Towards URLLC with Open-Source 5G Software

Aoyu Gong[†], Arman Maghsoudnia[†], Raphael Cannatà[†], Eduard Vlad[¶], Néstor Lomba Lomba[†], Dan Mihai Dumitriu[♧], Haitham Hassanieh[†]
EPFL[†], RWTH Aachen[¶], Pavonis LLC[♧]

Abstract

Ultra-Reliable Low-Latency Communication is a key feature of 5G, yet achieving its strict one-way latency target remains challenging in real-world deployments. While previous work proposes latency-reduction techniques, most are theoretical or simulation-based and overlook practical bottlenecks in actual systems. In this paper, we analyze and optimize latency with open-source 5G RAN software. We characterize latency sources arising from 5G specifications and implementation-level factors, along with their complex interplays. Guided by this analysis, we introduce improvements reducing one-way latency by 39.28 % in the downlink and 55.38 % in the uplink. Our results show the importance of system-level experimentation and provide a blueprint for advancing toward URLLC targets in both 5G and future cellular networks.

CCS Concepts

• **Networks** → **Mobile networks**; **Network performance analysis**; **Network experimentation**; **Network design principles**.

Keywords

5G, URLLC, System-level analysis, Open-source software

ACM Reference Format:
Aoyu Gong, Arman Maghsoudnia, Raphael Cannatà, Eduard Vlad, Néstor Lomba Lomba, Dan Mihai Dumitriu, Haitham Hassanieh. 2025. Towards URLLC with Open-Source 5G Software. In *Workshop on Open Research Infrastructures and Toolkits for 6G (OpenRIT 6G IV), September 8–11, 2025, Coimbra, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3750718.3750743

1 Introduction

5G is the first cellular generation to introduce Ultra-Reliable Low-Latency Communication (URLLC), unlocking new possibilities for mission-critical applications like autonomous driving, industrial automation, and virtual and augmented reality. Introduced in 3GPP Release 15 [1], URLLC specifies stringent requirements, including one-way latency as low as 0.5 ms and reliability of 99.999 %. Yet, despite years of commercial deployments, reaching the target latency alone remains elusive [12], especially under real-world constraints.

Although extensive research has explored approaches to reduce latency, including using higher subcarrier spacing, shortened slot durations, and flexible frame structures [6, 14, 16, 17, 19, 24], most of them are theoretical or simulation-based. Importantly, they fail to consider practical system-level bottlenecks. Even in scheduling research, where many scheduling algorithms have been developed to manage latency across multiple users [3, 11, 22, 26], the difficulty of achieving low latency for a single user is often underestimated.

Progress in this area has historically been limited by the opaque nature of commercial cellular systems, which restricts visibility and control over system behaviors. Recent open-source Radio Access Network (RAN) projects, such as srsRAN [21] and OAI [13], provide researchers with full-stack programmability and the ability to experiment on real hardware using software-defined radios (SDRs). These platforms have opened the door to system-level experimentation that is essential for understanding and addressing practical system bottlenecks, a need that becomes even more pressing as the community begins shaping the 6G vision.

In this paper, we analyze and optimize latency using open 5G RAN software, aiming at uncovering system-level factors that contribute to latency and advancing toward URLLC targets. The contributions of this work are threefold. First, we systematically characterize latency sources arising not only from the 5G specification but also from implementation-level factors across key components of the RAN, highlighting their complex interactions. Second, we present a stage-by-stage latency breakdown from a real-world experimental setup, revealing the complete processing pipelines for both downlink (DL) and uplink (UL) scheduling. Third, we propose and implement a series of improvements to the open 5G RAN software that yield substantial reductions in one-way latency: 39.28 % in the DL and 55.38 % in the UL.

Our work shows how non-obvious implementation factors can lead to substantial latency, even in standards-compliant 5G systems. By uncovering these effects, we offer a practical blueprint for pushing open 5G RAN software to meet URLLC targets under real-world constraints. These insights can not only inform the improved design of current 5G systems but also lay the foundation for low-latency architectures in future 6G RANs. In addition, our work highlights the value of open-source, programmable RAN platforms in enabling the community to explore, validate, and refine low-latency designs.

2 Background

Radio Access Network (RAN). 5G networks provide wireless connectivity to devices known as User Equipment (UE), which not only include mobile phones and tablets, but also cars, drones, robots, and more. It comprises two main subsystems: the RAN and the Mobile Core. The RAN manages radio resources and consists of a distributed set of base stations known as gNBs. The Mobile Core authenticates devices, provides and ensures Internet connectivity, and tracks user mobility and subscriber usage.

The primary function of the RAN is to transfer packets between the Mobile Core and the UE. The stages of the RAN's packet processing pipeline are shown in Fig. 1, progressing from left to right

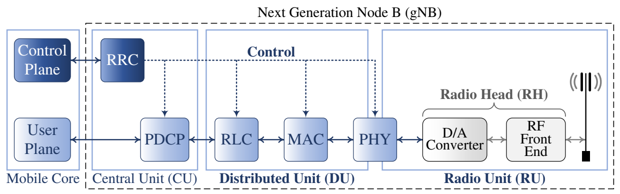


Figure 1: The RAN's packet processing pipeline.

as packets are transferred to the UE [18]. The key stages include the Radio Resource Control (RRC) layer, the Packet Data Convergence Protocol (PDCP) layer, the Radio Link Control (RLC) layer, the Media Access Control (MAC) layer, the upper and lower Physical (PHY) layer, and the Radio Head (RH). The functionality can be split across centralized and distributed locations. As shown in Fig. 1, the Radio Unit (RU) operates the lower PHY layer and the RH. The Distributed Unit (DU) and Centralized Unit (CU) act as the computational parts of the gNB. Specifically, the DU handles the RLC, MAC, and upper PHY layers, and is located at or near the RU. The CU runs the RRC and PDCP layers, and can be located near the Mobile Core.

5G frame structure. 5G introduced the support for multiple numerologies ($\mu \in \{0, 1, \ldots, 6\}$), corresponding to different OFDM subcarrier spacings (15 to 960 kHz). Each radio frame has a duration of 10 ms, which is divided into 10 subframes of 1 ms each. Each subframe consists of 2^{μ} slots, with each slot having a duration of $2^{-\mu}$ ms. Moreover, each slot contains 14 OFDM symbols, which can either be DL or UL. A slot is designated as DL or UL if all its symbols are of the same type; otherwise, it is considered a mixed slot. 5G supports two duplex modes: Frequency Division Duplex (FDD) and Time Division Duplex (TDD). The FDD mode allocates separate frequency bands for DL and UL transmissions. Each slot operates concurrently as DL in one frequency band and UL in another. In contrast, the TDD mode allocates the same band for both DL and UL transmissions, with each slot explicitly designated as DL, UL, or mixed. Sequencing DL and UL slots in a specific order enables flexible scheduling—this is known as the TDD pattern.

Open-source 5G. In recent years, significant open-source efforts have emerged to develop commercial-grade software-defined RAN solutions, leading to standards-compliant cellular networks such as srsRAN and OAI. In this paper, we mainly focus on srsRAN, as it is highly customizable, thoroughly documented, well-supported, and widely utilized in state-of-the-art research [7, 9, 23].

3 Latency Analysis

The 5G frame structure is a slot-based design. Before diving into the analysis, a natural question to ask is whether various parts of the RU and DU run on the same slot. The answer is no. As shown in Fig. 2, at any given time point, there is an H-slot offset between the RH and the RU-DL processor in the lower PHY layer, and an M-slot offset between the RU-DL and DU processors. These offsets arise as higher layers must process and deliver DL data in advance to ensure the uninterrupted operation of lower layers in the system.

3.1 Latency in Radio Units

The RU runs the lower PHY layer and the RH, whose functionalities are illustrated in Fig. 3. For DL transmission, the lower PHY layer

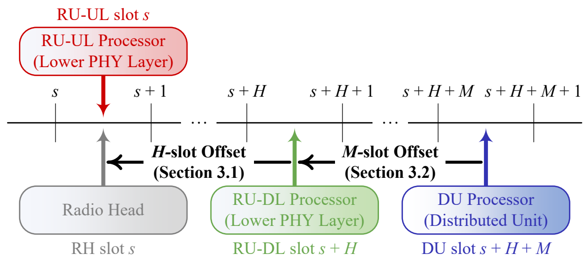

RU-UL slot s
RU-UL Processor (Lower PHY Layer)
s
$s+1$
$s+H$
$s+H+1$
$s+H+M$
$s+H+M+1$
...
H-slot Offset (Section 3.1)
M-slot Offset (Section 3.2)
Radio Head
RU-DL Processor (Lower PHY Layer)
DU Processor (Distributed Unit)
RH slot s
RU-DL slot $s+H$
DU slot $s+H+M$

Figure 2: The slot offsets among the RU and the DU.

transforms frequency-domain symbols on resource grids to time-domain samples and sends them to the RH. The RH then changes the samples to analog signals through DACs and upconverts them to Radio Frequency (RF) signals for transmission via the antenna. The reverse process occurs for UL reception. The lower PHY layer runs within general-purpose computers,[1] and the RH can be effectively deployed using SDR devices. The lower PHY layer and the RH communicate via software interfaces, which enable the wired transmission of time-domain samples between computers and SDR devices via USB, PCIe, or Ethernet.

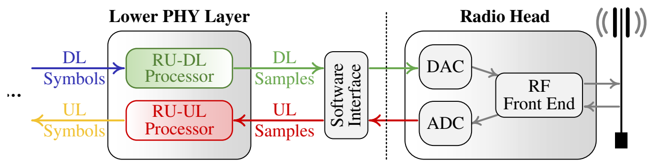


Figure 3: The lower PHY layer and the RH.

The RH, responsible for converting between digital samples and RF signals, operates on a fixed hardware clock. This clock advances at a constant rate and produces a single, continuous stream of sample indexes. Both transmission (DL) and reception (UL) are aligned to this unified sample timeline. This hardware-driven timing forms the basis from which slot boundaries in the 5G frame structure are derived. To interact with the RH, the RU consists of two processors in the lower PHY layer: one for UL (RU-UL) and one for DL (RU-DL). The two processors handle different processing tasks, each with its internal schedule. To coordinate timing across these components, a virtual slot index is computed for each based on its current sample index, the system's sampling rate, and the slot duration. Specifically, the slot indexes for the RH, the RU-UL processor, and the RU-DL processor are obtained by dividing their respective sample indexes by the number of samples per slot. We denote these indexes as the *RH slot*, *RU-UL slot*, and *RU-DL slot*, respectively. It is important to note that upper layers in the 5G stack do not inherently operate in terms of "slots." Instead, this concept must be inferred based on the RH's sample-level timing. Introducing a slot-based abstraction across various components enables a unified view of system behaviors and facilitates precise reasoning about timing alignment and latency-critical coordination. Based on these virtual slot indexes, the working procedure of the RU is illustrated in Fig. 4.

- **UL reception.** ① When RH slot s begins, the RH receives the UL signal over the air and immediately converts it into UL samples

[1]This applies to both the srsRAN and OAI software. However, the lower PHY layer can also be integrated into the RH.

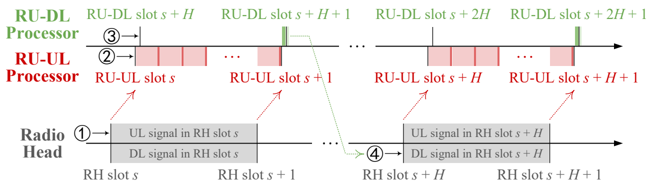


Figure 4: The working procedure of the RU.

using its ADC. These samples are then forwarded to the software interface. ② Shortly after this, RU-UL slot s begins. During this slot, the RU-UL processor retrieves these UL samples from the software interface and transforms them into UL symbols.

- **DL transmission.** Following the beginning of RU-UL slot s, RU-DL slot $s+H$ also begins. While the RU-UL and RU-DL slots are aligned in time, their indexes differ by a fixed offset of H. ③ In RU-DL slot $s+H$, the RU-DL processor first waits for the RU-UL processor to finish processing in RU-UL slot s.[2] After this, the RU-DL processor prepares DL samples and forwards them to the software interface. The software interface then passes these DL samples to the RH. ④ When RH slot $s+H$ begins, the RH reads the stored DL samples, converts them into DL signals using its DAC, and transmits the signal over the air.

It is important to distinguish between RU-UL and RU-DL, and the UL and DL slots defined in the 5G frame structure. RU-UL and RU-DL refer to two separate processors in the RU's lower PHY layer, indicating the directions of processing, but do not imply that each processor is only active during UL or DL slots. In FDD mode, both processors operate continuously, processing their respective data in every slot. In TDD mode, even in slots where no data is transmitted or received, both processors stay active to preserve synchronization and slot tracking. For example, the RU-DL processor still forwards zero-valued samples during UL slots to prevent underflow at the RH, and the RU-UL processor continues processing noise-only samples during DL slots to ensure alignment in the pipeline.

As introduced earlier, there is a time offset between the RH and the RU-DL processor. This offset accounts for both UL and DL radio preparation time. Here, the *UL radio preparation time* (red arrows in Fig. 4) refers to the time needed to downconvert RF signals to UL samples at the RH and forward them to the software interface. In contrast, the *DL radio preparation time* (green arrows in Fig. 4) refers to the time needed to forward DL samples from the software interface to the RH and upconvert them to RF signals. If these DL samples are not forwarded to the RH fast enough, the RH will lack samples to convert at the designated time, resulting in underflow. While the H-slot offset improves pipeline stability, it contributes directly to DL latency, since DL samples need to wait in the RH's memory before being upconverted to RF signals.

3.2 Latency in Distributed Units

The DU consists of the RLC, MAC, and upper PHY layers, operating within general-purpose computers. The functionalities of the three layers are shown in Fig. 5. The RLC layer handles the segmentation and reassembly of data. The MAC layer addresses real-time

[2]This waiting mechanism is implemented to slow down the RU-DL processor, i.e., to prevent it from running too far ahead. It ensures the offset between RU-DL and RU-UL slot indexes remains fixed at H slots.

scheduling regarding physical resource allocation. Moreover, for DL transmission, the MAC layer reads segments from the RLC layer, multiplexes them into transport blocks (TBs), and forwards the TBs to the upper PHY layer, which converts them into frequency-domain symbols on resource grids. The reverse process occurs for UL reception.

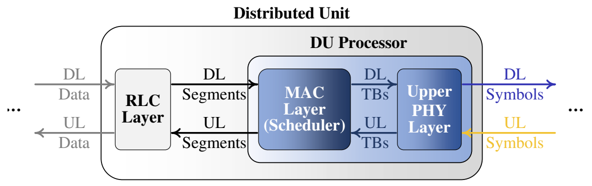


Figure 5: The RLC, MAC, and upper PHY layers.

To interact with the RU-DL and RU-UL processors in the lower PHY layer, the DU employs an event-driven processor whose execution is triggered by external events originating from the lower PHY layer. For ease of analysis, we denote the slot index associated with the DU processor as the *DU slot*. The working procedure of the DU is illustrated in Fig. 6.

- **DL transmission.** ① Once the RU-DL processor in the lower PHY layer finishes preparing DL samples in RU-DL slot $s+H-1$, it notifies the DU processor of the beginning of DU slot $s+H+M$. The RU-DL and DU slots are aligned in time, their indexes differ by a fixed offset of M. ② Upon receiving the notification, the DU processor first decides on resource allocation for DL transmission and/or UL reception. If the current DU slot is a DL slot, it then reads DL segments from the RLC layer and converts them into DL symbols. ③ These symbols wait in the computer's memory and will be read by the RU-DL processor after M slots (i.e., at the end of RU-DL slot $s+H+M$).
- **UL reception.** ④ After the RU-UL processor finishes preparing UL symbols in RU-UL slot s, it immediately forwards them to the DU processor. ⑤ Using the resource allocation results decided $(H+M)$ slots ago (i.e., in DU slot s), the DU processor converts the UL symbols into UL segments.

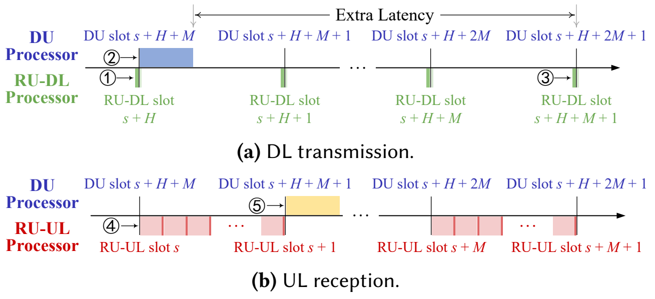


Figure 6: The working procedure of the DU.

As mentioned earlier, there is a time offset between the RU-DL and DU processors. This offset is leveraged by the DU processor to prepare resource allocation results and DL frequency-domain symbols in advance. It provides a margin for handling unexpected slowdowns in the DU processor, thus ensuring the uninterrupted

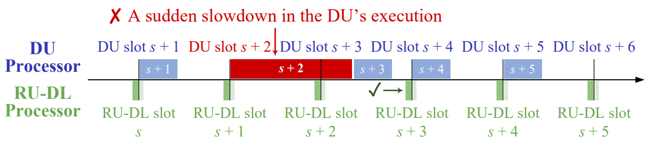


Figure 7: An example for the case of $M = 1$.

operation of the RU-DL processor. An example of $M = 1$ is shown in Fig. 7. In DU slot $s + 2$, a sudden slowdown in the DU's execution occurs. The DU's execution time in this DU slot is much longer than usual. As the DU processor prepares the DL symbols in advance, the RU-DL processor can still correctly access these symbols at the end of RU-DL slot $s + 2$. Moreover, the time offset quickly restores once the DU resumes normal execution. Note that this M-slot offset also contributes directly to DL latency, as DL symbols need to wait in the computer's memory before being transformed into DL samples.

3.3 Latency Breakdown

To present a comprehensive latency breakdown, it is essential to also consider the latency introduced by the 5G specifications. To achieve this, we first introduce Scheduling Requests (SRs). In UL scheduling, when a UE has UL data to send, it first transmits an SR to the gNB. The gNB then schedules and sends a grant to the UE. The UE transmits the UL data to the gNB after receiving this grant. Compared to DL scheduling, UL scheduling is more complex due to this procedure. For instance, in the TDD mode, a UE *cannot* send UL data and control information during DL slots. As a result, if the UE receives a grant and misses the last UL slot in a TDD period, it will need to wait at least for the first mixed slot in the next TDD period. As UL scheduling includes the DL scheduling of grants, we mainly focus on the latency breakdown of UL scheduling. The parameters M, H, and K_2 are set to the minimum values allowed by the open-source software, where K_2 is the number of slots assigned to a UE for UL data preparation after receiving a grant.

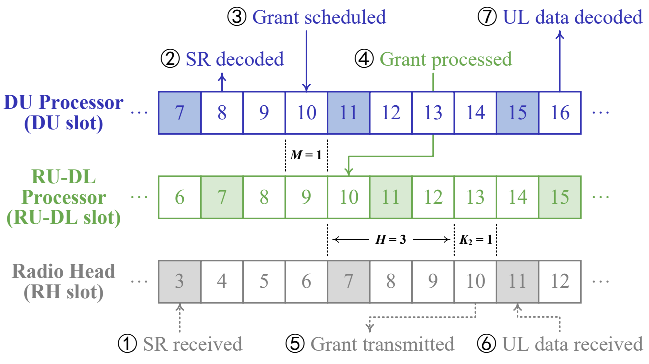


Figure 8: The latency breakdown of UL scheduling in TDD mode **DDDU** with Numerology 1, where $H = 3$, $M = 1$, and $K_2 = 1$. In each radio frame, slots 3, 7, 11, 15 are UL slots, while the remaining slots are DL slots.

The latency breakdown of UL scheduling in TDD mode **DDDU**[3] with Numerology 1 is shown in Fig. 8. The working procedure is explained as follows. ① At RH slot 3, the RH receives an SR from a UE. ② After one slot (i.e., in DU slot 8), the DU processor decodes the SR. ③ After two slots (i.e., in DU slot 10), the DU processor schedules a grant for the UE based on its resource demand. ④ After $M = 1$ slot (i.e., in RU-DL slot 10), the RU-DL processor processes the grant. ⑤ After $H = 3$ slots (i.e., in RH slot 10), the RH transmits the grant to the UE. ⑥ After $K_2 = 1$ slots (i.e., in RH slot 11), the RH receives the UE's data. ⑦ After one slot (i.e., in DU slot 16), the DU processor decodes the data. Based on this, DL and UL one-way latency are lower-bounded by:

$$③\rightarrow⑤\text{: DL latency} > (H + M + 1) \times 0.5 \text{ ms } = 2.5 \text{ ms}, \quad (1)$$

$$①\rightarrow⑥\text{: UL latency} > 9 \times 0.5 \text{ ms } = 4.5 \text{ ms}. \quad (2)$$

It is obvious that the UE misses the UL slot in the first TDD period (i.e., RH slot 7) due to the existence of the M- and H-slot offsets, leading to four more slots in the UL latency.

Remark: In srsRAN, the H-slot offset is hardcoded in the lower PHY layer with a fixed value of $H = 3$. Although a large UL radio preparation time improves robustness at the RU, it leads to a significant latency bottleneck. The M-slot offset is configurable via the gNB configuration file through the parameter `max_proc_delay`, with a minimum value of $M = 1$. Our analysis also applies to OAI, which provides less flexibility in control. Specifically, it uses a single parameter, `sl_ahead`, to configure the combined $H + M$ slot offset. Although implementations of the 5G stack may differ, we believe our analysis provides generalizable insights that can be extended to other implementations, particularly those with accessible source code, thus helping researchers better analyze latency behavior.

[3]The TDD pattern has a periodicity of four slots, with three DL slots (**DDD**) followed by one UL slot (**U**). The full cycle spans 2 ms, with each slot lasting 0.5 ms.

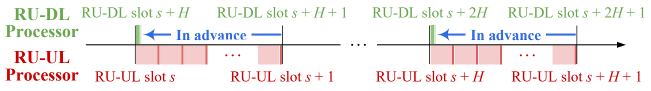


Figure 9: The improvement to the lower PHY layer.

4 Latency Improvement

In this section, we present improvements to the implementation of the DU, the RU, and the UL scheduling procedure.

4.1 Preparing Samples in Advance

As shown in Fig. 4–③, the RU-DL processor always prepares DL time-domain samples at the end of each RU-DL slot. This approach introduces extra latency if the DL symbols are ready at the beginning of a RU-DL slot or even earlier. To address this limitation, we decouple the preparation and transmission of DL samples from the boundaries of RU-DL slots. As shown in Fig. 9, without waiting for the RU-UL processor to finish, the RU-DL processor now prepares DL samples at the beginning of each RU-DL slot. This change allows the RU-DL processor to prepare DL samples in advance. Moreover, the slot indexes for the RU-UL and RU-DL processors are obtained based on the indexes of their respective time-domain samples. By allowing the maximum difference between the two sample indexes to be adjustable, we further make the parameter H configurable, with a minimum value of $H = 1$.

As illustrated in Fig. 10, we obtain the updated latency bounds:

$$③\rightarrow⑤\text{: DL latency} > (H + M + 1) \times 0.5\text{ ms} = 1.5\text{ ms}, \quad (3)$$

$$①\rightarrow⑥\text{: UL latency} > 9 \times 0.5\text{ ms} = 4.5\text{ ms}. \quad (4)$$

Compared to Inequalities (1) and (2), this reflects a 1 ms reduction in the DL latency, driven by decreasing the H-slot offset from 3 to 1. However, as the M-slot offset remains unchanged, the UE still misses RH slot 7, leaving the UL latency unaffected.

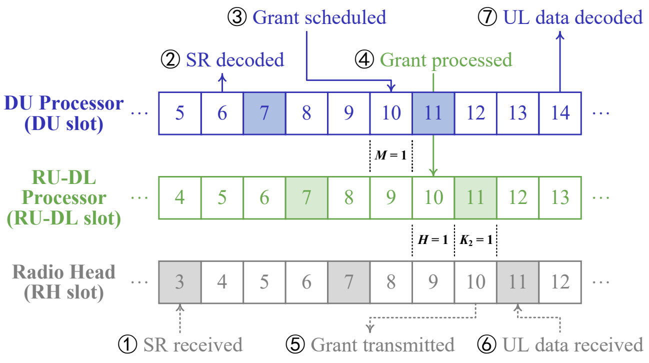


Figure 10: The updated latency breakdown of UL scheduling, where $H = 1$, $M = 1$, and $K_2 = 1$.

4.2 Event-Driven Symbol Transforming

As presented in Fig. 6a–②, the DU processor always prepares DL frequency-domain symbols M slots in advance. This early preparation introduces extra latency, as these symbols wait in the computer's memory and will be accessed by the RU-DL processor after M slots. To reduce this latency, a natural design goal is to have both processors run on the same slot, i.e., to enable $M = 0$. To achieve this, we implement a busy-waiting process in the RU-DL processor for DL slots,[4] changing its interaction with the DU processor. This improvement is outlined in Fig. 11. ① When DU slot s begins, the DU processor starts to prepare resource allocation results and DL frequency-domain symbols. ② In the meantime, the RU-DL processor enters the busy-waiting state, continuously checking whether the DL symbols are ready. ③ Once the condition is met, the RU-DL processor accesses the DL symbols and transforms them into DL time-domain samples.

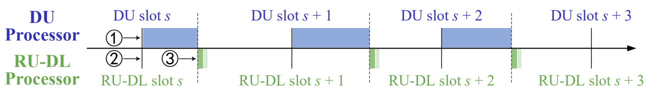


Figure 11: Improvement to the DU and RU interaction.

As illustrated in Fig. 12, incorporating the previous two improvements yields the following latency bounds:

$$③\rightarrow⑤\text{: DL latency} > (H + M + 1) \times 0.5\text{ ms} = 1\text{ ms}, \quad (5)$$

$$①\rightarrow⑥\text{: UL latency} > 5 \times 0.5\text{ ms} = 2.5\text{ ms}. \quad (6)$$

[4]Under the TDD mode, the busy-waiting process is skipped in UL slots since there are no DL frequency-domain symbols. So, the RU-DL processor prepares and transmits zero-valued samples immediately after the beginning of these slots.

Relative to Inequalities (3) and (4), this improvement further results in a 0.5 ms reduction in the DL latency, attributed to the elimination of the M-slot offset. Moreover, with both the M- and H-slot offsets reduced, the UE can catch up to RH slot 7, thus removing the second TDD period, i.e., shortening the UL latency by 2 ms.

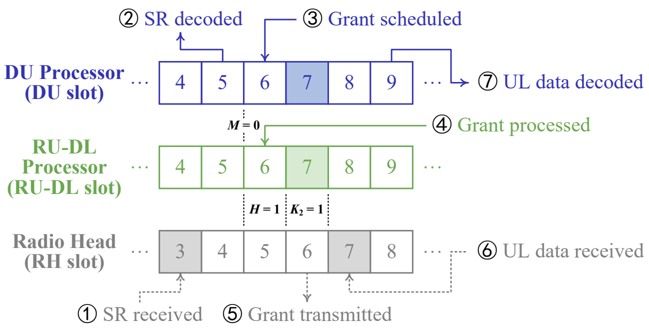


Figure 12: The updated latency breakdown of UL scheduling, where $H = 1$, $M = 0$, and $K_2 = 1$.

4.3 Scheduling-Request-Free Access

As shown in Fig. 8, the UL scheduling requires two slots of UL transmission (① and ⑥) and one slot of DL transmission (⑤) due to SRs. To streamline this procedure, we implement SR-free access. Specifically, once a gNB establishes a connection with a UE, it allocates a guaranteed amount of resources to the UE for UL transmission. A grant containing this allocation is sent to the UE before every UL slot. In consequence, when the UE has UL data to send, it can directly transmit the data to the gNB in the next available UL slot, without needing to first send an SR.

As illustrated in Fig. 13, with all three improvements in place, the latency bounds finally improve to:

$$①\rightarrow③\text{: DL latency} > (H + M + 1) \times 0.5\text{ ms} = 1\text{ ms}, \quad (7)$$

$$④\text{: UL latency} > 0.5\text{ ms}. \quad (8)$$

Compared to Inequalities (5) and (6), the UL latency sees an additional 2 ms reduction. This improvement stems from the fact that, under SR-free access, the UE can directly send UL data in a single slot without waiting for the next TDD period. Meanwhile, as SRs do not affect DL scheduling, the DL latency remains unchanged.

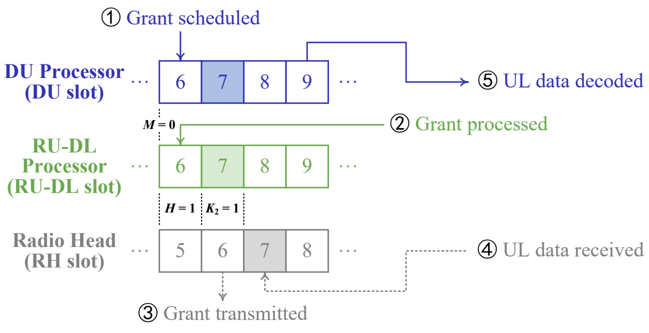
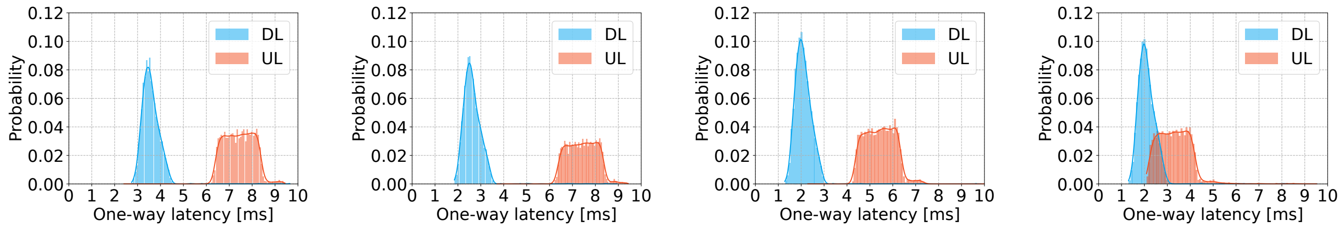


Figure 13: The updated latency breakdown of UL scheduling, where $H = 1$, $M = 0$, $K_2 = 1$, and SR-free access is adopted.



(a) Baseline ($H = 3$, $M = 1$). **(b)** **I1** ($H = 1$, $M = 1$). **(c)** **I1** + **I2** ($H = 1$, $M = 0$). **(d)** **I1** + **I2** + **I3** ($H = 1$, $M = 0$).

Figure 14: DL and UL one-way latency for TDD mode **DDDU** with Numerology 1 and $K_2 = 1$.

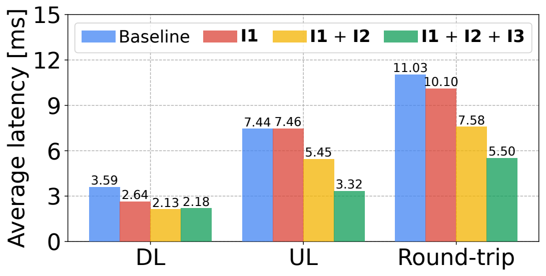


Figure 15: Comparison between the srsRAN baseline and cumulative implementations **I1**, **I1** + **I2**, and **I1** + **I2** + **I3**.

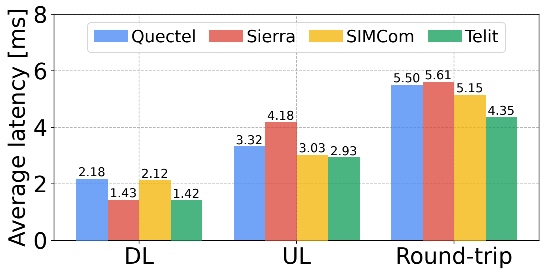


Figure 16: Comparison of four commercial 5G modules, including Quectel RM500Q-GL, Sierra EM9293, SIMCom SIM8200EA-M2, and Telit FN990A40, with the full implementation **I1 + I2 + I3**.

5 Evaluation

Testbed. We build a 5G testbed that contains an RAN based on srsRAN [21] and a Mobile Core based on Open5GS [10]. Both components are containerized using Kubernetes, operating on an Intel NUC 12 with a 12-core Intel i7-1260P Processor and 32 GB of DDR4 RAM. To deploy the RH, we use USRP B210, which offers an SDR platform with frequency coverage from 70 MHz to 6 GHz and USB 3.0 connectivity. We connect the Quectel RM500Q-GL 5G module to a second Intel NUC 12, enabling the device to run as a 5G UE.

Implementation details. Our improvements are implemented on top of srsRAN, which serves as the baseline. For simplicity, we denote the improvements in §4: **I1** for *preparing samples in advance*, **I2** for *event-driven symbol transforming*, and **I3** for *SR-free access*. The three improvements are made with approximately 150 lines of code in C++. To facilitate reproducibility and further research, we have made our implementation publicly available.[5]

Fig. 14 shows DL and UL one-way latency for TDD mode **DDDU** with Numerology 1 and $K_2 = 1$.[6] Compared to Fig. 14a, Fig. 14b reveals a 1 ms leftward shift in the DL latency distribution, enabled by reducing the buffering delay of DL time-domain samples from 1.5 ms ($H = 3$) to 0.5 ms ($H = 1$). A further 0.5 ms leftward shift in the DL latency distribution is seen in Fig. 14c, driven by reducing the buffering delay of DL frequency-domain symbols from 0.5 ms ($M = 1$) to 0 ms ($M = 0$). Versus Figs. 14a–b, Fig. 14c shows a 2 ms leftward shift in the UL latency distribution due to the combined reduction in both H and M, which allows the UE to catch up to the first UL slot (i.e., RH slot 7) following ① in Fig. 8. An extra 2 ms reduction in the UL latency distribution is seen in Fig. 14d, where the UE can transmit UL data without relying on SR signaling.

Fig. 15 compares the srsRAN baseline to cumulative implementations by reporting the average latency. The average DL latency decreases from 3.59 ms in the baseline to 2.64 ms with **I1**, 2.13 ms with **I1 + I2**, and 2.18 ms with the full implementation **I1 + I2 + I3**, reflecting a maximum reduction of approximately 1.5 ms. On the UL side, the corresponding latencies are 7.44 ms, 7.46 ms, 5.45 ms, and 3.32 ms, yielding reductions of up to approximately 4 ms. Overall, our full implementation achieves latency reductions of 39.28 % in DL, 55.38 % in UL, and 50.14 % in round-trip against the baseline, demonstrating substantial progress toward URLLC capabilities using open-source 5G software.

To further validate generalizability, we measure the average latency for four commercial 5G modules by individually replacing Quectel RM500Q-GL with Sierra EM9293, SIMCom SIM8200EA-M2, and Telit FN990A40 under the full implementation **I1 + I2 + I3**. As shown in Fig. 16, all devices exhibit average round-trip latencies below 6 ms. Among them, Telit reaches the lowest DL (1.42 ms), UL (2.93 ms), and round-trip latency (4.35 ms), closely followed by SIMCom and Quectel. These results confirm that our implementation can consistently deliver low-latency performance across diverse 5G modules, highlighting its compatibility with different chipsets and vendor-specific implementations. Additionally, the ability to reproduce sub-6 ms round-trip latency suggests the feasibility of applying our improvements to real-world deployments.

Fig. 17 presents the average round-trip latency when four commercial 5G phones are connected simultaneously to the gNB,[7] comparing the srsRAN baseline to the full implementation **I1 + I2 + I3**.

[5]Available at https://github.com/aygong/srsRAN_Project_Low_Latency

[6]We measure the one-way user-plane latency by capturing the time taken for ICMP packets to traverse from the UE's network interface to the User Plane Function (UPF)'s network interface in the UL direction, and vice versa in the DL direction. For each measurement, we send a total of 5000 ICMP packets.

[7]Due to the lack of access to low-level network interfaces and the difficulty of capturing user-plane traffic on commercial phones, we use `fping` from the UPF to the phone and directly report the round-trip latency as measured by `fping`.

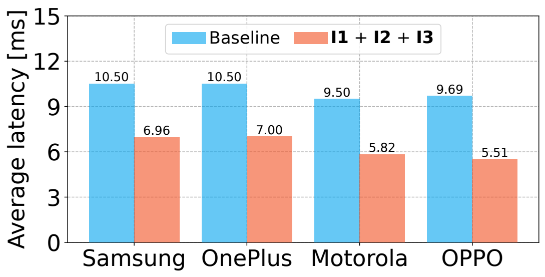


Figure 17: Comparison between the srsRAN baseline and the full implementation **I1** + **I2** + **I3** using four simultaneously connected commercial 5G phones: Samsung Galaxy A23, OnePlus Nord CE 3 Light, Motorola Moto G54, and OPPO A78.

The test devices include: Samsung Galaxy A23, OnePlus Nord CE 3 Lite, Motorola Moto G54, and OPPO A78. Despite the increased system load when scaling from a single UE to multiple UEs concurrently, the improved implementation consistently reduces latency for all devices, ranging from 33.3 % to 43.1 % relative to the baseline. These results demonstrate the robustness and scalability of our improvements in a realistic, multi-UE setup. Our experiment is conducted on a general-purpose SDR platform running at 20 MHz bandwidth with a 1×1 antenna configuration. Although this basic setup is intended to show multi-UE capability, we expect that more advanced hardware, featuring higher bandwidth and multiple antennas, will enable the system to scale to real-world demands.

6 Discussion and Future Work

System-Level Bottlenecks. Our analysis reveals several system-level bottlenecks. First, the radio preparation time imposes a hard bound on latency that cannot be bypassed without faster hardware interfaces (e.g., PCIe or Ethernet). Second, while 5G supports very short slot durations, these gains are limited if key operations still require a fixed duration of advance processing, underscoring the importance of aligning protocol timing with hardware constraints. Third, processing throughput becomes a bottleneck as shorter slots demand higher compute capacity within tighter time budgets, especially under high bandwidth. Finally, latency remains sensitive to execution variability, as unpredictable delays in general-purpose systems undermine tight timing coordination across layers. These bottlenecks suggest that further latency reduction will require not only software optimizations, but also tighter integration among protocol design, hardware capabilities, and execution environments.

TDD versus FDD. Our analysis and improvements apply to both TDD and FDD modes. However, our latency breakdown and evaluation primarily focus on the TDD mode. This choice is motivated by the greater analytical complexity introduced by TDD, where the 5G frame structure has a more pronounced impact on latency composition. Focusing on TDD is also well-aligned with the primary deployment scenarios for URLLC applications, which are often realized through private 5G networks. These include use cases such as industrial automation in factories, critical communications in hospitals, autonomous operations in ports, and safety systems in mines. Such private deployments typically rely on TDD-based spectrum allocations, as private 5G spectrum is most commonly provisioned in TDD bands rather than paired FDD bands. Additionally, TDD's inherent channel reciprocity enables advanced techniques such as beamforming and interference management, which are particularly valuable in dense, controlled environments [8].

Reliability. Our experiments are conducted under stable channel conditions, reflecting practical URLLC deployment environments. URLLC applications are usually deployed in highly controlled settings—such as factories, hospitals, or infrastructure sites—where channel conditions are engineered for stability, with predictable propagation, minimal interference, and limited mobility [2]. This low-mobility characteristic is critical, as high-speed movement introduces rapid channel fluctuations that significantly undermine URLLC reliability targets [20]. In contrast, stationary or quasi-static devices like industrial sensors and robotic systems provide more stable channels and justify the conditions used in our evaluation. Nevertheless, we acknowledge that real-world deployments may encounter less ideal conditions. In such cases, additional mechanisms, such as proactive packet replication [25], multi-connectivity [15], predictive link adaptation [4], and advanced error correction [5], can be used to meet reliability targets. Future work can integrate these techniques with our improvements to better support URLLC under more dynamic channel conditions.

Scalability and Spectrum Efficiency. SR-free access reduces UL latency by eliminating SRs but introduces a fundamental trade-off between latency and spectrum efficiency. Pre-allocating resources to all devices shifts the system from demand-based to reservation-based allocation, risking spectrum underutilization when resources go unused. The efficiency of this mechanism depends critically on the balance between the amount of reserved resources per UE and the number of active UEs. For small networks with frequent transmissions, the latency gains may justify the overhead. However, as UE counts grow or traffic becomes sporadic, spectrum inefficiency becomes more significant. A more scalable approach would involve predictive scheduling—dynamically allocating resources based on predicted transmission needs. This could leverage traffic pattern analysis, application-specific learning, or higher-layer signaling to forecast demand, which offers a middle ground between reactive scheduling and static reservation. Future work could explore hybrid approaches that adaptively switch between scheduling modes based on network conditions and application requirements.

7 Conclusion

This paper contributes to closing the gap between URLLC's theoretical latency targets and real-world 5G performance by analyzing and optimizing one-way latency based on open-source 5G software. Through an in-depth analysis of DL and UL scheduling, we reveal specification- and implementation-level factors and apply improvements that reduce one-way latency by 39.28 % (DL) and 55.38 % (UL). Our results indicate that non-obvious implementation factors can substantially influence latency, even in standards-compliant systems. As the 6G vision begins to take shape, our work underscores the importance of practical, hardware-backed experimentation in driving low-latency innovation. We hope our work paves the way for future efforts to meet—and go beyond—URLLC requirements in next-generation cellular networks.

References

[1] 3GPP. 2017. *NR; NR and NG-RAN Overall description; Stage-2.* Technical Report 3GPP TS 38.300 version 16.2.0 Release 16. 3GPP.

[2] 3GPP. 2019. *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Study on physical layer enhancements for NR ultra-reliable and low latency case (URLLC).* Technical Report 3GPP TR 38.824 version 16.0.0 Release 16. 3GPP.

[3] Arjun Anand, Gustavo de Veciana, and Sanjay Shakkottai. 2020. Joint Scheduling of URLLC and eMBB Traffic in 5G Wireless Networks. *IEEE/ACM Transactions on Networking* 28, 2 (2020), 477–490. https://doi.org/10.1109/TNET.2020.2968373

[4] Arjun Balasingam, Manikanta Kotaru, and Paramvir Bahl. 2024. Application-Level Service Assurance with 5G RAN Slicing. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24).* USENIX Association, Santa Clara, CA, 841–857. https://www.usenix.org/conference/nsdi24/presentation/balasingam

[5] Zeynep B Kaykac Egilmez, Luping Xiang, Robert G Maunder, and Lajos Hanzo. 2019. The Development, Operation and Performance of the 5G Polar Codes. *IEEE Communications Surveys & Tutorials* 22, 1 (2019), 96–122. https://doi.org/10.1109/COMST.2019.2960746

[6] Daquan Feng, Changyang She, Kai Ying, Lifeng Lai, Zhanwei Hou, Tony Q. S. Quek, Yonghui Li, and Branka Vucetic. 2019. Toward Ultrareliable Low-Latency Communications: Typical Scenarios, Possible Solutions, and Open Issues. *IEEE Vehicular Technology Magazine* 14, 2 (2019), 94–102. https://doi.org/10.1109/MVT.2019.2903657

[7] Minseok Kim, Namjo Ahn, and Song Min Kim. 2024. NR-Surface: NextG-Ready µW-Reconfigurable mmWave Metasurface. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24).* USENIX Association, Santa Clara, CA, 1641–1657. https://www.usenix.org/conference/nsdi24/presentation/kim

[8] Tobias Laas, Josef A Nossek, Samer Bazzi, and Wen Xu. 2020. On Reciprocity in Physically Consistent TDD Systems with Coupled Antennas. *IEEE Transactions on Wireless Communications* 19, 10 (2020), 6440–6453. https://doi.org/10.1109/TWC.2020.3003414

[9] Jon Larrea, Andrew E. Ferguson, and Mahesh K. Marina. 2023. CoreKube: An Efficient, Autoscaling and Resilient Mobile Core System. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking.* Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3570361.3592522

[10] Sukchan Lee. 2025. Open5GS. (2025). https://open5gs.org/.

[11] Jing Li and Xing Zhang. 2020. Deep Reinforcement Learning-Based Joint Scheduling of eMBB and URLLC in 5G Networks. *IEEE Wireless Communications Letters* 9, 9 (2020), 1543–1546. https://doi.org/10.1109/LWC.2020.2997036

[12] Arman Maghsoudnia, Eduard Vlad, Aoyu Gong, Dan Mihai Dumitriu, and Haitham Hassanieh. 2024. Ultra-Reliable Low-Latency in 5G: A Close Reality or a Distant Goal?. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks.* Association for Computing Machinery, New York, NY, USA, 111–120. https://doi.org/10.1145/3696348.3696862

[13] OpenAirInterface. 2025. OpenAirInterface: 5G Software Alliance for Democratising Wireless Innovation. (2025). https://openairinterface.org/.

[14] Stefan Parkvall, Erik Dahlman, Anders Furuskar, and Mattias Frenne. 2017. NR: The New 5G Radio Access Technology. *IEEE Communications Standards Magazine* 1, 4 (2017), 24–30. https://doi.org/10.1109/MCOMSTD.2017.1700042

[15] Rahul Arun Paropkari and Cory Beard. 2023. Multi-Connectivity-Based Adaptive Fractional Packet Duplication in Cellular Networks. *Signals* 4, 1 (2023), 251–273. https://doi.org/10.3390/signals4010014

[16] Klaus I. Pedersen, Gilberto Berardinelli, Frank Frederiksen, Preben Mogensen, and Agnieszka Szufarska. 2016. A Flexible 5G Frame Structure Design for Frequency-Division Duplex Cases. *IEEE Communications Magazine* 54, 3 (2016), 53–59. https://doi.org/10.1109/MCOM.2016.7432148

[17] Klaus I. Pedersen, Frank Frederiksen, Gilberto Berardinelli, and Preben E. Mogensen. 2016. The Coverage–Latency–Capacity Dilemma for TDD Wide Area Operation and Related 5G Solutions. In *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring).* IEEE, Piscataway, NJ, USA, 1–5. https://doi.org/10.1109/VTCSpring.2016.7504504

[18] Larry Peterson, Oguz Sunay, and Bruce Davie. 2023. *Private 5G: A Systems Approach.* Systems Approach LLC, Tucson, AZ, USA.

[19] Maria Raftopoulou and Remco Litjens. 2021. Optimisation of Numerology and Packet Scheduling in 5G Networks: To Slice or not to Slice?. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring).* IEEE, Piscataway, NJ, USA, 1–7. https://doi.org/10.1109/VTC2021-Spring51267.2021.9448814

[20] Yasantha Samarawickrama, Alvaro A. M. de Medeiros, and Victor Cionca. 2023. Mission-Level URLLC Under Variable Rician Fading Conditions. In *2023 International Conference on Computer, Information and Telecommunication Systems (CITS).* IEEE, Genoa, Italy, 01–06. https://doi.org/10.1109/CITS58301.2023.10188772

[21] Software Radio Systems. 2025. srsRAN: Open Source RAN. (2025). https://www.srsran.com/.

[22] Rana M. Sohaib, Oluwakayode Onireti, Yusuf Sambo, Rafiq Swash, Shuja Ansari, and Muhammad A. Imran. 2023. Intelligent Resource Management for eMBB and URLLC in 5G and Beyond Wireless Networks. *IEEE Access* 11 (2023), 65205–65221. https://doi.org/10.1109/ACCESS.2023.3288698

[23] Zhaowei Tan, Jinghao Zhao, Boyan Ding, and Songwu Lu. 2023. CellDAM: User-Space, Rootless Detection and Mitigation for 5G Data Plane. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23).* USENIX Association, Boston, MA, 1601–1619. https://www.usenix.org/conference/nsdi23/presentation/tan

[24] Thilina N. Weerasinghe, Vicente Casares-Giner, Indika A. M. Balapuwaduge, and Frank Y. Li. 2021. Priority-Enabled Grant-Free Access with Dynamic Slot Allocation for Heterogeneous mMTC Traffic in 5G NR Networks. *IEEE Transactions on Communications* 69, 5 (2021), 3192–3206. https://doi.org/10.1109/TCOMM.2021.3053990

[25] Jianzhe Xue, Kai Yu, Tianqi Zhang, Haibo Zhou, Lian Zhao, and Xuemin Shen. 2024. Cooperative Deep Reinforcement Learning Enabled Power Allocation for Packet Duplication URLLC in Multi-Connectivity Vehicular Networks. *IEEE Transactions on Mobile Computing* 23, 8 (2024), 8143–8157. https://doi.org/10.1109/TMC.2023.3347580

[26] Hao Yin, Lyutianyang Zhang, and Sumit Roy. 2021. Multiplexing URLLC Traffic Within eMBB Services in 5G NR: Fair Scheduling. *IEEE Transactions on Communications* 69, 2 (2021), 1080–1093. https://doi.org/10.1109/TCOMM.2020.3035582

Acronyms

CU	Central Unit
DL	Downlink
DU	Distributed Unit
FDD	Frequency Division Duplex
gNB	Next Generation Node B
I1	Preparing Samples in Advance
I2	Event-Driven Symbol Transforming
I3	SR-Free Access
MAC	Medium Access Control
NR	New Radio
OAI	OpenAirInterface
OFDM	Orthogonal Frequency Division Multiplexing
PDCP	Packet Data Convergence Protocol
PHY	Physical
RAN	Radio Access Network
RF	Radio Frequency
RH	Radio Head
RLC	Radio Link Control
RRC	Radio Resource Control
RU	Radio Unit
SDR	Software Defined Radio
SLA	Service-Level Agreement
SR	Scheduling Request
TB	Transport Block
TDD	Time Division Duplex
UE	User Equipment
UL	Uplink
URLLC	Ultra Reliable Low Latency Communications