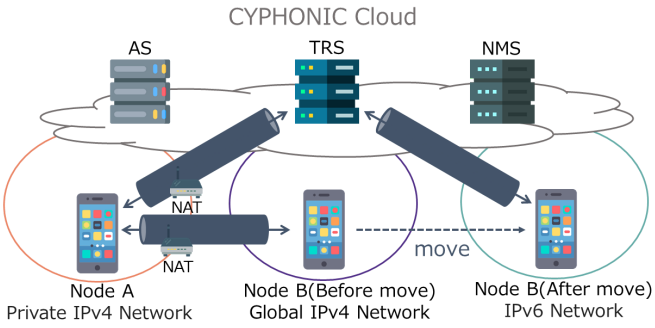

Fig. 1. Overview of CYPHONIC

- Authentication Service (AS)
  The AS authenticates the CYPHONIC node. It generates and distributes a Fully Qualified Domain Name (FQDN), which is the identifier of the CYPHONIC node, a public key certificate, and a shared key used for encrypted communication with the NMS. It also manages the device information of CYPHONIC nodes and works as an intermediate certification authority.
- Node Management Service (NMS)
  NMS manages network information such as IP addresses of CYPHONIC nodes and controls a tunnel construction process based on their information to establish optimal communication paths among CYPHONIC nodes. It also generates and distributes virtual IP addresses and encryption keys to CYPHONIC nodes.
- Tunnel Relay Service (TRS)
  The TRS relays communications between CYPHONIC nodes when they cannot directly communicate with each other. Specifically, when CYPHONIC nodes use different IP address versions, or when CYPHONIC nodes are under different NATs and those NATs are a combination of specific NAT types as shown in RFC 5780 [20], the communication is relayed through TRS.

The AS, NMS, and TRS are deployed in a dual-stack network so that they can be used from either IPv4 or IPv6 networks. By migrating a set of CYPHONIC cloud services to a local edge server, sensitive data can also be managed locally. When a CYPHONIC node moves and its IP address changes, the encrypted UDP tunnel is reestablished, however, the source/destination virtual IP addresses in the virtual IP packets remain unchanged, so the communication flow is maintained and mobility transparency can be achieved. Furthermore, it solves the compatibility problem between IPv4 and IPv6, and can establish end-to-end encrypted communication regardless of whether NAT is used or not.

We have implemented a packet processing mechanism to improve the throughput performance of CYPHONIC [21] and a route optimization function to switch communication between nodes in different private networks from via TRS to end-to-end [22]. We have also changed the tunneling specification from UDP to QUIC [13]–[16], a next-generation HTTP

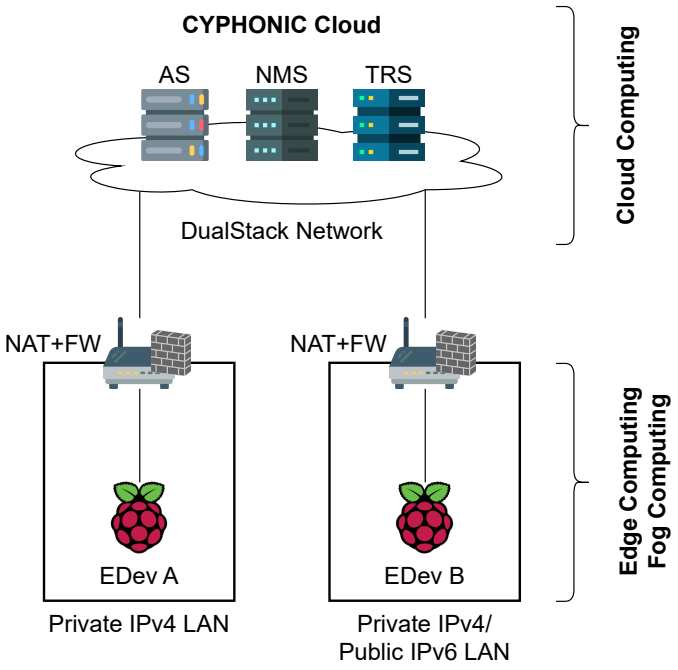

Fig. 2. System configuration

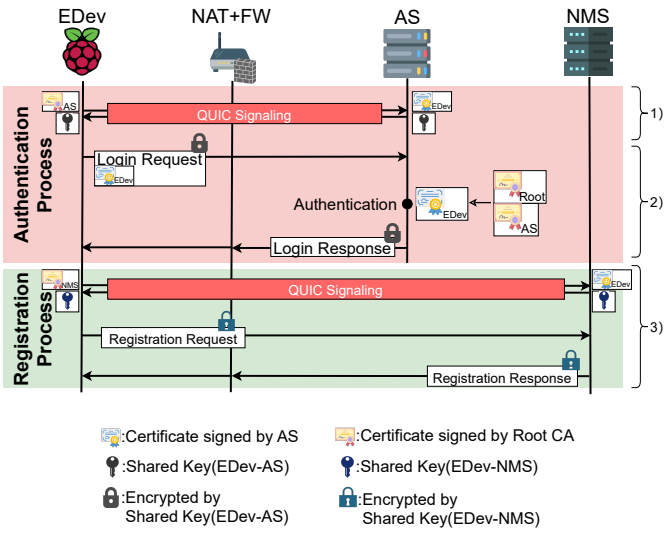

Fig. 3. CYPHONIC authentication and registration Process

protocol, to allow CYPHONIC tunneling communications to pass through firewalls [12], and are reaching a level where it can be used in the current Internet environment.

III. Proposed Method

A. Overview

In this section, we present an interconnection method between local edge devices located in different LANs using a QUIC-based CYPHONIC. Fig. 2 shows the system configuration assumed in this study. Edge devices (EDevs) include IoT devices that are endpoints in edge computing, and fog nodes and servers or gateways installed in LANs in fog computing. QUIC-based CYPHONIC has been introduced in these edge devices. It is assumed that edge devices are connected to a private LAN behind a NAT or firewall and have only a private IPv4 address or a combination of both a private and a public IPv6 address. The CYPHONIC Cloud is assumed to be deployed in a public dual-stack network, such as on a cloud environment.

B. Authentication and Registration processes

Fig. 3 shows a sequence diagram of the authentication and registration process using QUIC-based CYPHONIC.

1) QUIC Signaling: First, QUIC signaling is executed to the AS when EDev is launched. In QUIC signaling, the EDev and the AS perform mutual authentication using the public key certificate distributed by the AS and the public key certificate issued by the Root CA, respectively, and share the key for encrypting the communication between the EDev and the AS.

2) Authentication Process: In the authentication process, EDev sends a Login Request message to the AS to show its authenticity, and performs login authentication using a public key certificate. After successful authentication, the AS sends a Login Response message with the EDev's FQDN and NMS address information.

3) Registration Process: Next, the EDev sends the same QUIC signaling to the NMS, and shares the key to encrypt the communication between the EDev and the NMS. In the registration process (Registration Request/Response), the EDev's network information is registered with the NMS, and the NMS assigns a virtual IP address to the EDev.

C. Route Selection and Tunnel Establishment Processes

After completing the authentication and registration process described above, the EDev dynamically establishes a QUIC tunnel using CYPHONIC when it starts communicating with other devices. While three signaling patterns are shown in Reference [12] depending on the network location of the CYPHONIC node, this paper describes cases where both EDevs are under different NATs or cannot communicate directly, such as between an IPv4-only network and an IPv6-only network.

Hereafter, the initiating and corresponding local edge devices are denoted as EDevA and EDevB, respectively. When EDevA initiates communication with EDevB, it usually performs a name resolution process using the Domain Name System (DNS) in order to find the IP address from the FQDN of EDevB in the application. In CYPHONIC, the route establishment process between EDevs is triggered by the transmission of DNS query messages.

Fig. 4 shows the route selection and QUIC tunnel establishment process using QUIC-based CYPHONIC.

1) Route Selection Process: EDevA sends a Direction Request message to NMS to request the establishment of a tunnel route. This message includes the FQDN of EDevB,

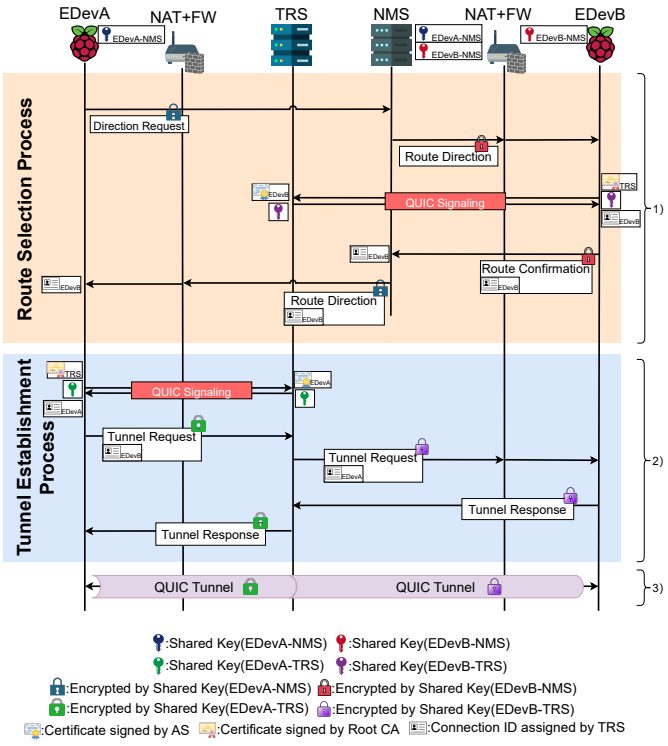

Fig. 4. CYPHONIC route selection and tunnel establishment processes

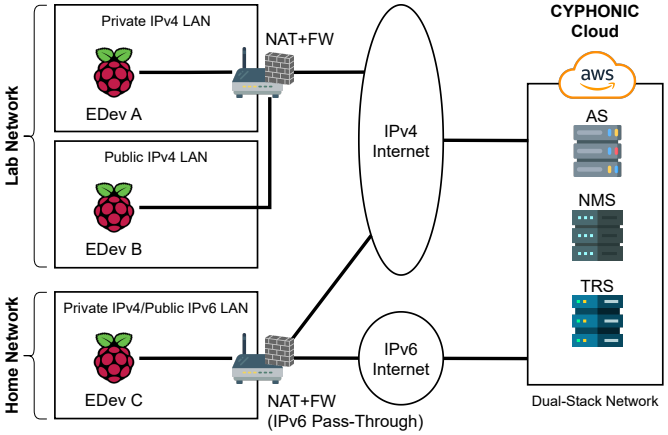

Fig. 5. Measurement environment

which is the communication partner, and NMS receives this message and obtains the network information of EDevA and EDevB, which are the communication pair. NMS calculates the optimal communication path based on the network locations of both EDevs. In this case, since the pattern is via the TRS, NMS sends a Route Direction message to EDevB, instructing it to construct a QUIC tunnel between the EDevB and the TRS. Since the EDev and the NMS have already performed QUIC signaling during the registration phase, the NMS can send a message over the NAT to the EDev.

After receiving the Route Direction, EDevB performs TRS and QUIC signaling, and then sends back a Route Confirmation message to NMS with information on the QUIC connection ID. The NMS includes the information of the connection ID notified by the EDevB in the Route Direction message and instructs the EDevA to construct a QUIC tunnel between the EDevB and the TRS.

2) Tunnel Establishment Process: Next, EDevA starts the tunnel establishment process in accordance with the instructions from the NMS. In this case, to establish a tunnel to the TRS, the EDevA sends a Tunnel Request message after QUIC signaling with the TRS. This message contains the ID of the QUIC connection between EDevB and the TRS, which was obtained from the Route Direction message. The TRS can then identify the QUIC connection on the EDevB side from the received connection ID and bind the QUIC connections between EDevA and the TRS and between the TRS and EDevB. The TRS then forwards a Tunnel Request message to EDevB, informing it of the ID of the QUIC connection on the EDevA side.

Finally, a Tunnel Response message is sent back from EDevB to EDevA via TRS to complete the tunnel establishment process.

3) QUIC Tunneling Communication: After the signaling process of CYPHONIC is completed, EDevA writes the virtual IPv6 address of EDevB obtained by the Route Direction message into the response message of the DNS query and passes it to the application.

Thereafter, when the EDevA application communicates with EDevB, the destination IP address is the virtual IPv6 address of EDevB, so the EDevA encapsulates this IP packet with QUIC to the real IP address of the TRS, which is the end point of the tunnel established by the tunnel establishment process, and sends it via tunnel communication. Since the EDevB connection ID is attached to this tunnel communication, the TRS can relay the received communication from EDevA to EDevB by re-entering the QUIC tunnel on the EDevB side.

In this way, end-to-end encrypted communication between EDevA and EDevB applications can be performed over the CYPHONIC overlay network.

IV. Evaluation

A. Verification Environment

In order to verify the interconnection of local edge computing devices by the proposed method, we constructed the environment shown in Fig. 5. EDev was built by installing QUIC-based CYPHONIC on a Raspberry Pi and deployed on the private and public IPv4 networks in our university laboratory and the author's home network (a dual stack network of private IPv4 and public IPv6), respectively. The CYPHONIC cloud services were built using AWS (Amazon Web Services) EC2, and were operated on a public IPv4/IPv6 stack network. The device and instance specifications of each device are as shown in TABLE I.

As a result of the operation verification in this environment, the following was confirmed.

TABLE I
EQUIPMENT SPECIFICATIONS FOR MEASUREMENT

CYPHONIC Cloud (AS, NMS)	
AWS Service	Amazon EC2
Instance Type	t2.micro
vCPU	1
Memory	1 GB
Network Link Speed	1 Gbps
Amazon Machine Image	Ubuntu Server 22.04 LTS
Region	Tokyo (ap-northeast-1)

CYPHONIC Nodes	
Machine	Raspberry Pi 4 Model B
CPU	Quad Core 1.5 GHz Broadcom BCM2711
Memory	4 GB
OS	Ubuntu 22.04 LTS
NIC	IEEE 802.3ab (1000BASE-T)
WAN (Lab)	SINET (Science Information NETwork)
WAN (Home)	Commufa (1 Gbps FTTH service)

- End-to-end encrypted tunnel communication between EDevB and other EDevs was successful.
- Encrypted tunnel communication between EDevA and EDevC was established via TRS.

B. Communication Delays

Using the validation environment shown in Fig. 5, we evaluate the signaling process and the communication delay in the application when using the proposed method for two patterns i.e., end-to-end encrypted communication and encrypted communication via TRS. The former end-to-end communication path is the case where EDevB and EDevC communicate through the IPv4 Internet. The latter communication path via TRS is the case where EDevB and TRS communicate through IPv4 Internet, and TRS and EDevC communicate through IPv6 Internet. EDevC executed the command textttping6 to EDevB 100 times, and the CYPHONIC signaling processing time and the Round-Trip Time (RTT) required for the ICMPv6 Echo Request/Reply transmitted through the QUIC tunnel were measured.

TABLE II shows the results of RTT measurements. For reference, the RTT (E2E) between EDevB and EDevC and the total RTT (via TRS) between EDevB and TRS and between TRS and EDevC in the general case (General) where CYPHONIC is not applied are also shown. In the case where end-to-end communication is possible, the application communication delay increased by using the proposed method is only 2.2 milliseconds on average, and 4.8 milliseconds on average for the case where the communication is via TRS. Considering the additional processing time for encrypting ICMPv6 packets, this increase in communication delay is acceptable.

Note that the route optimization function [22] allows EDevA and EDevC to switch to end-to-end communication when communicating via TRS through the IPv4 Internet, thus reducing the incremental communication delay.

TABLE II
RTT MEASUREMENT RESULTS FOR REAL NETWORK AND CYPHONIC OVERLAY NETWORK

	General [ms]		Proposed [ms]	
	(E2E)	(via TRS)	(E2E)	(via TRS)
min	14.783	23.505	17.135	27.836
avg	15.656	24.479	17.867	29.285
max	19.234	30.115	20.109	33.001
mdev	0.673	1.070	0.456	0.947

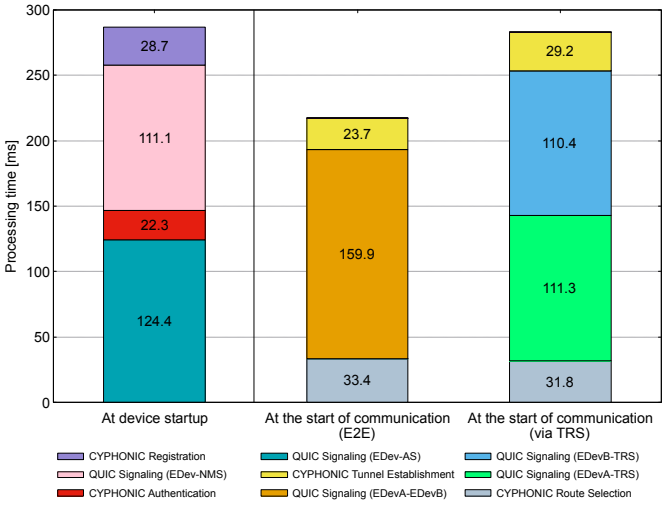

Fig. 6. CYPHONIC signaling time

Fig. 6 shows the signaling processing time of the proposed method. The authentication and registration process shown in Fig. 3 occurs only once at EDev startup, and the processing time was 286.85 milliseconds. Therefore, this processing time does not affect the actual application communication at all.

Next, the processing time of Fig. 4, which occurs at the start of application communication, was 217 milliseconds for end-to-end communication between EDevs and 282.7 milliseconds for communication via the TRS. This processing occurs only the first time an application communicates, however, since the QUIC encrypted tunnel between EDevs has already been constructed if the communicating EDevs are the same, this overhead is not incurred even if different applications are used. In addition, when communicating via the TRS, once QUIC signaling is performed with the TRS, only the connection IDs need to be mapped, thus reducing the time required for QUIC signaling. In this case, the overhead would be reduced to 61 milliseconds. Therefore, the signaling process of the proposed method has a limited impact on the application, and is not considered to be a problem in practical use.

C. Throughput Characteristics

We evaluate the throughput characteristics of the proposed method in the same environment and under the same conditions as shown in Subsection IV-B. A packet of 1,360 bytes was sent from EDevC to EDevB using the command iperf3

TABLE III
THROUGHPUT AND JITTER MEASUREMENT RESULTS BY IPERF3 (UDP)

	Proposed (E2E)		Proposed (via TRS)	
	Throughput [Mbps]	**Jitter [ms]**	**Throughput [Mbps]**	**Jitter [ms]**
min	29.5	0.138	29.8	0.179
avg	29.9	0.181	29.9	0.309
max	30.0	0.244	30.0	0.574
mdev	0.206	0.039	0.087	0.157

in UDP mode with a bandwidth limit of 30 Mbps per minute. TABLE III shows the measured throughput of the encrypted tunnel communication of the proposed method. The average throughput of 29.9 Mbps is maintained for both end-to-end and via TRS. The Jitter was not found to be significant enough to affect the performance of the tunneling communication.

V. CONCLUSION

This paper described a method for interconnecting local edge devices using QUIC-based CYPHONIC, which can build a virtual overlay network between nodes and provide end-to-end encrypted communication to applications. We implemented a hypothetical system using AWS and Raspberry Pi in a real environment and evaluated its performance. As a result, it was confirmed that the application communication was not affected by the increase in signaling processing time due to CYPHONIC, and that there was no significant change in throughput or RTT performance due to QUIC tunnel communication. Additionally, it has been verified that end-to-end encrypted communication over the overlay network via TRS is possible even when both EDevs are installed in private IPv4.

CYPHONIC provides mobility transparency, allowing communication to continue even if a node moves. Previously, edge devices such as cars and drones were assumed to be mobile, but the server side can also be mobile. If the migration of virtual server instances and processes between the cloud and edge environments can be realized using our method, it could be applied to dynamic switching between cloud computing and edge/fog computing.

ACKNOWLEDGMENT

Part of this work was carried out under the Cooperative Research Project Program of the Research Institute of Electrical Communication, Tohoku University.

REFERENCES

[1] E. A. Lee, "Cyber physical systems: Design challenges," in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 363–369.

[2] K. B. Egevang and P. Srisuresh, "Traditional IP Network Address Translator (Traditional NAT)," IETF, RFC 3022, Jan. 2001.

[3] M. Petit-Huguenin, G. Salgueiro, J. Rosenberg, D. Wing, R. Mahy, and P. Matthews, "Session Traversal Utilities for NAT (STUN)," IETF, RFC 8489, Feb. 2020.

[4] T. Reddy, A. Johnston, P. Matthews, and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," IETF, RFC 8656, Feb. 2020.

[5] C. Holmberg and J. Uberti, "Interactive Connectivity Establishment Patiently Awaiting Connectivity (ICE PAC)," IETF, RFC 8863, Jan. 2021.

[6] L. Yang and K. Lei, "Combining ice and sip protocol for nat traversal in new classification standard," in *Proc. of The 5th International Conference on Computer Science and Network Technology (ICCSNT 2016)*, Dec. 2016, pp. 576–580.

[7] F. Huang, L. Yu, T. Shen, and S. Hu, "The P2P Solution Research and Design Based on NAT Traversing Technology," in *Proc. of 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Oct. 2019, pp. 1347–1351.

[8] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[9] J. Portilla, G. Mujica, J.-S. Lee, and T. Riesgo, "The extreme edge at the bottom of the internet of things: A review," *IEEE Sensors Journal*, vol. 19, no. 9, pp. 3179–3190, 2019.

[10] T. Yoshikawa, H. Komura, C. Nishiwaki, R. Goto, K. Matama, and K. Naito, "Evaluation of new CYPHONIC: Overlay network protocol based on Go language," in *Proc. of 2022 IEEE International Conference on Consumer Electronics (ICCE)*, Jan. 2022, pp. 1–6.

[11] K. Matama, R. Goto, C. Nishiwaki, and K. Naito, "Extension mechanism of overlay network protocol to support digital authenticates Protocol," in *Proc. of The 26th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2022)*, Jul. 2022, pp. 81–86.

[12] S. Horisaki, K. Matama, K. Naito, and H. Suzuki, "A proposal of quic-based cyphonic for encrypted end-to-end communications," in *2022 Tenth International Symposium on Computing and Networking (CANDAR)*, 2022, pp. 27–35.

[13] M. Thomson, "Version-Independent Properties of QUIC," IETF, RFC 8999, May 2021.

[14] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," IETF, RFC 9000, May 2021.

[15] M. Thomson and S. Turner, "Using TLS to Secure QUIC," IETF, RFC 9001, May 2021.

[16] J. Iyengar and I. Swett, "QUIC Loss Detection and Congestion Control," IETF, RFC 9002, May 2021.

[17] H. Herry, E. Band, C. Perkins, and J. Singer, "Peer-to-peer secure updates for heterogeneous edge devices," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–5.

[18] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "Epidemic information dissemination in distributed systems," *Computer*, vol. 37, no. 5, pp. 60–67, 2004.

[19] Rainberry Inc, "BitTorrent — The World's Most Popular Torrent Client," https://www.bittorrent.com/.

[20] D. MacDonald and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)," IETF, RFC 5780, May 2010.

[21] R. Goto, K. Matama, R. Aihata, S. Horisaki, H. Suzuki, and K. Naito, "Implementation and evaluation of cyphonic client focusing on sequencing mechanisms and concurrency for packet processing," in *Proc. of The IEEE 12th Global Conference on Consumer Electronics (GCCE 2023)*, Oct. 2023, pp. 1001–1005.

[22] K. Matama, H. Komura, R. Goto, S. Horisaki, H. Suzuki, and K. Naito, "Enhancing route optimization for napt traversal in cyphonic: Design and implementation," in *Proc. of The IEEE 12th Global Conference on Consumer Electronics (GCCE 2023)*, Oct. 2023, pp. 994–998.