# Low Latency Redundant Network Architecture for Enhancing 5G Mobile Communication Quality

Nayuta Yamane*, Jin Nakazato*, Koki Ito*, Manabu Mikami**, Takahiro Tsuchiya**, Manabu Tsukada*, Hiroshi Esaki*
*Graduate School of Information Science and Technology, The University of Tokyo, Japan
**Advanced Technology Development Department., Research Institute of Advanced Technology, SoftBank Corp, Japan

Email: {e60nayuta, jin-nakazato, mtsukada}@g.ecc.u-tokyo.ac.jp, {manabu.mikami, takahiro.tsuchiya04}@g.softbank.co.jp, itokoki@hongo.wide.ad.jp, hiroshi@wide.ad.jp

***Abstract*—With a growing demand for high-capacity, low-latency communication driven by the increasing use of video streaming and real-time data applications, efficient data transfer methods have become essential. While 5G technology provides enhanced data transfer capabilities, its directional nature can lead to stability issues, including connection interruptions, frequent handoffs, data loss, and increased latency. To address these challenges, we propose a system architecture with a redundant configuration that separates LTE and 5G-SA within a single MNO, alongside a method to reduce overhead. Initial verification in a stationary environment demonstrated the reduction of latency by 38.1 ms compared to the conventional method, underscoring the potential of our approach for stable and efficient data transmission.**

***Index Terms*—5G, SA, LTE, mobile, redundant communication, low latency.**

## I. Introduction

While much research and development have been done regarding Beyond 5G technology, there has already been a consortium for standardization [1]. The concepts of eMBB (Enhanced Mobile Broadband), URLLC (Ultra-Reliable Low-Latency Communications), and mMTC (Massive Machine Type Communications) have been the main targets of consideration since 5G standardization. In these committees, not only are the 5G use cases being further investigated, but new scenarios that integrate multiple use case domains (e.g., Integrated Sensing and Communication (ISAC), AI, and Communication) are also emerging. These circumstances necessitate ultra-low-latency communication across a variety of applications [2]. For example, during the pandemic, many companies have moved to remote work, and meetings have gone online, making low-latency video communication essential to daily work. Additionally, for low-latency communication, requirements are being set from an End-to-End (E2E) perspective in terms of Quality of Service (QoS) and Quality of Experience (QoE) rather than just the latency of the radio segment discussed in 3rd Generation Partnership Project (3GPP). In addition, according to research in 2024 [3], approximately 70 % of the world network traffic is related to video, including on-demand streaming, such as Netflix, YouTube, and Tiktok, real-time broadcasting, e.g., concert live and videos from a drone, and online meetings such as Zoom, WebEx, and Google Meet. In addition, as autonomous driving technology progresses, it will not be unusual to participate in a Zoom session from a vehicle, leading to the need for wireless and mobility communication for video streaming. Thus, the demand for high-capacity and low-latency communication in mobile environments has been increasing because of the increase in video streaming and real-time data-driven applications.

However, in order to achieve these requirements, the effect of handover on the communication quality must be minimized. We have been investigating the effect and confirmed that when sending packets via a single mobile network operator (MNO) from a moving vehicle, latency and jitter degrade [4]. One approach to the problem is to use multiple MNOs or frequencies to provide redundancy to the communication path. There have been protocols designed for redundant communication. The most popular methods are multipath TCP (MPTCP) [5] and multipath QUIC (MP-QUIC) [6]. Although those methods are highly effective for famous use cases, including HTTP, they are not directly applicable to tightly connected or niche protocols such as WebRTC, which uses the User Datagram Protocol (UDP) protocol under a unique data format.

Therefore, we have proposed and researched a middlebox approach [7] that includes an IP layer tunneling using GENEVE protocol [8]. To demonstrate the effectiveness of this proposal, we conducted experiments in a real field environment that measure the quality of mobile communication between a moving vehicle and a static environment, and confirmed that redundancy during handovers resulted in an improvement in packet loss [7]. However, there were two problems; one of them was extremely high time overhead in our program, and the other was that we could not obtain detailed information inside the mobile network of MNOs, such as frequency, band, or service network (e.g., Long-Term Evolution (LTE), Non-standalone (NSA), standalone (SA)). Thus, we could not know or verify whether the chosen mobile networks are optimal from the viewpoint of practical frequency usage.

In order to tackle these problems, this research proposes a network architecture that divides the frequency in a single MNO. Specifically, we propose to use LTE and 5G-SA networks to make a redundancy. In addition, we reduce the overhead by improving the capsulation program to remove

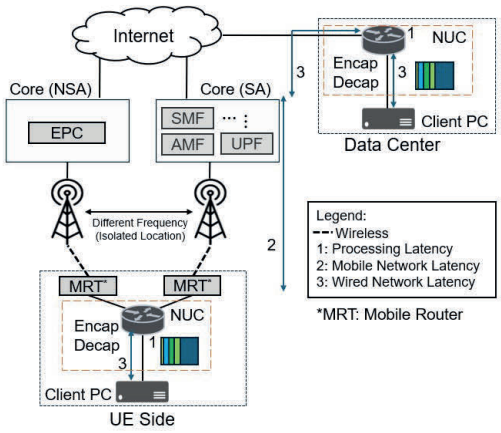



Fig. 1: System architecture of our proposed method.

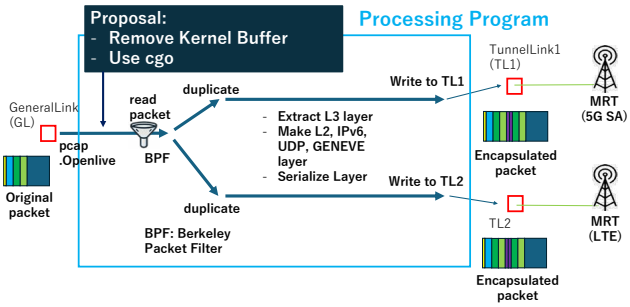



Fig. 2: Proposed software diagram.

TABLE I: Experiment Specification.

| Parameter | | Value |
|---|---|---|
| LTE | Operating Band | B8 |
| | Frequency | UL: 905–915 / DL: 950–960 MHz |
| | Bandwidth | 10 MHz |
| | Duplexing | FDD |
| | SCS | 15 kHz |
| 5G-SA | NR Band | Operating Band |
| | Frequency | 3.40–3.44 GHz |
| | Bandwidth | 40 MHz |
| | Duplexing | TDD |
| | SCS | 30 kHz |
| MRT [9] | Operating Band (LTE) | B1/B3/B8/B28/B41/B42 |
| | Operating Band (NR) | n3/n28/n77 |
| | Module | SIM8202G-M.2 (R2) |
| Encap Decap | CPU | Intel NUC |
| | Core | 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz |
| | Memory | 64 GB |

the buffering upon capturing packets to achieve low latency. To verify this, we experiment in collaboration with an MNO to identify the effects on the type of mobile network by measuring multiple metrics. This time, we focus on latency to compare the ping values in different networks and capsulation programs.

The structure of this paper is as follows: Section II describes the system architecture of the proposed method. Section III explains the system setup and the result and analysis of the experiment, and Section IV concludes the paper.

## II. System Architecture

### A. System Overview

Fig. 1 illustrates the system architecture of this study, which consists of four segments: the User Equipment side (UE side), mobile network, internet, and data center (DC). The UE side consists of the Client PC hosting applications such as WebRTC, a middlebox computer with the proposed redundancy software, and, two mobile routers (MRTs). The mobile network segment includes RAN, 4G EPC (NSA), and 5G-SA Core (e.g., AMF (Access and Mobility Management Function ), SMF (Session Management Function), UPF (User Plane Function)), each operating on distinct frequencies. The Internet serves as a bridge between the mobile network and the DC. On the DC side, the same proposed redundancy software as that on the UE side, is also deployed on a middlebox computer. Here, the UE side represents a device or a system that can be moved dynamically, whereas the data center is expected to be a static environment such as a content cache server. Even though the real use cases include data flow in both direction, from UE to DC and vice versa, this research focuses only on the UDP data transfer from UE to DC.

In the architecture, the packets (such as ping or WebRTC UDP packets) originating from the client personal computer (PC) in the UE side will be encapsulated at the middlebox device, which we call ”Encap”. The Encap device will intake the packets from the client PC and encapsulate them with the GENEVE header described in our previous paper [7]. The encapsulated packets are duplicated and sent to the data center via different mobile service networks. They are desnited to the other middlebox device called ”Decap” in DC, where the packets are decapsulated, put in the queue and sent to the client PC in DC, or the final destination. In this system, the round-trip time between two clients, that is, the time required for a packet to travel from the client PC in UE, to that in DC, and to the client PC in UE back again, is represented as the following formula:

$$T_{\text{E2E}} = 2 * T_{\text{program}} + T_{\text{mobile}} + T_{\text{wired}} \quad (1)$$

where $T_{\text{program}}$ is the latency attributed to the encapsulation and decapsulation process in the middlebox devices. $T_{\text{mobile}}$ is the latency in the mobile network. $T_{\text{wired}}$ is caused by network devices outside the mobile network the packets will flow, such as by switches and routers near the central environment or Internet Exchanges. By sending packets via multiple routes and because of Decap selecting the packets arriving fastest, $T_{\text{mobile}}$ would be the minimum of the mobile paths. Despite the additional latency $T_{\text{program}}$, it will be still useful in that it may give the users redundancy, which may lead to less latency in some congested areas where packet losses are quite often. As to the scalability, because the packets are encapsulated in the IP layer and above, a new device can easily be added in either UE or DC side just by connecting it to the respective network. Therefore, the architecture can handle a large number of users,

aside from the throughput constraint of the encapsulation devices.

### B. Proposed Redundancy Software

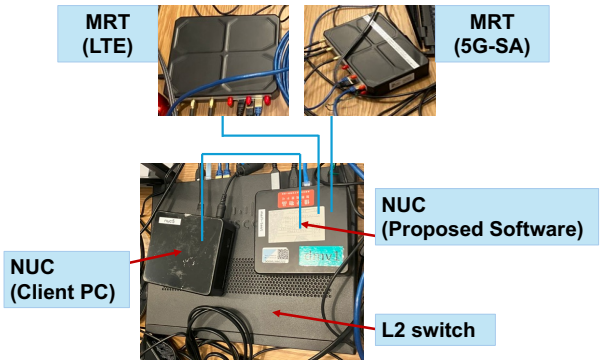
Fig. 2 represents the encapsulation program overview, written in Go. It reads the original packet using packet capture function to read packet and apply Berkeley Packet Filter (BPF). Then it will be duplicated and outer layers are added by the software. It encapsulates the IP layer and above, so the outer layers include GENEVE headers, Outer UDP layer, Outer IP layer. After the encapsulation, it will be written to tunnel interfaces. Besides the encapsulation program, there is also the decapsulation program, also written in Go. It reads from the uplink, decapsulates the packets, and stores the information of the latest sequence number it received. If the packet with larger sequence number arrives before the previous one, it waits a fixed amount of time before sending the packet in order to prevent triggering of congestion control. Then, the packet segment will be sent to the client PC in DC. These programs are based on our previous research [7].

The proposed redundancy software implementation is the removal of the kernel buffer. In our previous research, the latency was quite high for unknown reasons. This time, We identify the reason to be the buffering by the Linux system on packet capturing in the encapsulation and decapsulation programs. In the program, libpcap is used to read packets from an interface, and the packet buffering is activated by default in some newer version of Linux including our configuration [10]. Which causes packets not being delivered until sufficient amount of data arrive or the timeout, resulting in high latency. Notably, it happens four times in a round-trip in our architecture, as packets are read when encapsulated and decapsulated, both on the way and on the way back, so it increases latency intensively. Therefore, we simply disable the default buffering by setting the capture vhandle to immediate mode.

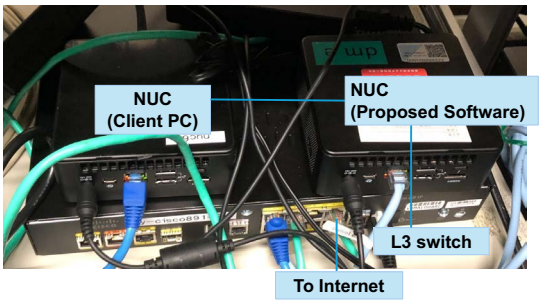
## III. Experiment Results

### A. Experiment Setup

Fig. 3 illustrates the physical setup of this experiment. We used two types of mobile networks, LTE and 5G-SA in a single MNO. All specifications are summarized in Table I. An Internet Control Message Protocol (ICMP) packet is sent from the UE side client PC to that on the DC side and the latency will be measured at the UE side client PC. On the UE side, there is a monitoring apparatus that is attached to a mobile router, which obtain metrics such as Reference Signal Received Power (RSRP) and Signal-to-Interference-plus-Noise Ratio (SINR). The capsulation programs are all written in Go. The minimum ping frequency is set to 9 ms and we compared the conventional version used in our previous research [7] with the improved version. In order to separate the latency into network-related and program-related one, the latency between the tunnel links of two middleboxes are also measured independently.




(a) UE side.




(b) Data center.

Fig. 3: Physical configuration of the experiment.

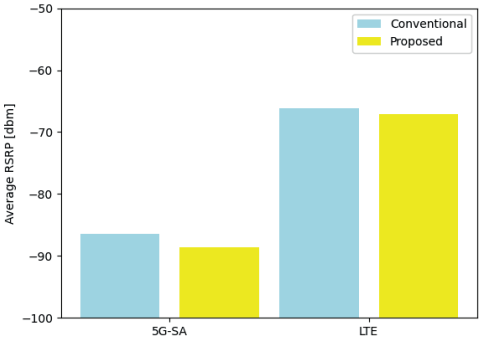



Fig. 4: RSRP values: LTE and 5G-SA.

### B. Preliminary Results

To compare latency between the proposed method and the conventional method, we conducted two rounds of experiments, measuring the state of LTE and 5G-SA mobile networks during each round. The average RSRP and SINR values are shown in Figs. 4 and 5, respectively. While there were no significant differences in SINR under any conditions, the RSRP was higher for LTE, indicating stronger signal strength for LTE in a static environment. Additionally, the proposed method recorded a slightly lower RSRP, but they would be attributed to the differences in the field environment. More trials are needed in the future research. RSRP indicates the signal strength between the base station and

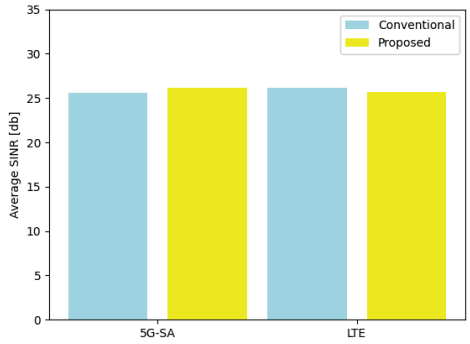



Fig. 5: SINR values: LTE and 5G-SA.

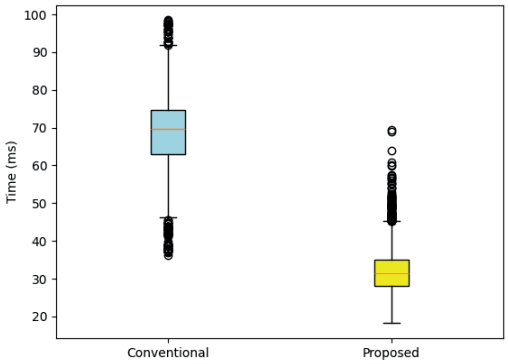
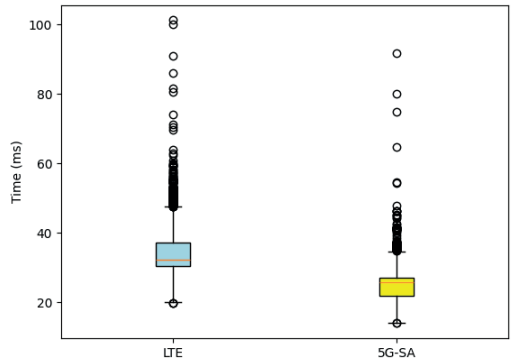


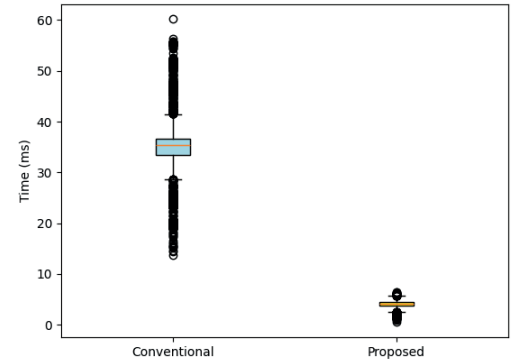

Fig. 6: End-to-End latency Results: Conventional vs. Proposal.

the terminal, while SINR affects throughput. Consequently, although there was no difference in throughput in the two measurements between LTE and 5G-SA, it is thought that differences in base station and terminal configurations, as well as higher propagation loss for 5G-SA compared to LTE, may contribute to the observed results. Fig. 6 shows the evaluation results of E2E latency in the conventional and the proposed network architecture, demonstrating that the proposed method significantly improves latency. Using the proposed method, the median latency is reduced by 38.1 ms (from 71.2 ms to 33.1 ms), the minimum being 18.2 ms. This indicates that the proposed method outperforms conventional methods by suppressing overhead, enabling highly reliable and low-latency communication.

In the E2E latency results shown, Eq. (1) comprises three elements, but these can be categorized into two main roles: network and processing. Fig. 7 shows the results classified by these two roles. In Fig. 7(a), it can be seen that 5G-SA achieves lower latency than LTE in the network. This is because 5G-SA has a larger SCS than LTE, resulting in finer sub-frames, which enables lower latency in the wireless segment. Furthermore, LTE and 5G-SA differ in the core network, with 5G-SA separating the control plane and user plane, allowing a simplified user-plane-only configuration that facilitates low-latency communication. These factors enable 5G-SA to achieve lower latency communication than LTE. Next, as shown in Fig. 7(b), processing latency of the proposed method is 4.10 ms, which is 31.3 ms lower than the conventional method. This indicates that the improved program in the proposed method contributes significantly to latency reduction. However, the reduction range is smaller than E2E, suggesting that the redundant configuration of 5G-SA and LTE might have impacted the latency reduction in the network.




(a) Total Network Latency $T_{\text{mobile}} + T_{\text{wired}}$.




(b) Round-trip Processing Latency $2 * T_{\text{program}}$.

Fig. 7: Breakdown Latency Result of Network and Processing.

## IV. Conclusion

In this study, we proposed two key components: a system architecture that separates frequencies and networks (LTE, SA) within a single MNO and a method for reducing redundancy overhead. As a preliminary evaluation of the proposed approach, we conducted verification using a real network in a stationary environment, achieving a significant latency improvement by 38.1 ms. In the future, In the future research, we plan to evaluate other metrics including jitter, packet loss, and throughput with more trials. We also possibly investigate into error recovery mechanism or compare our system with other existing frameworks. We plan to implement this system

in vehicle-to-everything (V2X) scenarios and conduct testing in mobile environments.

## References


[1] J. T. J. Penttinen, "On 6g visions and requirements," *Journal of ICT Standardization*, vol. 9, no. 3, pp. 311–325, 2021.

[2] "Recommendation itu-r m.2160-0: Framework and overall objectives of the future development of imt for 2020 and beyond," International Telecommunication Union, Technical Report M.2160-0, 2023, accessed: 2024-10-01. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.2160-0-202311-I%21%21PDF-E.pdf

[3] "The global internet phenomena report 2024," Sandvine, Tech. Rep., 2024.

[4] J. Nakazato, K. Nakagawa, K. Itoh, R. Fontugne, M. Tsukada, and H. Esaki, "WebRTC over 5G: A study of remote collaboration qos in mobile environment," *Journal of Network and Systems Management*, vol. 32, 10 2023.

[5] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses," RFC 8684, Mar. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8684

[6] Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Multipath Extension for QUIC," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-multipath-06, Oct. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/06/

[7] K. Ito, J. Nakazato, R. Fontugne, M. Tsukada, and H. Esaki, "Enhancing real-time streaming quality through a multipath redundant communication framework," in *2024 IFIP Networking Conference (IFIP Networking)*, 2024, pp. 1–10.

[8] J. Gross, I. Ganga, and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation," RFC 8926, Nov. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8926

[9] SoftBank Corp, "CTL-6550," Accessed: Oct. 28, 2024. [Online]. Available: https://tm.softbank.jp/content/dam/common/services/iot-module/pdf/cidna-ctl-6550-catalog.pdf

[10] The Tcpdump Group, "pcap_set_immediate_mode(3pcap) man page," Accessed: Oct. 31, 2024. [Online]. Available: https://www.tcpdump.org/manpages/pcap_set_immediate_mode.3pcap.html