Computer Communications 239 (2025) 108157

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A multipath redundancy communication framework for enhancing 5G mobile communication quality☆

Koki Ito [a], Jin Nakazato [a,*], Romain Fontugne [b], Manabu Tsukada [a], Esaki Hiroshi [a]

[a] The University of Tokyo, 7 Chome-3-1 Hongo, Bunkyo, 113-8654, Tokyo, Japan
[b] Internet Initiative Japan Inc., Iidabashi Grand Bloom, 2-10-2 Fujimi, Chiyoda, 102-0071, Tokyo, Japan

ARTICLE INFO

Keywords:
5G
Multipath redundant communication
Mobile network
Vehicular communication
UDP streaming

ABSTRACT

As networks increasingly become the backbone of modern society, the demands placed on them by various applications have become more complex. In particular, the demand for high-capacity, low-latency services such as real-time streaming is increasing every year. Although 5G has been deployed to meet these needs, its effectiveness can vary significantly by location and time, and sometimes falls short of requirements. Traditionally, much of the research to improve communication stability has focused on TCP-based systems, which do not translate well to real-time UDP streaming applications. To address the above challenges, we propose a multipath redundant communication framework designed to improve the quality of real-time media streaming. This framework has been tested using multipath redundant communication over two mobile networks with a moving vehicle in an urban environment. Using a real-time streaming application based on WebRTC, our framework demonstrates a significant reduction in packet loss and an increase in bitrate, outperforming existing multipath redundant communication systems without interfering with the application's congestion control mechanisms.

1. Introduction

Networks have continuously evolved to become essential components of modern society. They support transformative changes such as Industry 4.0 and Society 4.0, and their importance is expected to grow with the advent of Industry 5.0 and Society 5.0 [1]. According to a global traffic survey [2], video content accounts for approximately two-thirds of all network traffic, reflecting the diverse technical requirements of different video applications. For example, on-demand streaming platforms like Netflix, Amazon Prime, and Disney+ require significant bandwidth to deliver high-quality Full High Definition (FHD) and 4 K content [3]. In addition, live streaming services like YouTube Live and Instagram Live, as well as online meeting platforms like Zoom and Microsoft Teams [4–6], which experienced exponential growth during the COVID-19 pandemic, prioritize low latency to ensure real-time interaction. Emerging applications such as remote healthcare and remote operations also require highly reliable, low-latency communications due to their critical impact on safety and security [7]. These diverse use cases underscore the need for network infrastructures tailored to the unique requirements of each application.

The types and nature of media traffic and the environments in which these applications are used have diversified. Traditionally, network connections were made over fixed networks in offices or homes. However, the development of mobile networks has made it possible to use networks on the move or outdoors, especially benefiting the Internet of Things (IoT), where various devices, not just PCs and smartphones, are connected to the network. Among these, connected autonomous vehicles (CAVs) have attracted significant attention. Connected to the Internet and other devices, CAVs provide various functions, such as enabling sophisticated traffic systems through real-time data collection and analysis, providing safety features such as emergency calls, and offering entertainment content. With the advancement of autonomous driving technology, the perception of vehicles as a second living space is emerging among vehicle manufacturers, increasing the demand for convenient in-vehicle communications [8]. A 2022 report from Zoom indicates that 43% of people who participate in meetings from locations other than their desks do so from within vehicles [9].

However, the quality of current cellular networks is only sometimes adequate. This is especially true in urban areas, where communication quality could depend on time, location, and environmental obstacles such as crowded trains or buildings. A previous study [10] involved real-time media communication using Web Real-Time Communication (WebRTC) over cellular networks while driving through urban areas.

☆ This paper is an extended version of the IFIP NETWORKING 2024 conference paper (Ito et al., 2024).

* Corresponding author.
E-mail address: jin.nakazato@ieee.org (J. Nakazato).

https://doi.org/10.1016/j.comcom.2025.108157
Received 7 January 2025; Received in revised form 24 March 2025; Accepted 27 March 2025
Available online 23 April 2025

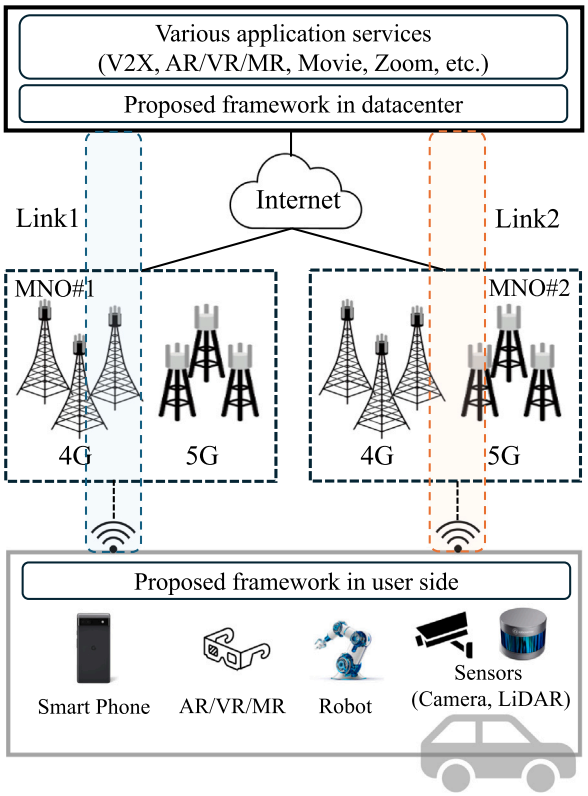


Fig. 1. Overview of this study.

This study reported that even within the same area, signal conditions could vary between mobile operators, leading to differences in communication quality. Using multiple mobile networks simultaneously could improve communication quality. In addition, Lee et al. [11], who studied the use of two cellular networks and WiFi to communicate from moving vehicles, showed that the best communication path can change within seconds, making it difficult to predict which path will have the best quality at the next moment. As a solution to the constant changes in network quality, redundant communication has been proposed, where a device connects to multiple networks simultaneously and sends duplicate packets through each link. The receiving side then processes the first packet that arrives and discards any later packets. While this increases bandwidth consumption by the number of networks connected, it allows communication over the best path at any given time. Several studies [11–14] have extended multipath TCP (MPTCP) [15] to perform redundant communication. However, these methods are tailored for TCP-based communication and are not directly applicable to media traffic, which often uses UDP. In addition, in media streaming using UDP, such as WebRTC, session management and transport control are handled at the application layer, making it challenging to implement multipath without altering existing applications, requiring methods using proxies or tunnels. Our research uses a multipath UDP proxy as middleware to facilitate redundant communication for media traffic and demonstrate its ability to reduce packet loss and delay. However, redundant communication can cause packet order inconsistencies, potentially misleading the application's congestion control and thus limiting bandwidth [16]. In this previous study, Kaneko et al. used techniques based on Real-time Transport Protocol (RTP) header information to transmit WebRTC media data that do not apply to non-RTP media traffic.

In light of these challenges, this research introduces a multipath redundant communication framework that operates at the IP layer to improve the quality of real-time communication without interfering with the application's congestion control mechanisms, regardless of the transport layer protocol, as shown in Fig. 1. The framework achieves redundancy by replicating packets at the sender side and sending the same packet over multiple Mobile Network Operator (MNO) links. This approach can seamlessly integrate into existing media streaming applications using a middleware strategy. In addition, we have implemented a buffering mechanism at the receiving end to address packet order inconsistencies due to redundant communication. This mechanism aims to prevent bandwidth congestion caused by misinterpretation by the application's congestion control system. The effectiveness of this framework is demonstrated through practical experiments involving a vehicle navigating in an urban environment, using the networks of two MNOs, and a real-time streaming application based on WebRTC. In our previous study, we proposed the framework [17], and this study mainly analyzes the results of the proposed content in more detail and discusses the versatility of the experimental environment by implementing it in other environments. In addition, it analyzes not only reliability but also latency and analyzes the system overhead. The main contributions of this paper are summarized below:

- We discuss in detail the framework for multipath redundant communication that we proposed in the uplink scenario, which involves tunneling at the IP layer. This can be applied to existing media streaming applications without modification.
- We resolve packet order inconsistencies caused by redundant communication by introducing a buffering function at the receiver end.
- We evaluate the implemented system using multiple mobile networks in an urban area with a vehicle, demonstrating the effectiveness of the proposed framework in two different environments. Meanwhile, we also evaluate and discuss the limitations of the developed framework.

The remainder of this paper is structured as follows: Section 2 provides an overview of related work and highlights distinctions from previous studies. Section 3 outlines the proposed framework and details its implementation. Section 4 discusses the software and hardware setups utilized in our experiments, describes the experimental setup, and presents the findings. The paper concludes with a summary and explores potential future research directions.

2. Related work

This section reviews previous research on multipath redundant communication to highlight the novelty of this research.

2.1. Study of end-to-end multipath communication

One notable protocol that enables multipath communication is MPTCP. MPTCP is an extension of the TCP protocol that uses multiple paths simultaneously for data transmission. It is compatible with traditional TCP and has the advantage of being applied to existing applications without modification. Multipath communication improves throughput using multiple base stations (multiple frequency bandwidths) and can maintain connections even if one of the base stations (communication paths) fails. However, traditional MPTCP distributes the application's data across multiple communication paths, which can throttle the overall communication if there are significant differences in quality, or more specifically, latency, between the paths used. In addition, packet loss always triggers retransmissions. Several studies have extended MPTCP to reduce packet loss and latency by sending duplicate packets over multiple communication paths. Frommgen et al. [12] proposed ReMP TCP, which incorporates redundant communication into MPTCP, and demonstrated through simulations that it could halve the average round-trip delay. Their study used TCP option headers to assign sequence numbers for packet identity discrimination. Wang et al. [13]

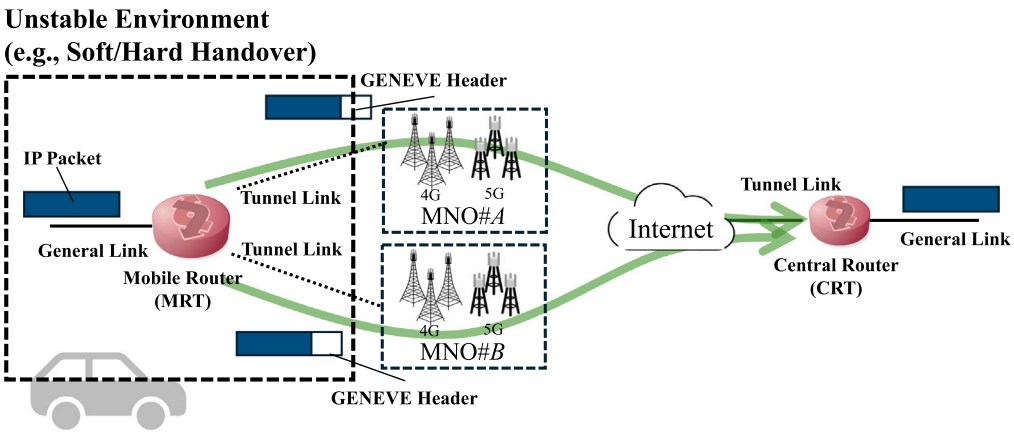


Fig. 2. Overview of the proposed framework.

proposed a set of extensions to MPTCP (called MPTCP-L) for latency-sensitive data communications in cloud data centers, and evaluated it using real data center traffic, demonstrating its effectiveness. Lee et al. [11] proposed a scheduler for MPTCP that performs redundant communication when network latency indicates low reliability, and evaluated its performance from a moving vehicle. Guo et al. [14] also proposed a scheduler that observes path delays and performs redundant communication when quality deteriorates, demonstrating performance improvements in cellular and WiFi networks. These studies demonstrate the effectiveness of redundant communication, but they all rely on TCP, which poses challenges in applying them to UDP, which is commonly used for media traffic. QUIC [18], developed by Google in 2012, is an alternative transport protocol to TCP. While the replacement of TCP by QUIC is underway, as in HTTP/3, MP-QUIC is currently in the process of standardization and practical implementation [19,20]. Standardization discussions include options to duplicate all packets or only those that require retransmission for redundant communication [21]. However, as a reliable protocol with congestion and retransmission control, QUIC cannot directly replace the transport layer of WebRTC. Discussions are ongoing for extensions such as RTP over QUIC, and methods for handling media traffic over QUIC, such as MOQT (Media over QUIC Transport) [22,23]. The implementation of multipath video traffic over QUIC is still pending and warrants further discussion. Boutier et al. [24] adapted Mosh, an SSH-like remote terminal application, for multipath communication, which required significant additions and modifications due to its UDP-based.

2.2. Study of multipath communication

Protocols optimized for real-time media streaming, such as WebRTC, are tightly integrated across all layers, making changes difficult. Middlebox approaches excel in their applicability to existing applications, but they hide the processing performed en route from the end host, making performance tuning difficult, especially when addressing potential bottlenecks such as latency, jitter, and bandwidth. In addition, middlebox approaches make it difficult to control applications using end-host statistics, increasing system complexity for flexible control. To our knowledge, several studies have investigated middlebox techniques for multipathing UDP traffic, as opposed to MPTCP. Liu et al. [25] proposed a multipath UDP (MPUDP) framework through overlay networks, where a central controller manages paths between nodes in the framework and distributes traffic based on usage. Lukaszewski et al. [26] proposed a VPN-based multipath framework and evaluated its performance using TCP and UDP as the underlying protocols. Amend et al. proposed a framework using a multipath datagram congestion control protocol (MP-DCCP) and evaluated its performance through simulation. DCCP [27] is a transport layer protocol designed for applications where timely data delivery is more critical than reliability, such as real-time communications and online gaming. SMPTE 2022–7, a standardization document in the broadcast industry, defines redundant communication for RTP packets, typically implemented using specialized hardware within data center networks [28]. Kawana et al. [29] used QUIC proxy and OpenFlow for redundant communication, reducing packet loss. They also implemented a system that redundantly transmits real-time video from WebRTC over multiple cellular networks, performed measurements from a moving vehicle, and demonstrated that redundant communication improves the communication quality of WebRTC [16]. This method duplicates packets using an MPUDP proxy, with the receiving side's proxy using RTP packet sequence numbers for duplicate resolution. Although it significantly reduced latency and packet loss, it reported a bitrate degradation due to redundant communication. This degradation occurs when packets from a faster path are lost and recovered by a slower path, causing packet reordering. Therefore, a framework that can handle packet reordering is needed.

3. Proposed methodology

This research assumes a system designed to improve the quality of service for remote control services such as robots, autonomous driving, and UAVs. To ensure reliability within this system, it will be connected to the networks of two MNOs, where one operates in an unstable mobile network environment while the other operates in a stable mobile network environment. The unstable mobile network environment is characterized by fluctuating communication delays and frequent packet losses due to handovers and weak signals, including bursts of losses occurring within seconds. The system in an unstable mobile network environment can connect to multiple mobile networks. The communication transmitted and received is real-time media traffic, where packet loss significantly degrades the quality. It is assumed that the connecting mobile networks support IPv6 and take into account the presence of stateful firewalls in the communication path.

3.1. System framework

Fig. 2 shows the overview of the proposed framework. The proposed framework consists of a Mobile Router (MRT) deployed in an unstable mobile network environment and a Central Router (CRT) deployed in a stable network environment. Each router (RT) is equipped with interfaces connected to a General Link and one or more Tunnel Links. RTs receive packets transmitted by regular hosts over the General Link and duplicate these packets. Duplicated packets are encapsulated using Generic Network Virtualization Encapsulation (GENEVE) [30] and transmitted through the Tunnel Links. By appending sequence numbers to the GENEVE option header, it is possible to identify which

Table 1
Software implementation in this study.

OS	Ubuntu 20.04.5 LTS
golang	v1.21.1 linux/amd64
gopacket	v1.1.20-0.20220810144506

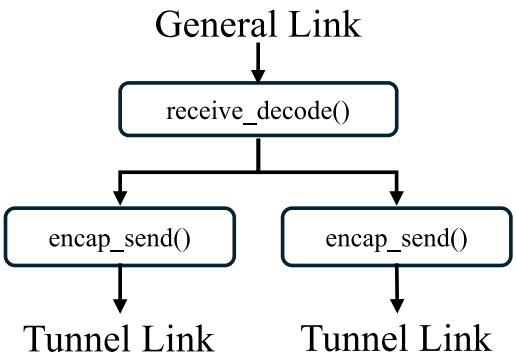


Fig. 3. Design of transmitter.

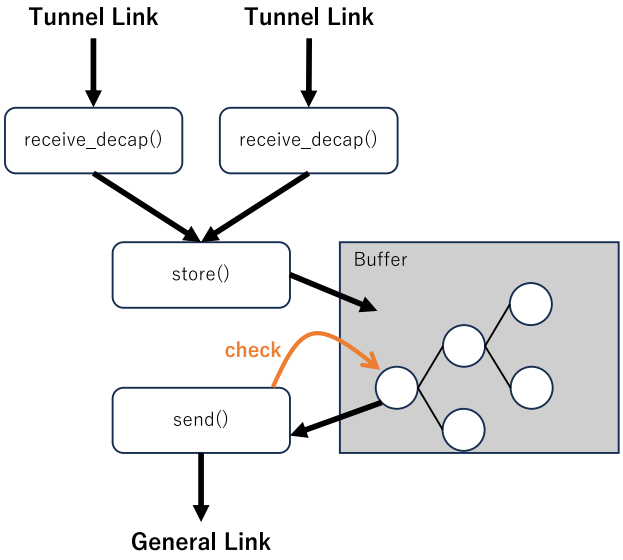


Fig. 4. Design of receiver.

packets are identical. The opposite RT receives the encapsulated packets through the Tunnel Link and checks the sequence numbers in the header. If the sequence number has already arrived, the packet is discarded; otherwise, it is decapsulated and placed in the internal buffer. RT extracts the packet with the lowest sequence number from the buffer and transmits it through the General Link. If the sequence number of the packet to be transmitted is more than two greater than the previous one, it waits a certain amount of time for the missing packets. If they still do not arrive, it advances to the next sequence number. This method achieves communication redundancy via middleboxes without requiring any changes to the application. Furthermore, the framework's tunneling encapsulates IP packets in GENEVE, making it independent of specific transport and higher layer protocols. Since GENEVE operates over UDP, intermediate nodes treat it as UDP packets, allowing it to traverse existing infrastructure.

3.2. Implementation

The proposed framework is implemented to run a common program on the MRT and the CRT, involving two processes running in each router: Receiver and Transmitter. The communication between the threads is primarily through channels that trigger each thread's operation when data arrives. This approach eliminates the need for busy loops throughout the system. The framework was implemented using the Go programming language, with all components developed in userspace. Packet processing uses the gopacket library [31]. The implemented software versions, including operating system (OS), are summarized in Table 1.

The transmitter encapsulates packets from the General Link and sends them through the Tunnel Link. Its design is illustrated in Fig. 3. Upon startup, the Transmitter initializes a sequence number to zero and maintains it internally. The sequence number is incremented with each received packet. After decoding a packet, the thread `encap_send()` is launched for each Tunnel Link. In `encap_send()`, a capsule header is created using the associated Tunnel Link's IP and MAC addresses, enveloping the decoded packet for transmission. Destination IP addresses for each Tunnel Link are statically set via a configuration file, while other information is retrieved from interface data. Packet encapsulation and transmission, requiring significant processing, are parallelized across Tunnel Links.

Conversely, the receiver decapsulates packets received from the Tunnel Link, buffers them, and transmits them over the General Link, as shown in Fig. 4. The receiver must maintain a sequence table to manage the sequence numbers of received packets. Since the sequence table is accessed by multiple threads, it requires exclusive control features, in this case implemented with an in-memory. Given the potential for multiple Tunnel Links, the `receive_decap()` thread for receiving packets is also plural. In `receive_decap()`, packets are decoded to obtain sequence numbers from the GENEVE option header. The sequence table is updated when the sequence number arrives for the first time. The packet, sequence number, and time of receipt tuple is then passed to the `store()` thread. However, if the sequence number has already arrived, the packet is discarded and processing moves to the next packet. `store()` asynchronously receives data from multiple threads and continuously inserts it into a buffer. The buffer, which contains tuples of packets, sequence numbers, and receive times, is managed as a heap sorted by sequence number. As a heap, it rearranges its elements in logarithmic order on each insertion or retrieval, allowing constant time access to the minimum value. Buffer timeout is handled by the thread `send()`, shown in Fig. 5 for processing a single packet. First, the minimum element of the buffer (the one with the smallest sequence number) is checked (`GetMin()`) without extraction. As described, if the sequence number of the minimum element is immediately after the sequence number of the last transmitted packet, it is sent immediately; if not, it is discarded if smaller, or waited if larger. If waiting, a timer is set after `waitTime`, during which the minimum value of the buffer is waited for. `waitTime` is the lesser of `MaxBufferTime` (set at receiver startup) and `limitTime -- now`. `limitTime` starts with a high enough value. If the buffer's minimum value is updated before `waitTime` passes, `limitTime` is updated to the packet's receive time + MaxBufferTime, and the process revisits the buffer's minimum element. This update prevents delaying the transmission of packets that have already arrived for more than `MaxBufferTime`, even if later packets arrive out of sequence. After waiting `waitTime`, `limitTime` is reset to a high value and the packet is sent. Also, dummy packets are sent once per second for hole-punching to maintain connectivity.

4. Evaluation

To verify that the proposed framework solves the problem of interference with application congestion control, which was a problem with existing methods, live streaming communication was conducted from a vehicle traveling in an urban area (Shinjuku, Tokyo, Japan) using the proposed framework. A live streaming system developed with

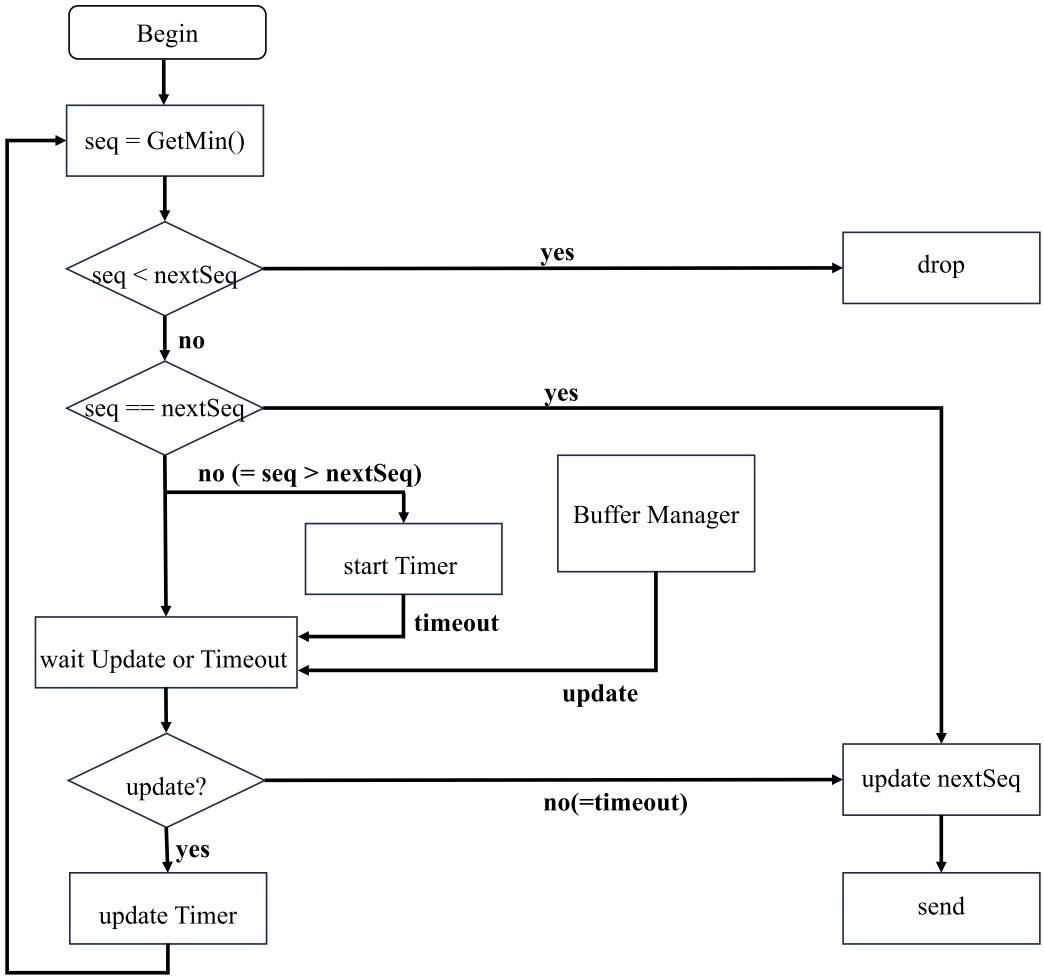


Fig. 5. Flow chart of buffering implementation.

WebRTC using the proposed framework was used for the measurement, and peer-to-peer (P2P) communication was performed between a PC in the vehicle and another PC in a data center. The proposed framework was evaluated based on statistical information obtained from the WebRTC API and information about packets processed by the proposed framework. For comparison purposes, we implemented a conventional study [16] that is similar to our framework but lacks the packet reordering feature. This conventional study sends packets from the General Link to the client PC in the order they are received from multiple Tunnel Links. Consequently, from a WebRTC perspective, packet reordering occurs, causing congestion control to reduce the bitrate.

4.1. Experiment setup

The measurement network setup is illustrated in Fig. 6. This figure shows a configuration connecting two mobile networks of 4G and 5G (Sub6 n 77/n 78/n 79 and millimeter wave n 257). In this experiment, we use commercial mobile networks for which we have no insider view. The have no insights about the setup of the networks, the base stations within each MNO, and we not collecting any logs on the mobile router side. For this reason, we carry out verifications from the perspective of the user application. The MRT operates with two network namespaces: the default namespace and an additional namespace called subNS. In the default namespace, the software components of the proposed framework, namely the transmitter and receiver, are active. The subNS provides internet connectivity to the client within the vehicle, functioning as a router. The MRT is connected to two different mobile networks via two Tunnel Links. The addresses for these interfaces are set via Router Advertisements sent by the mobile router. These two namespaces are connected by a virtual link, designated as the General Link. The virtual link's address uses the Unique Local Address space of fd00::/64. The local network within the vehicle uses the address space of 2001:200:0:1cdc::/64. In subNS, radvd is utilized to advertise this prefix, and ip6tables is employed for MSS (Maximum Segment Size) clamping to ensure that the MSS for TCP communications is set to 1340. This adjustment results in a client PC's TCP communication MTU being 1400 bytes, with packets sent from the MRT at 1464 bytes, below the commonly acceptable size of 1500 bytes for internet transit. In subNS, the default route is directed towards the virtual link, ensuring that all traffic from the client within the vehicle's local network uses the proposed framework for communication. Meanwhile, the CRT has each Tunnel Link as well as General Link. The Tunnel Link has the address of 2001:200:0:1cd1::6666:92, and the General Link has the address of 2001:200:0:1cd2::6. Although the client PC also connects to the 2001:200:0:1cd1::/64 network, it only needs to ensure internet connectivity. The internet connectivity for the client PC is provided by an external router, meaning the CRT does not require multiple network namespaces. To facilitate communication from the internet to the vehicle's local network, routing was set up by the external router to direct traffic for 2001:200:0:1cdc::/64 to the General Link (see Fig. 7). In addition, the hardware setup is shown in Fig. Fig. 7. Figure (a) shows the vehicle side and Figure (b) shows the data centre.

A separate service was established to facilitate WebRTC communication between a PC inside a vehicle and a PC within a data center. The system's architecture is illustrated in Fig. 8. All components of

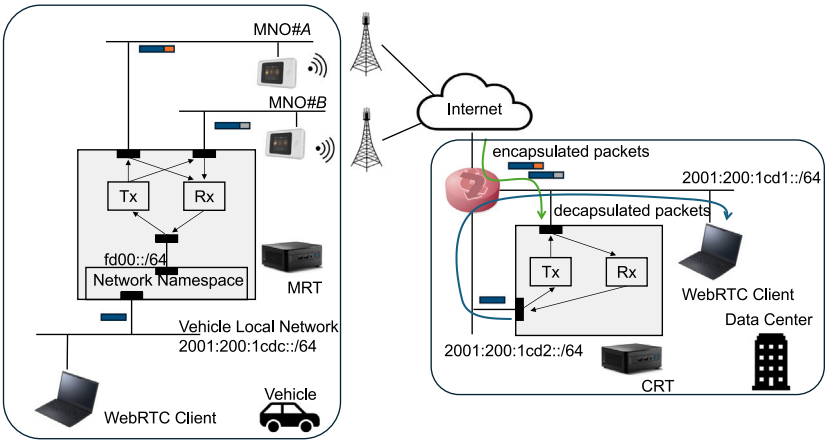
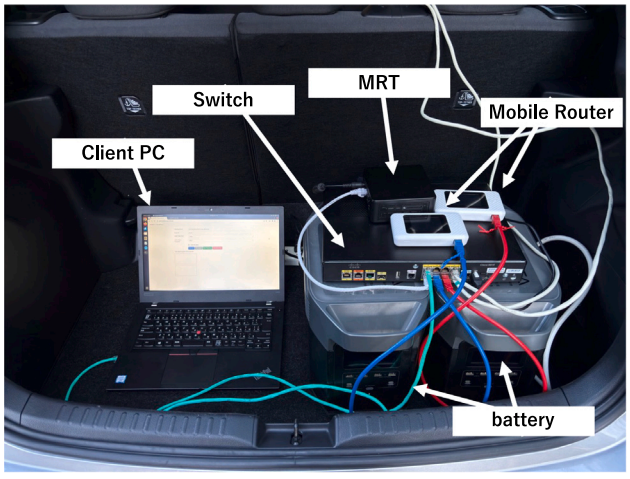
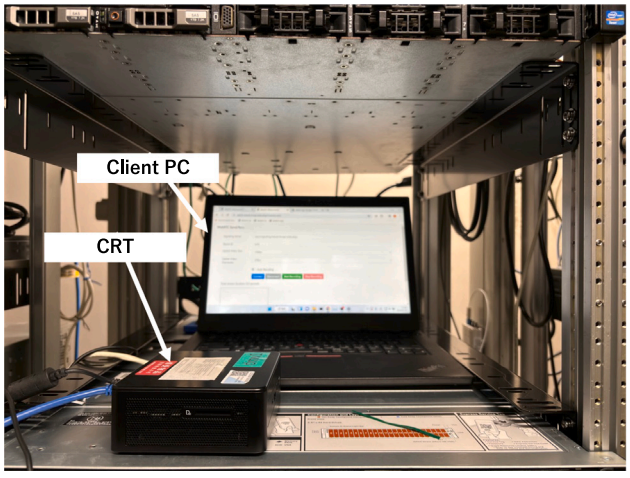


Fig. 6. Measurement network setup.



(a) Vehicle.



(b) Datacenter.

Fig. 7. Hardware configuration setup.

this system were deployed using Docker containers, which include a reverse proxy (nginx), a web server providing the WebRTC client program, and a signaling server. The nginx container serves multiple roles, including encryption for HTTPS and WebSocket over SSL/TLS, access logging, authentication, and site distribution based on domain.

Table 2
Software version of WebRTC.

docker	20.10.21
nginx	1.23.0
Vite	4.0.4
ayame–web–sdk	2022.1.0
ayame	2022.2.0
Chrome (PC in vehicle)	120.0.6099.71
Chrome (PC in datacenter)	120.0.6099.217

Table 3
Comparison of packet statistics (Shinjuku).

	MNO#1	MNO#2	Proposal
Transmitter packet count	170,445	170,445	170,445
Packet loss count	3542	25,462	4
Packet loss ratio [%]	2.078	14.938	0.002
Adopted packet count	123,619	46,822	–
Adopted packet ratio [%]	72.53	27.47	–

The WebRTC program is written in JavaScript, enabling the transmission of media traffic from devices running this frontend program. The web server lacks backend processing capabilities, serving only static HTML and JavaScript files to access hosts. The web server's interface is hosted using Vite. Ayame by Shiguredo [32] was employed for the signaling server. Hosts that receive the WebRTC client source from the web server connect to the signaling server, exchanging information to initiate P2P communication between hosts. The software versions used are listed in Table 2. The WebRTC streaming service collected statistics for inbound–rtp, outbound–rtp, remote–inbound–rtp, and remote–outbound–rtp every second via `getStats()` API. VP9 was used for video compression, and Opus was used for audio. The video resolution options available were 1920×1080, 1280×720, 854×480, and 426 × 240, with only 1920 × 1080 used for this study's measurements. However, the resolution can automatically adjust based on communication quality. The frame rate is also automatically regulated by WebRTC's adaptive bitrate.

In addition, Fig. 9 shows driving course of experiments. Fig. 9(a) shows the course in Shinjuku, and Fig. 9(b) shows the course in Shibuya. In both areas, the car was driven at a speed of 0–60 km/h. In Shinjuku, the car drove a distance of about 2.9 km, and in Shibuya, it drove a distance of about 10.6 km.

4.2. Results

This experiment is two-fold. First, we evaluated the performance of our proposal by operating two client PCs in a vehicle, each communicating with the same client PC in a data center. One client PC

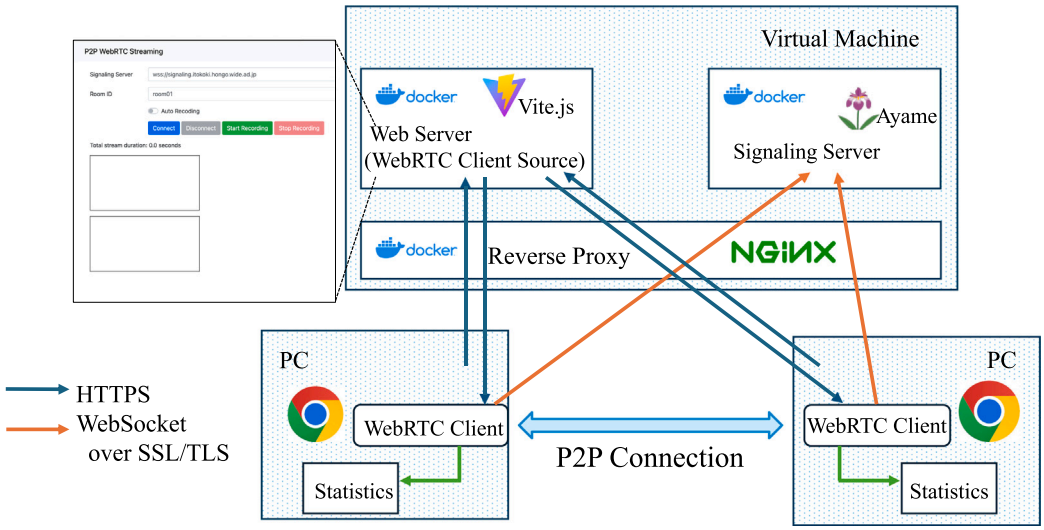


Fig. 8. Configuration of WebRTC streaming.

Table 4
Comparison of packet statistics (Shibuya).

	MNO#1	MNO#2	Proposal
Transmitter packet count	692,708	692,708	692,708
Packet loss count	28,360	27,570	14
Packet loss ratio [%]	4.094	3.980	0.002
Adopted packet count	373,623	319,071	–
Adopted packet ratio [%]	53.94	46.06	–

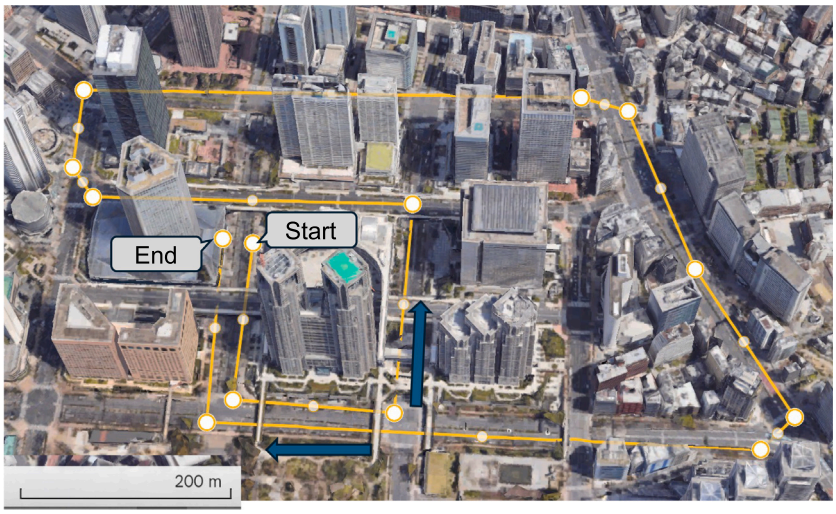
communicated using the proposed framework, while the other communicated using a system that lacked the proposed framework's packet reordering buffering feature (conventional method). We set the maximum buffer time to 300 ms. Since redundant communication was performed on each client PC, two streams flowed through each mobile router. The purpose of this experiment was to verify that (i) multipath redundant communication reduces packet loss and (ii) reordering improves bitrate. Packet loss was evaluated using packet logs passing through the CRT, and bitrate was evaluated using WebRTC statistical information. This experiment was conducted in Shinjuku, Tokyo, and Shibuya, Tokyo. In Shinjuku, the experiment was evaluated after 15 min, and in Shibuya, it was evaluated after 40 min. In the second experiment, only one client PC was operated in the vehicle, and the maximum buffer time was varied for each loop. The maximum buffer times tested were 10 ms, 100 ms, 300 ms, 500 ms, and 1,000 ms. This experiment evaluated the relationship between maximum buffer time and bitrate and was conducted at a single location (Shinjuku, Tokyo). In both experiments the vehicle was driven between 0 and 40 km/h.

The first experiment's results, which used the proposed framework for communication, showed the number of packets sent to each mobile network, the loss rate, and the number of packets adopted, which are summarized in Tables 3 and 4. In both tables, the "adopted packets" refer to the number of packets with the same sequence number that arrived first and were transmitted from the General Link in redundant communication. In Table 3, packet loss in mobile network links showed about 2.078% in MNO#1 network and 14.938% in MNO#2. The proposed framework reduced the packet loss rate to 0.002%. Similarly, in Shibuya (Table 4) the packet loss of the proposed framework is better than MNO#1 and MNO#2. These results show that reliability has improved with the proposed framework. There are several possible reasons for the high packet loss of MNO#1 and MNO#2 are high. In this experiment, we used commercially available SIM cards and conducted the experiment using two MNOs. In the MNOs, we were not sure whether they were 5G NSA or LTE based on 3GPP Release 15 or Release 16. In such cases, the current 5G NSA and LTE cannot prevent the occurrence of Radio Link Failure (RLF) (i.e., HO failure) due to sudden changes in the radio environment during HO preparation or HO execution. As a result, the amount of packet loss due to handover failure increases. This RLF problem occurs during HO and depends on the deployment of the base stations, so the locations where it occurs differ from MNO to MNO. Therefore, by using two or more MNOs, it is possible to increase the diversity gain so that it can be said that packet loss has improved. The remaining of this section provides detailed analysis explaining these results.

Fig. 10 shows the relationship between sequence numbers and the cumulative number of lost packets across different mobile networks and redundant communications in Shinjuku and Shibuya, Tokyo. Packet losses do not occur uniformly throughout the measurement but tend to cluster within certain time intervals. In Fig. 10(a), MNO#1 experiences no packet loss in the range from sequence number 0k to 90k. Therefore, packets that passed through MNO#1's route are selected. However, MNO#1 experiences packet loss in the sequence number range 90k–100k, but MNO#2 does not, so the packet loss that occurred on MNO#1's route is selected. There are more packet losses in the sections with sequence numbers above 100k, which is why the total number of packets accepted by MNO#1 exceeds that of MNO#2. On the other hand, Fig. 10(b) shows that packet loss is greater for MNO#1 for packets with sequence number below 100k. From 380k onwards, MNO#2 deteriorates rapidly, and after 400k both feature similar packet loss. Overall, the packet loss is almost the same, and MNO#2 has a slightly lower packet loss.

Fig. 11 illustrates the relationship between sequence numbers and the cumulative number of packets adopted. In Fig. 11(a), for the interval from 20k to 50k, where no packet loss occurs on either MNO#1 or MNO#2 links, the slope of the cumulative adopted packets is steeper for MNO#1, indicating a tendency toward lower latency for MNO#1. Fig. 11(b) shows that the cumulative number of adoptions for both sides is increasing at a similar rate. This result suggests that it is challenging to continue to predict a path with little delay at each moment in the mobility scenario of driving in the city.

From the above results in Shinjuku, the number of packets adopted did not increase in the section where packet loss occurred on one of the mobile networks. In such sections, there was a significant difference in the quality of the two mobile network links. This change in quality



(a) Drinving course of Shinjuku, Tokyo.

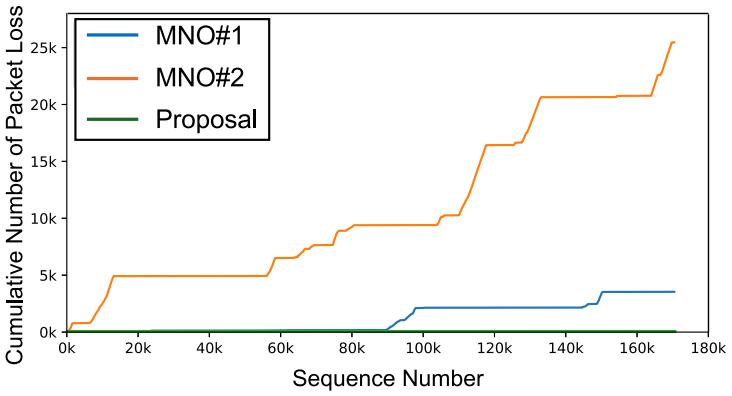
(b) Drinving course of Shibuya, Tokyo.

Fig. 9. Experiment conditions.

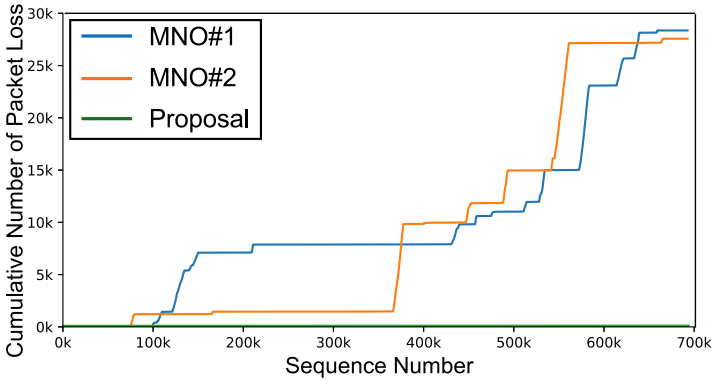
was compensated by redundant communication. On the other hand, in Shibuya, the number of packets adopted continued to increase even in the section where packet loss occurred, suggesting that the packet loss was intermittent rather than continuous. In such cases, packet loss recovery methods using coding techniques such as forward error correction (FEC) are also effective. In mobile networks, intermittent and constant packet loss occurs, and we confirmed that redundant communication is an effective method for dealing with both.

In addition, the results comparing the bitrate of WebRTC communications using the proposed framework and communications without reordering (conventional [16]) are shown in Fig. 12. Both figures show that the bitrate is higher with the proposed framework than the conventional method [16]. The number of times reordering occurred in the communication without reordering was 7,155 in Shinjuku and 33,165 in Shibuya, respectively. The percentage of the total number of packets was 4.20% for Shinjuku and 4.79% for Shibuya. The number of order changes was calculated by counting the number of times that the sequence number corresponding to the packet sent from the General Link of the CRT to the client PC did not increase monotonically. both results confirm that multipath redundant communication can lead to packet reordering and that the proposed framework can mitigate the bitrate reduction caused by such reordering.

The results of the second experiment in Shibuya, Tokyo, showing the bitrate for each maximum buffer time, are shown in Fig. 13. This figure shows an improvement in bitrate compared to a single mobile network (MNO#1 or MNO#2) in all cases due to the reduction of packet loss due to redundant communications. The higher bitrate compared to the first experiment is because two video streams were flowing through each mobile network in the first experiment. In contrast, there was only one stream in this experiment.



(a) Result of Shinjuku, Tokyo.



(b) Result of Shibuya, Tokyo.

Fig. 10. Number of packet loss.

Table 5
Maximum buffer time (T) vs. timeout occurrences count.

Maximum buffer time (T)	10	100	300
Transmitter packet count	211,694	249,408	602,361
Packet loss count	2	141	0
Timeout count	706	15	3
Discarded packet count	1027	23	14

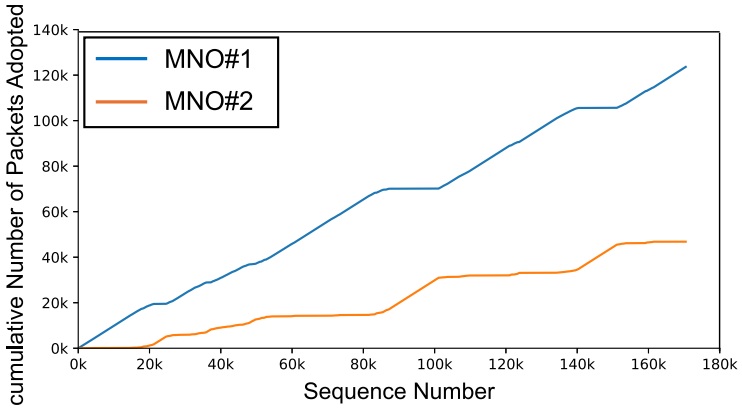
The number of packet losses, timeouts, and packets discarded by the system for communications using different maximum buffer times are summarized in Table 5. The system-discarded packets refer to those arriving after a sequence number has been skipped due to timeouts and thus discarded. The measurement with a maximum buffer time of 300 ms was conducted twice, resulting in more transmitted packets. With a maximum buffer time of 10 ms, packet loss recovery for one link did not occur, resulting in 1,027 packets being discarded. From the client PC's perspective, this discarding by the framework is equivalent to packet loss, contributing to a reduction in bitrate. In measurements with a 100 ms maximum buffer time, packet losses co-occurred on both MNO#1 and MNO#2 links, increasing the timeouts. Almost no timeouts occurred when we set the maximum buffer time to 300 ms or more, and there was no significant difference in bitrate. These results suggest that providing a buffer over 100 ms in multipath redundant communication can mitigate bitrate reduction.

4.3. Discussions

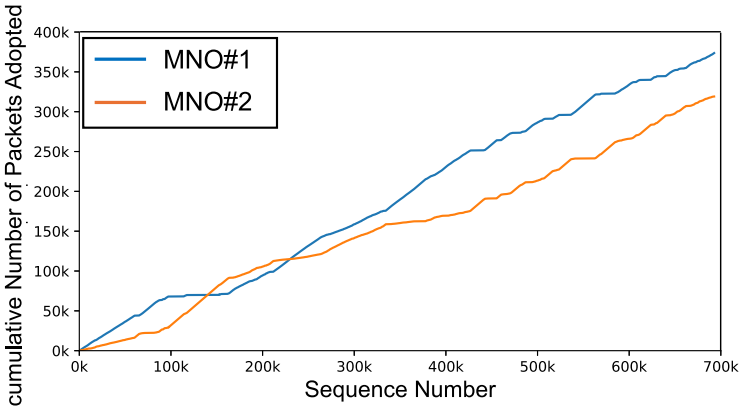
The above results focus primarily on improving reliability and bitrate, but the proposed framework has unexpected limitations. While it improves reliability, it also introduces some latency, mainly due to the overhead of the system. The results of the round-trip delay measurements in Shinjuku, Tokyo, are shown in Fig. 14. The results of the proposed method presented here are for when the maximum buffer time is set to 300 ms. It was also found that changes in the maximum buffer time made almost no difference to the delay. When communicating over a single carrier, where MNO#1 had smaller delays, 75% of the measured values were below 50 ms. In contrast, with the proposed framework the delay that accommodates 75% of the values is 85 ms, which is 38 ms higher than MNO#1 in both the 75% interval and the median. The delay measured here is round-trip time, so the one-way overhead of the framework is estimated at around 19 ms. The main cause of this delay is the significant processing delays in the kernel buffer. Although it is important to ensure low delay while improving reliability, we leave for future work the implementation and analysis using fast packet processing frameworks (e.g., DPDK). At this point, we will continue our analysis of the state of the art.

5. Conclusion

This study proposed a redundant communication framework designed to improve the quality of real-time media streaming communication. It addresses the limitations of transport protocols in traditional redundant communication systems by using GENEVE for tunneling at



(a) Result of Shinjuku, Tokyo.



(b) Result of Shibuya, Tokyo.

Fig. 11. Number of packets adopted.

the IP layer. In addition, it resolves application congestion control interference due to packet order inconsistencies by implementing buffering at the receiver side. We have demonstrated significant packet loss reductions and bitrate improvements compared to existing redundant communication systems by using multiple mobile networks from a vehicle in two real environments. These results confirm the effectiveness of the proposed framework.

However, the overhead of the implemented system is currently around 40 ms, and while there are benefits to the improved reliability provided by redundant communication, this is also a drawback. This is particularly important in real-time media communication, where end-to-end latency is kept at around 100 to 150 ms, and reducing the overhead is essential in future efforts. Furthermore, this research focuses only on the data plane of redundant communication. In actual operation, it is necessary to exchange the IP addresses of the GENEVE tunnel endpoints and to communicate the maximum buffer time according to the quality of communication. This research has shown that a short maximum buffer time has a negative impact on the communication bitrate, but further investigation is needed to determine the optimal maximum buffer time. In addition, in the proposed redundancy communication, the method of reducing the frequency utilization efficiency is used. Therefore, it is necessary to perform redundant communication only in specific areas where reliability is degraded.

CRediT authorship contribution statement

Koki Ito: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Jin Nakazato:** Writing – review & editing, Writing – original draft, Validation, Methodology, Data curation. **Romain Fontugne:** Writing – review & editing, Writing – original draft, Validation, Supervision, Investigation. **Manabu Tsukada:** Writing – review & editing, Validation, Methodology. **Esaki Hiroshi:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

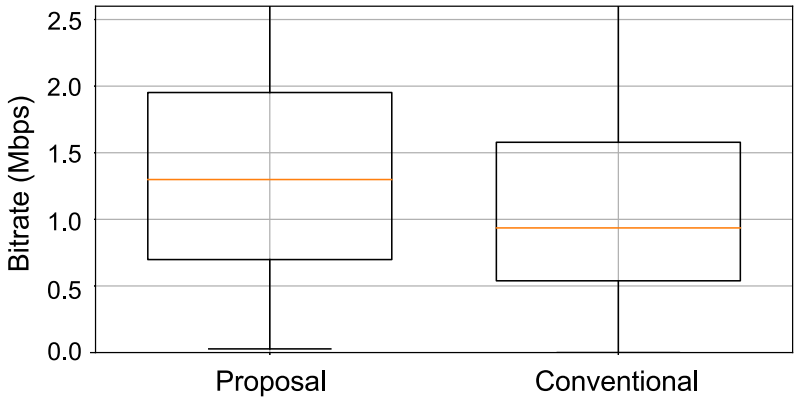
The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Manabu Tsukada reports financial support was provided by Japan Science and Technology Agency. Hiroshi Esaki reports financial support was provided by Japan Science and Technology Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

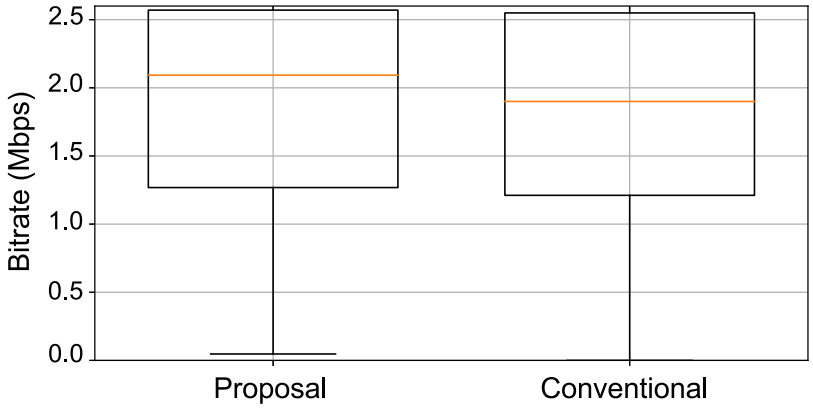
This research was supported in part by JST, Japan, CREST, Japan Grant Number #JPMJCR22M4; and in part by JST ASPIRE Grant Number #JPMJAP2325.

Data availability

No data was used for the research described in the article.



(a) Result of Shinjuku, Tokyo.



(b) Result of Shibuya, Tokyo.

Fig. 12. Comparison of bitrate of WebRTC.

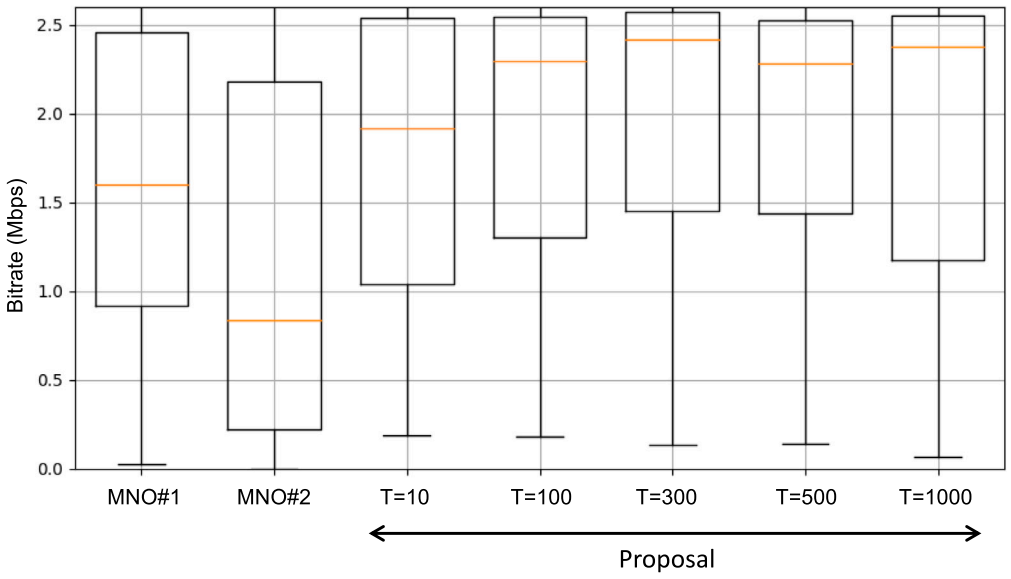


Fig. 13. Relationship between maximum buffer time (T) and bitrate of WebRTC.

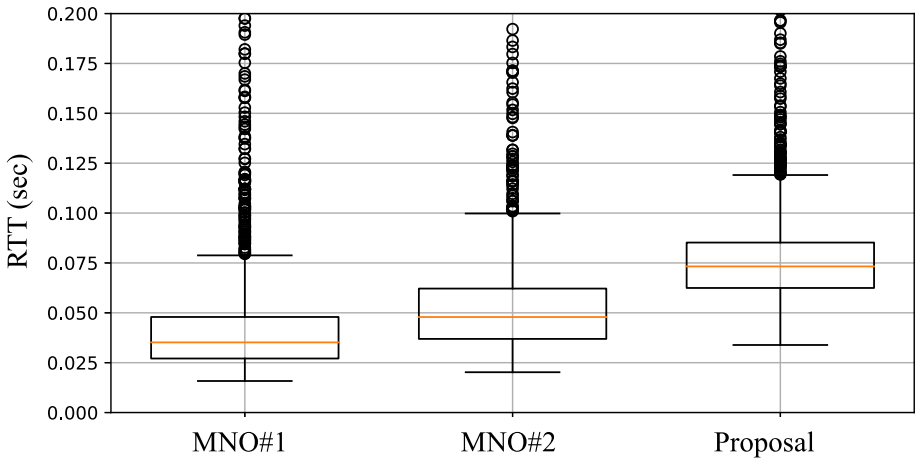


Fig. 14. Comparison of latency.

References

[1] Elias G. Carayannis, Joanna Morawska-Jancelewicz, The futures of Europe: Society 5.0 and industry 5.0 as driving forces of future universities, J. Knowl. Econ. 13 (4) (2022) 3445–3471.

[2] The Global Internet Phenomena Report 2023, Vol. 1, Technical Report, Sandvine, 2023.

[3] James Nightingale, Pablo Salva-Garcia, Jose M. Alcaraz Calero, Qi Wang, 5G-QoE: QoE modelling for ultra-HD video streaming in 5G networks, IEEE Trans. Broadcast. 64 (2) (2018) 621–634.

[4] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Eric Pujol, Igmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, Oliver Hohlfeld, Georgios Smaragdakis, Implications of the COVID-19 pandemic on the internet traffic, in: Broadband Coverage in Germany; 15th ITG-Symposium, 2021, pp. 1–5.

[5] Cisco webex helps customers stay remotely connected and reimagine work, 2020, https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2020/m06/cisco-webex-helps-customers-stay-remotely-connected-and-reimagine-work.html. (Accessed 22 January 2024).

[6] 2020-Zoom blog, 2020, https://blog.zoom.us/ja/2020/. (Accessed 22 January 2024).

[7] Daquan Feng, Lifeng Lai, Jingjing Luo, Yi Zhong, Canjian Zheng, Kai Ying, Ultra-reliable and low-latency communications: applications, opportunities and challenges, Sci. China Inf. Sci. 64 (2) (2021) 120301.

[8] Yu Asabe, Ehsan Javanmardi, Jin Nakazato, Manabu Tsukada, Hiroshi Esaki, Enhancing reliability in infrastructure-based collective perception: A dual-channel hybrid delivery approach with real-time monitoring, IEEE Open J. Veh. Technol. 5 (2024) 1124–1138.

[9] Here's how you used Zoom in 2022 - Zoom Blog, 2022, https://blog.zoom.us/how-you-used-zoom-2022/. (Accessed 22 January 2024).

[10] Jin Nakazato, Kousuke Nakagawa, Koki Itoh, Romain Fontugne, Manabu Tsukada, Hiroshi Esaki, WebRTC over 5G: A study of remote collaboration QoS in mobile environment, J. Netw. Syst. Manage. 32 (2023).

[11] HyunJong Lee, Jason Flinn, Basavaraj Tonshal, RAVEN: Improving interactive latency for the connected car, in: Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, MobiCom '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 557–572.

[12] Alexander Frommgen, Tobias Erbshäußer, Alejandro Buchmann, Torsten Zimmermann, Klaus Wehrle, ReMP TCP: Low latency multipath TCP, in: 2016 IEEE International Conference on Communications, ICC, 2016, pp. 1–7.

[13] Wei Wang, Liang Zhou, Yi Sun, Improving multipath TCP for latency sensitive flows in the cloud, in: 2016 5th IEEE International Conference on Cloud Networking, Cloudnet, 2016, pp. 45–50.

[14] Yihua Ethan Guo, Ashkan Nikravesh, Z. Morley Mao, Feng Qian, Subhabrata Sen, Accelerating multipath transport through balanced subflow completion, in: Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 141–153.

[15] Alan Ford, Costin Raiciu, Mark J. Handley, Olivier Bonaventure, Christoph Paasch, TCP extensions for multipath operation with multiple addresses, 2020, RFC 8684.

[16] Kaneko Naoya, Ito Takahiro, Katsuda Hajime, Watanabe Toshinobu, Abe Hiroshi, Onishi Ryokichi, Applying and evaluating multipath redundant communication technology for WebRTC-based video streaming, J. Digit. Pr. 3 (3) (2022) 21–31.

[17] Koki Ito, Jin Nakazato, Romain Fontugne, Manabu Tsukada, Hiroshi Esaki, Enhancing real-time streaming quality through a multipath redundant communication framework, in: 2024 IFIP Networking Conference, IFIP Networking, 2024, pp. 1–10.

[18] Jana Iyengar, Martin Thomson, QUIC: A UDP-based multiplexed and secure transport, 2021, RFC 9000.

[19] Yanmei Liu, Yunfei Ma, Quentin De Coninck, Olivier Bonaventure, Christian Huitema, Mirja Kühlewind, Multipath Extension for QUIC, Internet-Draft draft-ma-quic-mpqoe-01, Internet Engineering Task Force, 2023, in preparation.

[20] Tobias Viernickel, Alexander Froemmgen, Amr Rizk, Boris Koldehofe, Ralf Steinmetz, Multipath QUIC: A deployable multipath transport protocol, in: 2018 IEEE International Conference on Communications, ICC, 2018, pp. 1–7.

[21] Yunfei Ma, Yanmei Liu, Christian Huitema, Xiaobo Yu, An Advanced Scheduling Option for Multipath QUIC, Internet-Draft draft-ma-quic-mpqoe-01, Internet Engineering Task Force, 2022, in preparation.

[22] Joerg Ott, Mathis Engelbart, Spencer Dawkins, RTP Over QUIC (RoQ), Internet-Draft draft-ietf-avtcore-rtp-over-quic-07, Internet Engineering Task Force, 2023, in preparation.

[23] Luke Curley, Kirill Pugin, Suhas Nandakumar, Victor Vasiliev, Media Over QUIC Transport, Internet-Draft draft-ietf-moq-transport-01, Internet Engineering Task Force, 2023, in preparation.

[24] Matthieu Boutier, Juliusz Chroboczek, User-space multipath UDP in MOSH, 2015, arXiv preprint arXiv:1502.02402.

[25] Shaowei Liu, Weimin Lei, Wei Zhang, Hao Li, MPUDP: Multipath multimedia transport protocol over overlay network, in: 2017 5th International Conference on Machinery, Materials and Computing Technology, ICMMCT 2017, Atlantis Press, 2017, pp. 731–737.

[26] Daniel Lukaszewski, Geoffrey Xie, Multipath transport for virtual private networks, in: 10th USENIX Workshop on Cyber Security Experimentation and Test, CSET 17, USENIX Association, Vancouver, BC, 2017.

[27] Sally Floyd, Mark J. Handley, Eddie Kohler, Datagram congestion control protocol (DCCP), 2006, RFC 4340.

[28] ST 2022-7:2019 - SMPTE standard - Seamless protection switching of RTP datagrams, 2019, pp. 1–11, ST 2022-7:2019.

[29] Rei Nakagawa Tomoya Kawana, Communication multiplexing of server with QUIC and SDN in multihomed networks, in: The 38th International Conference on Information Networking, ICOIN2024, Ho Chi Minh City, Vietnam, 2024.

[30] Jesse Gross, Ilango Ganga, T. Sridhar, Geneve: Generic Network Virtualization Encapsulation, 2020, RFC 8926.

[31] Graeme Connell, gopacket. https://gitlab.com/google/gopacket.

[32] Shiguredo, OpenAyame, 2022, https://github.com/OpenAyame/ayame.