Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Telecommunications Engineering

Quality of Service-aware scheduling for distributed unit in Open Radio Access Network

Master Thesis

Supervisor: prof. Ing. Zdenek Becvar, Ph.D.

Field of study: Communications Systems and Networks

Bc. Lin, Xiu-Wei

May 2024

CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Lin Xiu Wei**	Personal ID number:	**525661**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Telecommunications Engineering**		
Study program:	**Electronics and Communications**		
Specialisation:	**Communication Systems and Networks**		

II. Master's thesis details

Master's thesis title in English:

Quality of Service-aware scheduling for distributed unit in Open Radio Access Network

Master's thesis title in Czech:

Přidělování komunikačních prostředků s ohledem na kvalitu služby pro distribuovanou jednotku v otevřené rádiové přístupové síti

Guidelines:

Study principles of radio resource scheduling in the mobile networks with a focus on its implementation in a distributed unit in the Open Radio Access Network (O-RAN). Outline a solution that allows an enhancement of the scheduler in the distributed unit towards a multi-user scheduling per transmission time internal (TTI). Implement the solution for the multi-user scheduling in the O-RAN software community source code and verify correctness of the implementation. Furthermore, consider requirements of individual users on quality of service (QoS) and enhance the implemented multi-user scheduling so that QoS requirements of users are reflected. Demonstrate a correctness of the implementation and effectiveness of the implemented QoS-aware multi-user scheduler by experiments.

Bibliography / sources:

[1] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," IEEE Communications Surveys & Tutorials, vol. 25, no. 2 (Second quarter), 2023.
[2] Y. Zaki, T. Weerawardane, C. Gorg, and A. Timm-Giel, "Multi-QoS-Aware Fair Scheduling for LTE," 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring), Budapest, Hungary, 2011.
[3] O-RAN Software Community, "I release," 2023. [Online]. Available at: https://wiki.o-ransc.org/display/REL/I+Release
[4] 3GPP, "5G; NR; MAC protocol specification," 3GPP TS 38.321, version 17.2.0 release 17, 2022.

Name and workplace of master's thesis supervisor:

prof. Ing. Zdeněk Bečvář, Ph.D. Department of Telecommunications Engineering FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **19.02.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **15.02.2026**

prof. Ing. Zdeněk Bečvář, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

Acknowledgment

I am sincerely grateful to Professor Becvar for his invaluable guidance and insightful advice throughout the process of writing my thesis and papers. His mentorship has been instrumental in shaping my understanding and approach to academic writing. I am equally indebted to Professor Ray for challenging me to push beyond my limits, fostering my growth and resilience in the face of academic challenges. Their mentorship has not only enriched my academic journey but also imparted valuable lessons in critical thinking and problem-solving.

I extend my heartfelt appreciation to my friends whose unwavering support and encouragement have been a constant source of strength during challenging times. Their willingness to lend an ear and offer guidance over countless phone calls has been truly uplifting.

Special thanks are due to my seniors, Dennis Huang and Mick Lin, whose guidance and advice have been invaluable in terms of implementation of the thesis. Their insights and expertise have been instrumental in shaping the development of my work.

Lastly, I am deeply grateful to my parents for their unwavering support and sacrifices. Their belief in my abilities and unwavering financial support have enabled me to pursue my academic and professional aspirations. I am forever indebted to them and aspire to make them proud by achieving my goals and providing them with the retirement they deserve.

Declaration

I declare that the submitted thesis was developed individually and that I listed all used information sources in accordance with the Methodical Guideline on Ethical Principles for College Final Work Preparation. In Prague, 24. May 2024.

Abstract

The Open Radio Access Network (O-RAN) architecture offers flexibility and efficiency compared to traditional architecture of mobile networks. One crucial aspect of O-RAN is management of radio resources, which directly impacts the network performance and users' quality of service. This thesis focuses on radio resource management, specifically on scheduling, in O-RAN. The scheduling implemented in the current O-RAN Software Community can only handle one user within one transmission interval (TTI). This limit efficiency of radio resource usage. To improve this, in this thesis, we introduce bit rate control mechanism to develop and implement a new scheduler that handles multiple users simultaneously while also considering guaranteed flow bit rate and maximum flow bit rate of individual users to ensure users service quality. Besides, the implemented scheduler also considers retransmission of erroneous blocks and provides fairness to users. The enhanced scheduler is integrated into the O-RAN distributed unit source code and undergoes extensive testing to validate its performance and reliability. The experiments demonstrate an increase in throughput by 15.1% introduced by the implemented multi-user scheduler compared to traditional O-RAN single user scheduler.

Keywords: Open RAN, O-RAN

Abstrakt

Architektura Open Radio Access Network (O-RAN) nabízí flexibilitu a efektivitu ve srovnání s tradiční architekturou mobilních sítí. Jedním z klíčových aspektů architektury O-RAN je správa rádiových prostředků, která přímo ovlivňuje výkon sítě a kvalitu služeb pro uživatele. Tato práce se zaměřuje na správu rádiových prostředků, konkrétně na přidělování rádiových prostředků (scheduling), v O-RAN sítích. Přidělování rádiových prostředků implementované v existujícím softwaru pro sítě O-RAN může zpracovat pouze jednoho uživatele v rámci jednoho přenosového intervalu. V této diplomové práci je zaveden mechanismus pro kontrolu přenosové rychlosti pro vývoj a implementaci nového bloku pro přidělování rádiových prostředků, který obsluhuje více uživatelů současně, přičemž bere v úvahu také garantovanou přenosovou rychlost a maximální přenosovou rychlost uživateli, abychom zajistili požadovanou kvalitu služeb. Kromě toho, implementovaný blok pro přidělování rádiových prostředků bere v úvahu retranslace chybných bloků dat a spravedlivé rozdělení prostředků mezi uživatele. Řešení je integrováno do zdrojového kódu distribuované jednotky O-RAN za účelem jeho testování a ověření funkčnosti a spolehlivosti. Experimenty ukazují zvýšení propustnosti sítě o 15.1% díky implementovanému řešení oproti řešení pro jednoho uživatele integrovaném v síti O-RAN.

Klícová slova: Open RAN, O-RAN

Contents

List of Figures

List of Tables

1. Introduction

The Open Radio Access Network (O-RAN) is a new emerging concept of a fully open and intelligent radio access network developed by the O-RAN Alliance [1]. The O-RAN provides open interfaces enabling vendors and operators to customize the network according to their own needs. The intelligence is embodied by the RAN intelligent controller (RIC), which monitors and controls the RAN by utilizing closed-loop automation exploiting artificial intelligence (AI). The collaborative efforts of the O-RAN Alliance and the Linux Foundation have given rise to the O-RAN Software Community (OSC). This community focuses on the development of reference software for essential O-RAN components in line with O-RAN specifications.

The O-RAN separates a traditional RAN into three components: O-RAN radio unit (O-RU), O-RAN centralized unit (OCU), and O-RAN distributed unit (O-DU). Each component plays a crucial role in enhancing the flexibility and efficiency of the network. The O-RU is responsible for handling radio transmission and reception including conversion of digital signal to analog and vice versa. The O-CU acts as a centralized entity responsible for a higher-layer processing including functions, such as radio resource management and network control. The O-DU is responsible for facilitating the communication between the network infrastructure and the user equipment (UE) ensuring reliable and efficient data transmission and overseeing functionalities of the Radio Link Control layer (RLC), medium access control layer (MAC), and physical layer (PHY) in the 5G protocol stack [2]. Meanwhile, the Open RAN Software Community (OSC) [3][4] collaborates with organizations like the O-RAN Alliance and the Linux Foundation to foster the development of open-source solutions for radio access networks, ensuring accessibility to every developer. Embracing the architecture and specifications outlined by the O-RAN Alliance workgroups, OSC endeavors to create an open-source software ecosystem aimed at establishing an intelligent 5G Radio Access Network. Over time, OSC has released eight versions of its open-source code, with each version denoted alphabetically from the earliest Release A to the latest Release H (released in June 2023). The community's code repository, accessible at [5], serves as a hub for ongoing developments and contributions. In this thesis, our primary emphasis lies on Layer 2 of the protocol stack, specifically addressing the MAC layer and the RLC layer.

Within OSC, several ongoing projects are underway, ranging from the Near-Real-time RAN Intelligent Controller Platform to developments in radio access network. This thesis primarily focuses on the development of a UE per Transmission Time Interval (TTI) scheduler and the implementation of a QoS-aware scheduler. This QoS-aware scheduler takes into account critical parameters such as Guaranteed Flow Bit Rate (GFBR) and Maximum Flow Bit Rate (MFBR), aiming to optimize resource allocation while ensuring stringent quality of service requirements.

In addition to OSC, also other projects, bodies, and activities contribute to the open-source RAN landscape. One example is Open Air Interface (OAI) [6], maintained by the OAI Software Alliance (OSA), providing open-source software for RAN and other network components and functions for 4G and 5G. Similarly, srsRAN [7], developed by Software Radio System (SRS) [8], offers an open-source software collection for 4G and 5G radio implementations. These projects, primarily developed in C and C++ programming languages, enrich the open-source RAN ecosystem.

However, despite OSC's advantages, the OSC software implementation lacks crucial functionalities within the MAC layer, particularly in terms of scheduling capabilities. Key absent features include multi-UE scheduling per transmission time interval (TTI) and multi-queue scheduling based on Guaranteed Bit Rate (GBR) information [9]. Multi-UE scheduling per TTI allows for the simultaneous scheduling of multiple UEs within a single TTI, thereby enhancing resource utilization. Meanwhile, multi-queue scheduling ensures that GBR traffic meets individual Quality of Service (QoS) requirements while maintaining transmission reliability. The absence of these functionalities imposes limitations on resource efficiency and may adversely impact network performance. Single-UE scheduling per TTI restricts spectrum resource utilization, potentially leading to resource underutilization and even starvation scenarios where minimum bit rate guarantees are not met and maximum bit rate limits are not enforced based on QoS requirements. To address these limitations within the MAC layer implementation of OSC, this paper aims to enhance the ORAN scheduler in the OSC O-DU [10] by introducing critical functionalities currently not implemented. Our primary objective is to overcome the challenge posed by scheduling of multiple UEs per TTI, with the ultimate goal of improving overall Physical Resource Block (PRB) utilization. Through the implementation of multi-UE per TTI scheduling, we intend to enhance resource allocation efficiency. Additionally, we propose the introduction of QoS-aware scheduling, leveraging Guaranteed Flow Bit Rate (GFBR) and Maximum Flow Bit Rate (MFBR) mechanisms to ensure the satisfaction of GBR traffic flows while preventing traffic from exceeding its maximum flow bit rate.

The paper [11] discusses resource scheduling based on diverse quality-of-service (QoS) requirements while considering channel status. In [12], the authors introduce a concept for controlling the maximum bit rate. Reference [13] highlights the absence of a standardized data communication mechanism to disaggregate QoS flows into multiple routes and demonstrates how to incorporate retransmission into the scheduling mechanism while ensuring precise fairness and rate requirements. Paper [14] explores prioritizing retransmissions for different classes of traffic.

However, these works overlook considerations for practical implementation. For instance, both papers fail to account for parameters defined in specifications or the nuanced factors influencing real-world deployment. In our analysis, we adopt a practical implementation perspective, delving into these overlooked aspects to develop a comprehensive understanding of the problem. By doing so, we propose a solution that bridges the gap between theoretical concepts and practical application, ensuring the efficacy and viability of our approach.

The main contributions of the thesis include enhancing the MAC scheduler in O-DU to support multi-UE scheduling in a single TTI, thereby improving spectral efficiency. Moreover, our solution is uniquely distinguished in the open-source domain by the introduction of a leaky bucket mechanism that incorporates Guaranteed Bit Rate (GBR) information. This innovative approach prioritizes both spectral efficiency and ensures adherence to QoS requirements. Furthermore, the development of SCH to support multiple UEs per TTI is compliant with current OSC slice-enabled SCH, along with the implementation of a framework supporting multiple queues based on different QoS flows, including GFBR, MFBR, and retransmission, significantly contributes to the enhancement of the open-source codebase. Overall, our enhanced scheduler not only achieves higher overall throughput but also guarantees the Guaranteed Flow Bit Rate (GFBR) for GBR traffic, while ensuring fairness through Maximum Fair Bit Rate (MFBR) in accordance with system requirements.

The rest of the paper is organized as follows. Section 2 presents the system architecture considered in this paper. Section 3 discusses the proposed method. Section 4 presents the experimental result. Finally, the conclusion and future works will be presented in section 5.

2. System Architecture

In this section, we present an in-depth examination of the system architecture, beginning with an overview of the O-RAN architecture to provide a comprehensive understanding of its structure and principles. We then delve into the architecture of the O-DU, the focal point of our study within the open-source domain. Subsequently, we explore the functionalities of the Medium Access Control (MAC) layer, offering insights into the specific components and functionality that form the crux of our investigation. Finally, we provide background information on UE attachment and UE dedicated scheduling, laying the groundwork by elucidating fundamental concepts governing the interaction between the radio access network and user equipment (UE) side.

2.1 O-RAN Architecture

In this section, we begin by providing an overview of the general O-RAN architecture. Following this, we delve into the specifics of the O-RAN component known as O-DU, which serves as the host for the MAC scheduler —a central focus of our paper.

The O-RAN architecture, depicted in Figure 2.1-1, represents a transformative framework that reshapes the architecture and functionality of mobile networks. At its core, the near real-time RIC emerges as a centralized entity pivotal for network optimization, policy enforcement, and intelligent control functions. This component significantly enhances network performance and efficiency by enabling dynamic resource control and optimization with minimal latency. It facilitates critical functions, such as mobility management and service orchestration, ensuring seamless operations. Complementing the near real-time RIC is the non-real-time RIC, which handles longer-term optimization tasks while providing similar functionalities. Additionally, the Service Management and Orchestration Framework (SMO) oversees network management and automates the deployment of network virtualization.

The O-CU, O-DU, and O-RU represent the disaggregated and virtualized components of the radio access network [15], establishing a noteworthy benchmark for other open-source entities like the O-RAN Alliance [16] and the Small Cell Forum [17]. These elements introduce flexibility, scalability, and interoperability through standardized interfaces and open architectures. The Y1 consumer block symbolizes the end-user, emphasizing their pivotal role as the ultimate beneficiary of the O-RAN architecture's advancements. Furthermore, the integration of O-eNB (Open eNodeB) signifies an open-source base station implementation within cellular networks, fostering innovation and accessibility. Last, the O-Cloud serves as a crucial cloud-based infrastructure hosting RAN control applications, thereby enhancing the agility and adaptability of the O-RAN system.

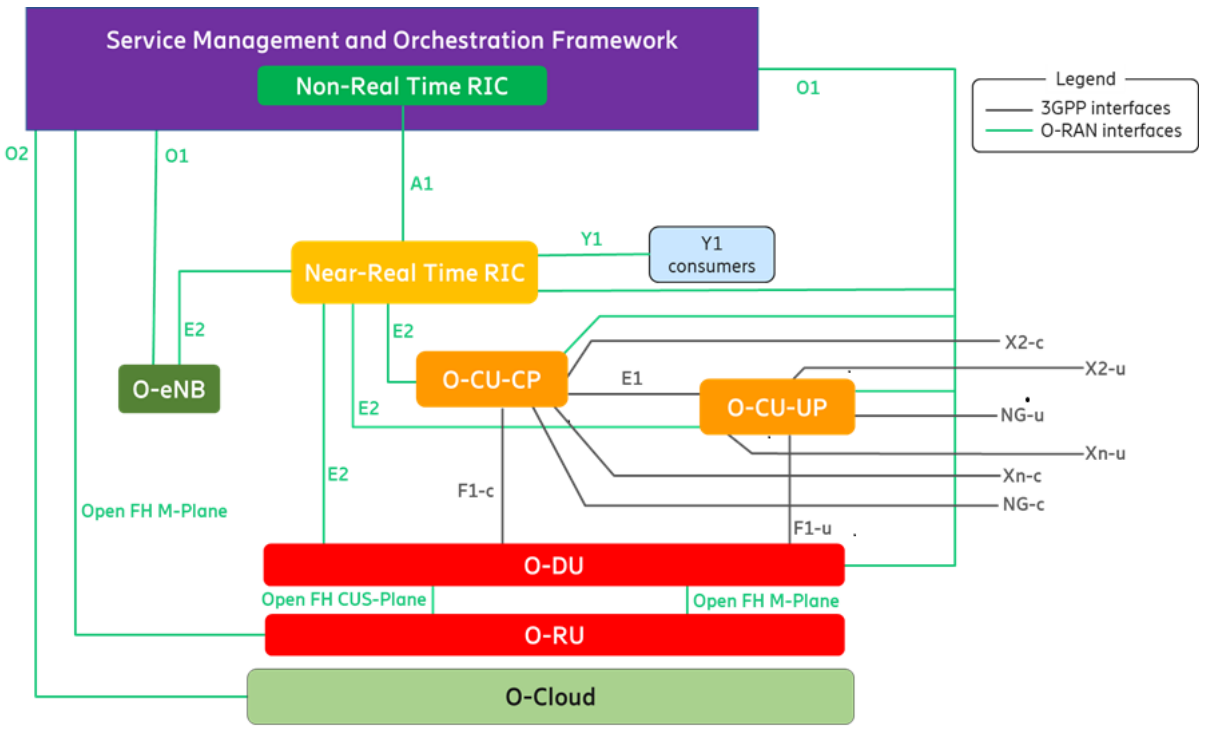


Figure 2.1-1: Logical Architecture of O-RAN [18]

Our main focus is O-DU High in this thesis. As shown in Figure 2.1-2, O-DU High interfaces with the following modules and each plays a crucial role in facilitating two primary functions: UE attachment with the radio access network and dedicated scheduling. Before initiating dedicated scheduling for user data transmission, the O-DU must complete connection setup with the O-CU via the F1 interface and exchange UE context and RRC messages. The management procedures are outlined as follows:

- The O-CU is responsible for communication with the O-DU High on the F1 Application Protocol (F1AP) interface. This interface facilitates both control and data message exchanges between the O-CU and O-DU High. The following key F1AP messages are implemented on the F1-C interface, with detailed specifications provided in reference [9]:
  - Interface Management
    - F1 Setup: Initiates the establishment of the F1 interface between the O-CU and O-DU High.
    - gNB-DU Configuration Update: Allows for updates to the configuration of the gNB-DU.
    - F1 Reset: Enables the reset of the F1 interface.
    - PAGING: Facilitates paging procedures for user equipment (UE) mobility management.
  - UE Context Management
    - UE Context Setup: Establishes and initializes UE context within the O-CU.
    - UE Context Modification: Allows for modifications to existing UE context parameters.
    - UE Context Release: Initiates the release of UE context resources.
  - RRC Message Transfer
    - Initial UL RRC Message Transfer: Transfers initial uplink (UL) Radio Resource Control (RRC) messages.
    - DL RRC Message Transfer: Transfers downlink (DL) RRC messages.
    - UL RRC Message Transfer: Transfers uplink (UL) RRC messages.
    - RRC Delivery Report: Provides acknowledgment and delivery status of RRC messages.

- O-DU Low serves as the interface between the O-DU High and the physical layer of the radio access network. Communication between O-DU High and O-DU Low occurs via the FAPI (Flexible Application Platform Interface) interface. Below are the supported FAPI messages, based on the FAPI interface files provided by Intel:
    - P5 messages - PHY mode control interface
        - PARAM.request/PARAM.response: Requests and responses for configuring PHY parameters.
        - CONFIG.request/CONFIG.response: Requests and responses for configuring PHY configuration parameters.
        - START.request: Initiates the start of PHY processing.
        - STOP.request: Requests the stop of PHY processing.
        - STOP.indication: Indicates the stop of PHY processing.
    - P7 messages - Main data path interface
        - DL_TTI.request: Requests downlink transmission time interval (TTI) processing.
        - UL_TTI.request: Requests uplink transmission time interval (TTI) processing.
        - SLOT.indication: Indicates the occurrence of a time slot.
        - UL_DCI.request: Requests uplink downlink control information (DCI) processing.
        - TX_Data.request: Requests transmission of data.
        - RX_Data.indication: Indicates reception of data.
        - CRC.indication: Indicates cyclic redundancy check (CRC) status.
        - UCI.indication: Indicates uplink control information (UCI) status.
        - RACH.indication: Indicates random access channel (RACH) status.

At the outset, O-DU High synchronizes its configuration with the physical layer through PARAM and CONFIG messages, establishing a common operational framework. The initiation of slot indication via START request from the physical layer triggers the commencement of crucial message exchanges between O-CU, O-DU High, and O-DU Low, as depicted in Figure 2.1-2. Within the F1AP message exchange, F1 setup and gNB-DU Configuration Update play pivotal roles in the cell-up procedure, facilitating the establishment of connectivity between O-RU and O-DU. Subsequently, messages pertaining to UE context management and RRC message transfer come into play, facilitating the establishment of UE context within the radio access network. The transmission of messages across the FAPI interface is transferred through P7 messages, where DL_TTI.request and UL_TTI.request dictate data transfer timings, while UL_DCI.request and UCI.indication transfer the data on control channel. Additionally, TX_Data.request and RX_Data.indication transfer the data on shared channel communication. Each of these message exchanges is integral to the setup of our environment, laying the foundation for the operation of radio access network.

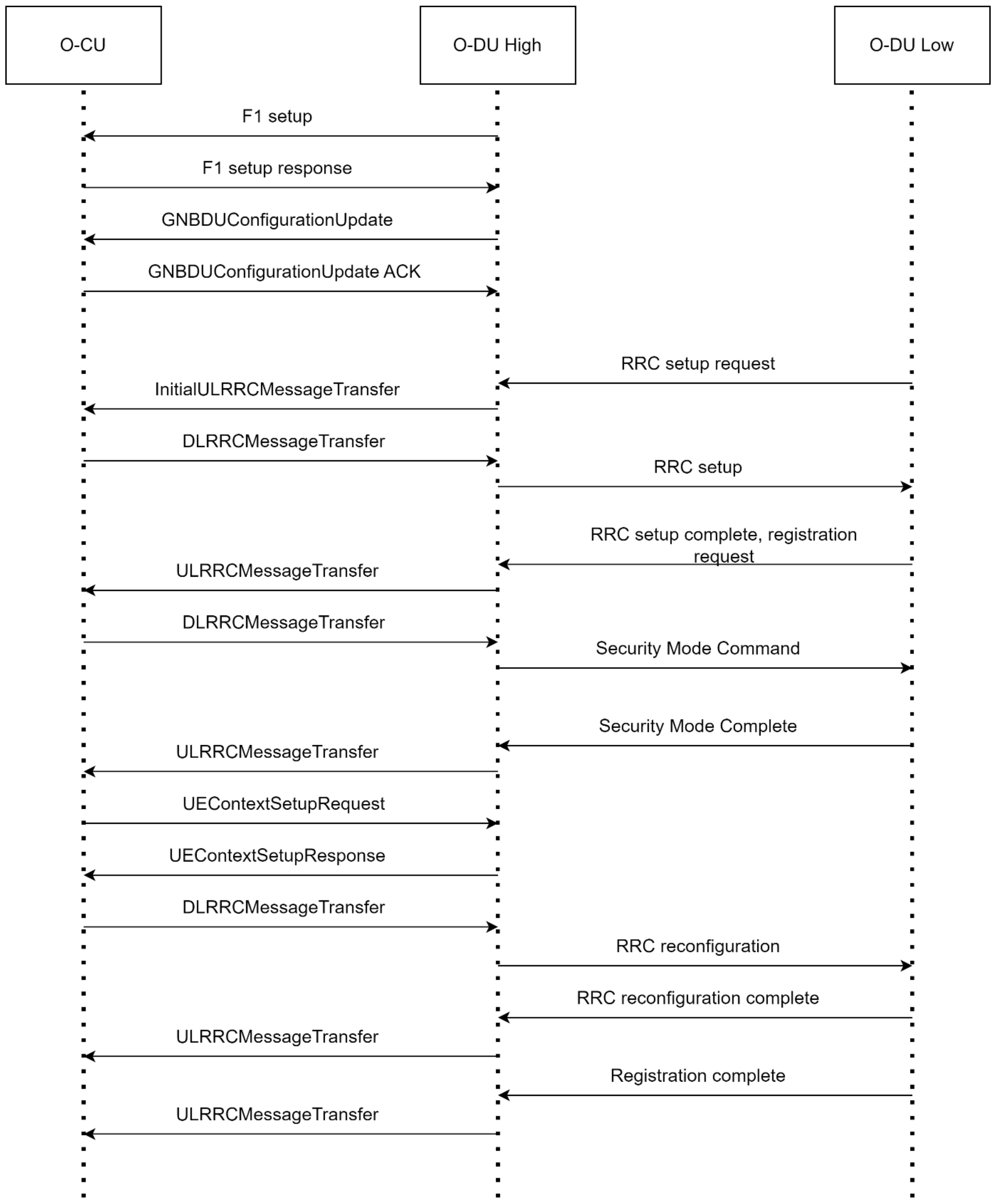


Figure 2.1-2: MSC of F1 setup, UE Context Management, RRC message transfer

2.2 O-DU High architecture

Broadly, the O-DU is horizontally divided into two distinct functional blocks: O-DU High, responsible for hosting RLC and MAC functionalities, and O-DU Low, managing specific aspects of the PHY layer functionalities [19]. The O-DU High operates through several threads, each dedicated to its unique network function. The architecture of the O-DU High is depicted in Figure 2.2-1 revealing numerous components, such as DU APP, RLC, MAC, Scheduler, or O1 Module [20], each serving specific functions across eight running threads.

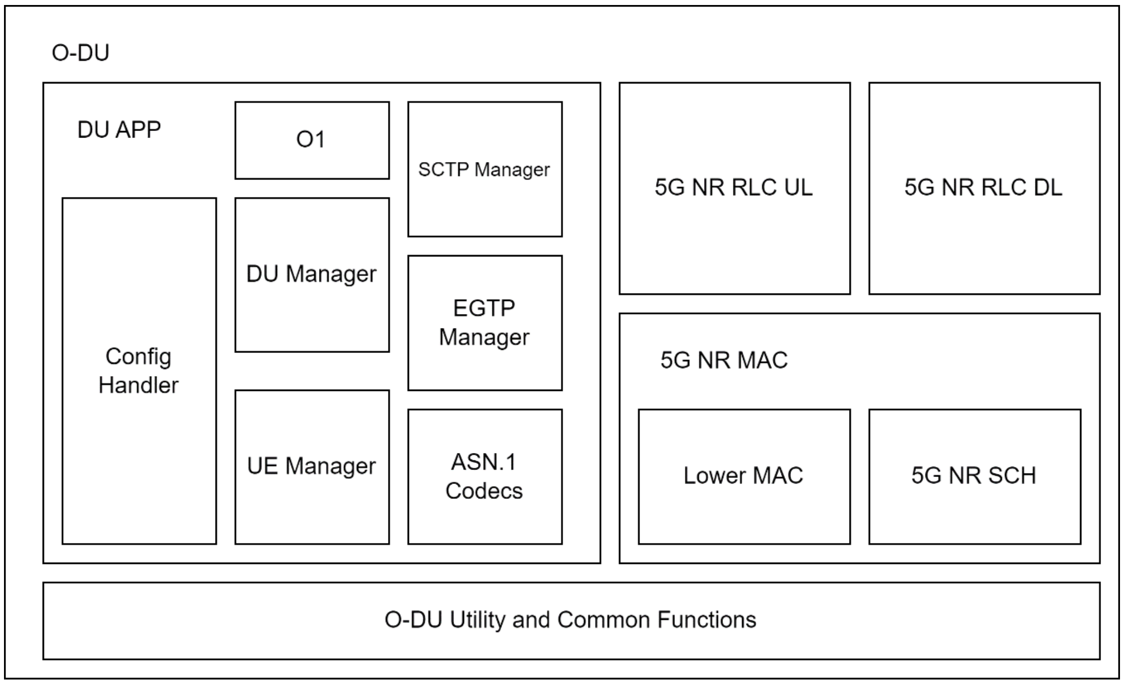


Figure 2.2-1: O-DU High architecture

A brief explanation of each module and submodule is provided below.

- DU APP: It serves as a central hub for communication and coordination within the Open RAN ecosystem, facilitating seamless interactions with various network entities. In its interface with the Operation, Administration, and Maintenance (OAM) system, DU APP leverages the O1 interface to manage configuration settings, monitor alarms, and conduct performance management tasks. Additionally, DU APP interfaces with the O-CU component, establishing connections over the F1 interface, which is underpinned by the Stream Control Transmission Protocol (SCTP). Within this interface, control messages traverse the F1-C channel, while data messages flow through the F1-U channel, enabling efficient exchange of RAN functionalities. Moreover, DU APP is responsible for storing the configurations of both O-DU High and UEs, as well as handling message forwarding within the system. It comprises several submodules, each serving specific functions within DU APP. Additionally, DU APP engages with the RAN Intelligent Controller (RIC) via the E2 interface, leveraging SCTP to facilitate seamless communication. Through these interfaces, DU APP plays a pivotal role in orchestrating network operations, optimizing resource allocation, and ensuring the reliability and performance of the Open RAN infrastructure.

    - Config Handler: Manages the configurations received on O1 interfaces and stores them within DU APP context.
    - DU (Cell) Manager: Handles all cell operations at the DU APP.
    - UE Manager: Manages UE contexts at the DU APP.
    - Stream Control Transmission Protocol (SCTP) Handler: Responsible for establishing SCTP connections with O-CU and RIC on the F1AP and E2AP interfaces, respectively.
    - Enhanced GPRS Tunneling Protocol (EGTP) Handler: Responsible for establishing EGTP connections with O-CU for data message exchange on the F1-U interface.
    - ASN.1 Codecs: Contain ASN.1 encode/decode functions used for System information, F1AP, and E2AP messages.
- O-DU Utility and Common Functions: The library for utility functions, typically utilized for message exchange purposes.
- 5G NR RLC: This component provides the functions of the NR RLC layer, ensuring the dependable and effective transmission of data across the radio interface. Its primary operations involve dividing and reconstructing data packets, crucial for maintaining data integrity and optimizing transmission efficiency. For a more comprehensive understanding of the NR RLC's operations, detailed explanations can be found in [21].
- 5G NR MAC: This component provides the functions of the NR MAC layer; it facilitates efficient management of radio resources in the 5G New Radio (NR) system. Its primary functions include scheduling and coordination of data transmissions, handling contention resolution, and managing access to the radio channel. Additionally, the NR MAC layer supports Quality of Service (QoS) management, prioritizing traffic based on application requirements and network conditions, detailed explanations can be found in [22].
- 5G NR SCH: A critical submodule of the MAC layer, the Scheduler, plays a pivotal role in orchestrating the allocation of radio resources to user equipment (UEs) across the network. Its primary function encompasses the strategic determination of transmission parameters tailored to each UE, encompassing factors like time-frequency resources and modulation schemes. By meticulously optimizing spectrum utilization, the Scheduler ensures streamlined and efficient data transmission throughout the network. It accomplishes this by dynamically assessing channel conditions, Quality of Service (QoS) criteria, and network congestion levels, enabling informed resource allocation decisions. Ultimately, the Scheduler's adept management maximizes network capacity and enhances the overall user experience.
- Lower MAC: It is also known as the MAC sublayer, is responsible for managing access to the physical transmission medium in a network by functional application platform interface (FAPI).

To implement multiple UEs per TTI scheduling, the primary software modules requiring modification are SCH, MAC, and lower MAC layers. For QoS-aware scheduling based on Guaranteed Flow Bit Rate (GFBR) and Maximum Flow Bit Rate (MFBR), the key modules in need of adjustment are SCH and RLC. Further elaboration on the implementation methodology is provided in subsequent sections of this paper.

2.3 Functionality of MAC layer

The MAC layer is responsible for crucial functions within the O-RAN architecture, encompassing tasks like channel mapping, MAC service data unit (SDU) management, error correction through hybrid automatic repeat request (HARQ), and prioritization of logical channels. In this paper, we focus on the intricate challenges and considerations surrounding scheduling within the O-RAN framework. Specifically, the MAC scheduler plays a pivotal role in managing various aspects, including User Equipment (UE) status reporting, allocation of physical layer resources, and rate control to uphold quality of service (QoS) standards [23]. Effective QoS provisioning hinges on the selection and implementation of appropriate scheduling algorithms tailored to radio resource allocation needs.

2.4 UE Attachment and UE dedicated scheduling

In the domain of 5G NR, scheduling stands as a fundamental process governing the allocation of limited spectrum resources for data transmission. At its core, the scheduler module orchestrates this allocation dynamically and efficiently. Figure 2.4-1 provides a visual representation of the sequential phases a User Equipment (UE) undergoes, transitioning from an unconnected state to establishing connectivity with a Next Generation Node B (gNB) and commencing data transmission. This progression unfolds across three distinct phases: the broadcast procedure, the random access procedure, and the UE transmission procedure.

During the broadcast procedure, synchronization between the UE and gNB is achieved through periodic broadcasts of system information by the gNB. These broadcasts include synchronization signals and essential system information such as the master information block (MIB) and system information block (SIB) [19]. Upon the conclusion of the broadcast procedure, the UE achieves Downlink (DL) synchronization after assimilating the necessary information from the gNB.

Subsequently, the random access procedure ensues to establish Uplink (UL) synchronization. This phase involves the exchange of four messages, known as MSG1 through MSG4, between the UE and gNB. Central to this process is the role of the scheduler in allocating spectrum resources for these messages. The scheduling activities conducted during both the broadcast and random access procedures fall under the common channel scheduling paradigm.

Upon successful attachment to the gNB, the UE gains the capability to initiate the transmission of user data to other UEs, marking the commencement of dedicated channel scheduling. This phase entails the allocation of spectrum resources based on Channel State Information (CSI) and Quality of Service (QoS) requirements. Downlink Control Information (DCI) serves as the conduit through which instructions regarding available spectrum resources and their utilization timing are conveyed to the UE. In response, the UE retrieves the corresponding data based on the information provided in the DCI.

When the need arises for Uplink (UL) data transmission, the UE dispatches a scheduling request (SR) and buffer status report (BSR) to the gNB, soliciting the allocation of resources for UL data transmission in the frequency domain. Subsequently, the gNB communicates the availability of spectrum resources and the corresponding timing to the UE through the DCI, empowering the UE to execute the transmission of UL data.

The overview of the procedure of OSC MAC SCH is shown in Figure 2.4-2. In this thesis, our initial focus is on ensuring the seamless functionality of User Equipment (UE) attachment following code modifications. Subsequently, we proceed to assess the efficacy of our scheduling mechanism through numerical analysis in dedicated scheduling scenarios. This sequential approach allows us to validate our implementation and demonstrate the effectiveness of the scheduling mechanism in practical deployment scenarios.

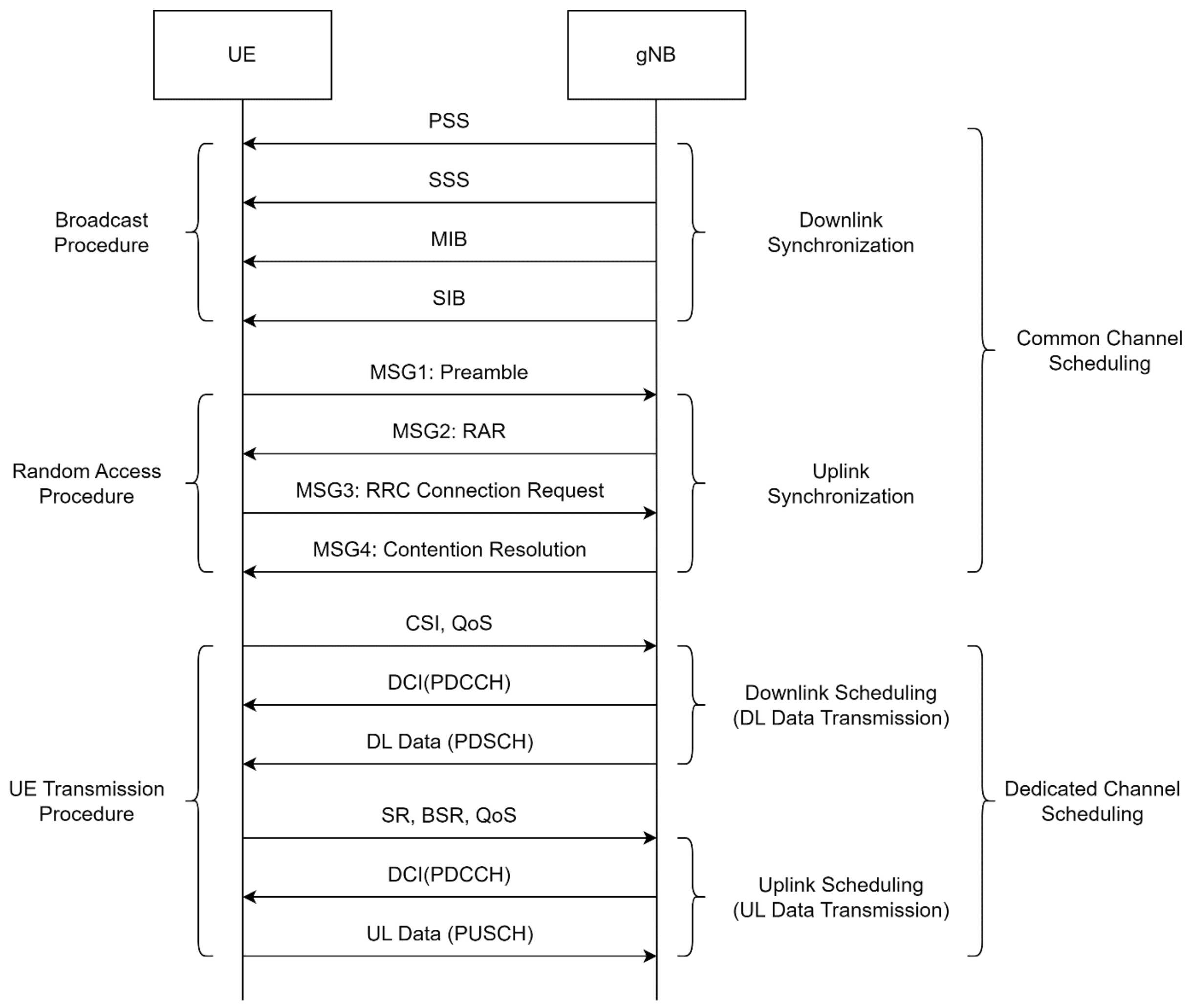

UE
gNB
PSS
SSS
MIB
SIB
Broadcast Procedure
Downlink Synchronization
MSG1: Preamble
MSG2: RAR
MSG3: RRC Connection Request
MSG4: Contention Resolution
Random Access Procedure
Uplink Synchronization
Common Channel Scheduling
CSI, QoS
DCI(PDCCH)
DL Data (PDSCH)
Downlink Scheduling (DL Data Transmission)
SR, BSR, QoS
DCI(PDCCH)
UL Data (PUSCH)
Uplink Scheduling (UL Data Transmission)
UE Transmission Procedure
Dedicated Channel Scheduling

Figure 2.4-1: The procedure of UE Attachment and UE dedicated scheduling in NR

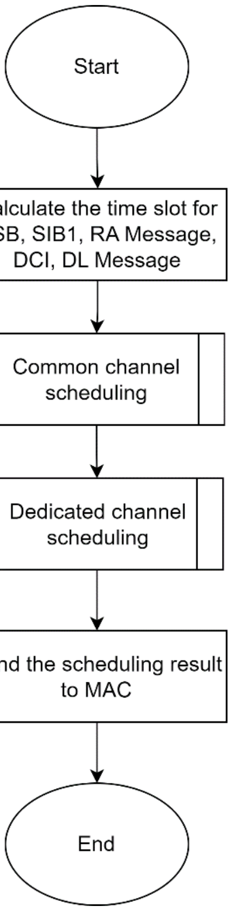

Start
Calculate the time slot for SSB, SIB1, RA Message, DCI, DL Message
Common channel scheduling
Dedicated channel scheduling
Send the scheduling result to MAC
End

Figure 2.4-2: The flow chart of OSC MAC scheduler

2.5 Scheduling of PRACH, PDCCH, PDSCH, PUCCH, and PUSCH

Scheduling in NR encompasses the allocation of resources for various channels, each serving a unique purpose. PRACH (Physical Random Access Channel) scheduling allows UEs to initiate communication with the base station by transmitting random access preambles. PDCCH (Physical Downlink Control Channel) scheduling involves allocating control information to UEs, carrying essential commands for resource assignments, and downlink data transmission. PDSCH (Physical Downlink Shared Channel) scheduling determines the allocation of downlink data resources, optimizing time-frequency blocks to meet QoS requirements. PUCCH (Physical Uplink Control Channel) scheduling allocates resources for UEs to send uplink control information such as acknowledgments, channel quality indicators, and scheduling requests. Finally, PUSCH (Physical Uplink Shared Channel) scheduling manages uplink data resources, assigning specific time-frequency blocks for UEs to transmit user data, ensuring a balanced load, and maintaining QoS for various applications.

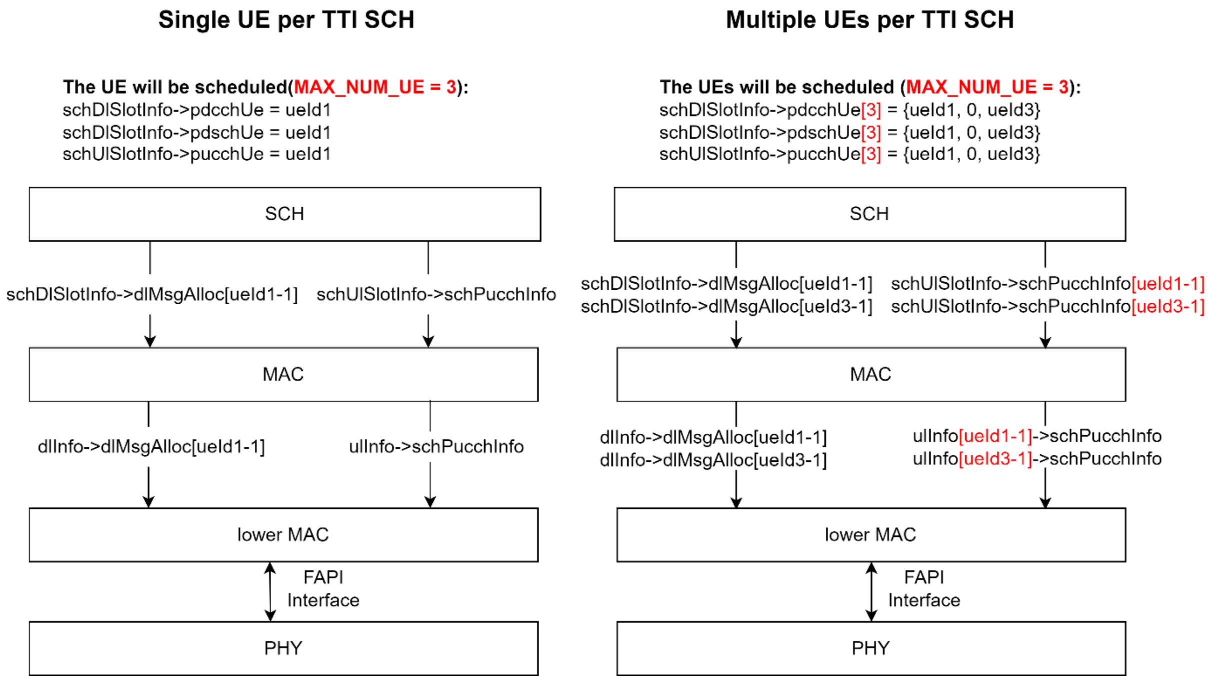
3 Developed Scheduler

In this section, we present our implementation approach to enhance scheduling capabilities within OSC O-DU High. Firstly, we detail the process of extending Single-UE (SU) per Transmission Time Interval (TTI) scheduling to support Multi-UE (MU) per TTI scheduling, thereby optimizing resource utilization and enhancing network efficiency. Following this, we delve into the implementation of a multiple queues scheduling framework based on 5G Quality indicator (5QI), incorporating features such as Guaranteed Flow Bit Rate (GFBR) and Maximum Flow Bit Rate (MFBR). By introducing this framework, we aim to ensure robust Quality of Service (QoS) provisioning while dynamically adapting to varying traffic demands. Lastly, we discuss the integration of retransmission queues into our scheduling architecture, accompanied by the application of our proposed algorithm. This comprehensive approach enables the handling of transport block retransmissions.

3.1 Scheduling of Multiple UE per TTI

To enable the scheduling of multiple UEs per TTI, our proposed solution unfolds in four stages, each validated through rigorous verification processes. We begin by modifying the O-DU's data structure to transition from single UE per TTI to supporting multiple UEs. The subsequent stages involve handling multiple Functional Application Platform Interface (FAPI) Protocol Data Units (PDUs), modifying the scheduler framework for efficient scheduling, and customizing the Centralized Unit (CU) simulator to tailor packet size and transmission period for individual traffic flows. Each stage is followed by thorough verification, ensuring the robustness of our solution.

The code modifications for supporting multiple UEs per TTI are illustrated in Figure 3.1-1. Here, *MAX_NUM_UE* represents the total number of UEs in the system. For example, *ueId1* is the identifier for UE1, *ueId2* is the identifier for UE2, and *ueId3* is the identifier for UE3. In the scenario of multiple UEs per TTI, both UE1 and UE3 are scheduled within a single TTI, demonstrating the enhanced scheduling capability.



Single UE per TTI SCH
The UE will be scheduled(MAX_NUM_UE = 3):
schDlSlotInfo->pdcchUe = ueId1
schDlSlotInfo->pdschUe = ueId1
schUlSlotInfo->pucchUe = ueId1
SCH
schDlSlotInfo->dlMsgAlloc[ueId1-1]
schUlSlotInfo->schPucchInfo
MAC
dlInfo->dlMsgAlloc[ueId1-1]
ulInfo->schPucchInfo
lower MAC
FAPI Interface
PHY
Multiple UEs per TTI SCH
The UEs will be scheduled (MAX_NUM_UE = 3):
schDlSlotInfo->pdcchUe[3] = {ueId1, 0, ueId3}
schDlSlotInfo->pdschUe[3] = {ueId1, 0, ueId3}
schUlSlotInfo->pucchUe[3] = {ueId1, 0, ueId3}
SCH
schDlSlotInfo->dlMsgAlloc[ueId1-1]
schDlSlotInfo->dlMsgAlloc[ueId3-1]
schUlSlotInfo->schPucchInfo[ueId1-1]
schUlSlotInfo->schPucchInfo[ueId3-1]
MAC
dlInfo->dlMsgAlloc[ueId1-1]
dlInfo->dlMsgAlloc[ueId3-1]
ulInfo[ueId1-1]->schPucchInfo
ulInfo[ueId3-1]->schPucchInfo
lower MAC
FAPI Interface
PHY

Figure 3.1-1: The illustration of code modification for MU per TTI SCH

To initiate the enhancement process, the foremost step involves the modification of data structures *schUlSlotInfo* and *macUlSlot*. This encompasses extending variables *schPucchInfo* and *ulInfo* within the data structure from a single element to multiple elements, facilitating the storage of information for multiple User Equipments (UEs). Following this, each function associated with the modified data structures undergoes necessary adjustments.

Subsequently, modification is needed for the FAPI message handler within the lower MAC functional block. The original O-DU faces challenges in handling multiple FAPI PDUs from diverse UEs, particularly concerning UL TTI requests. Thus, it becomes imperative to adapt the FAPI message handler in the lower MAC, enabling it to adeptly traverse each UL TTI request from individual UEs.

In the realm of modifying the scheduler framework, pivotal changes are instituted in two core functions: *schSliceBasedScheduleSlot()* and *schSliceBasedDlScheduling()*. The *schSliceBasedScheduleSlot()* function is responsible for managing Random Access, DL user data, and UL user data, while the *schSliceBasedDlScheduling()* function handles DL user data allocation and PRB allocation to users.

In function *schSliceBasedScheduleSlot()*, the original function maintains the Hybrid Automatic Repeat reQuest (HARQ) list for a single User Equipment (UE) and orchestrates UE removal or reordering on the scheduling list. Post-modification, the maintenance of the HARQ list shifts to the function *schSliceBasedDlScheduling()*. In this altered structure, the UE list is systematically traversed for effective scheduling list management. When a UE does not have UL data and there is no DL data to be sent, no action is taken. However, if a UE has UL data and DL data needs to be sent, the UE node is added to the end of the scheduling list. Conversely, if both DL and UL data from a UE have already been sent, the UE node is removed from the scheduling list.

In function *schSliceBasedDlScheduling()*, the modification involves relocating the generation of HARQ processes to the function's outset and their removal to the conclusion. In addressing the allocation of Downlink Control Information (DCI), a comprehensive loop is implemented to traverse the entire UE list. Concurrently, the UE list is seamlessly integrated into the scheduling algorithm function to facilitate resource allocation. This holistic approach optimizes the scheduler framework for enhanced multi-UE per Transmission Time Interval (TTI) performance.

Following the modification of the scheduler framework, essential changes are implemented to improve the Physical Downlink Control Channel (PDCCH) and the Physical Uplink Control Channel (PUCCH) for multiple UEs. The PDCCH is used to convey control information, such as scheduling decisions and resource allocations, from the gNodeB to the UE. The PUCCH carries control information, including acknowledgments and channel state information, from the UE back to the gNodeB. This adaptation adheres to the specifications outlined in “TS 38.211 section 7.3.2” [24] and “TS 38.213 section 9.2.1” [19], respectively. Specifically, a new function is introduced to allocate resources for PUCCH, accommodating varying numbers of UEs. Unlike the original hardcoded approach limited to a single UE in the OSC scheduler (SCH), this dynamic allocation method ensures adaptability to diverse UE scenarios. the allocation of Control Channel Elements (CCEs) has been expanded from supporting a single UE to accommodating multiple UEs. In our thesis, the allocation ranges from two CCEs for one UE to eight CCEs for four UEs. The spectrum of these implementations is visually represented in Figure 3.1-2.

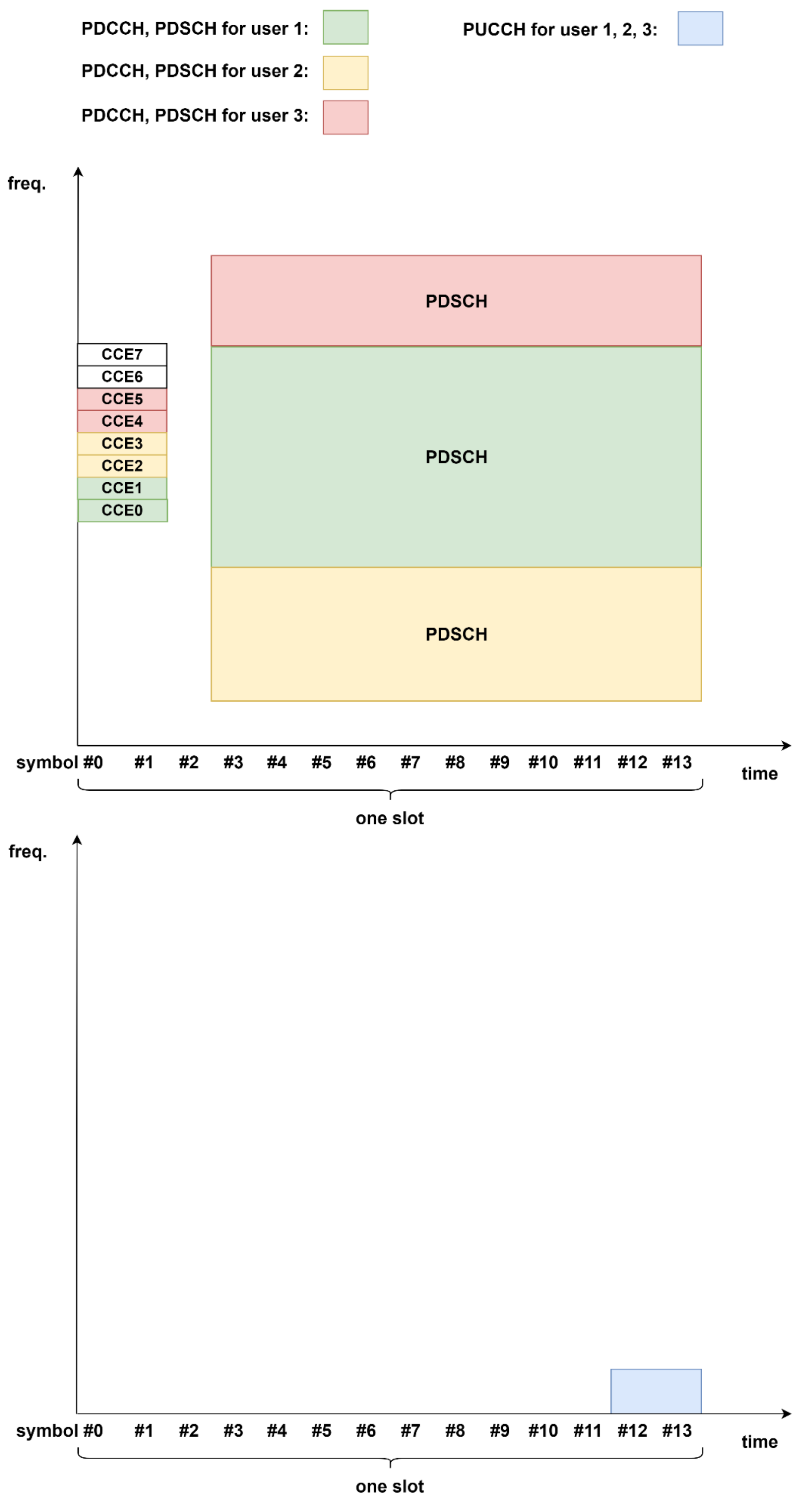


Figure 3.1-2: Resource allocation of PDCCH, PDSCH and PUCCH

3.2 QoS-aware scheduling

In the QoS-aware scheduling implementation, the initial phase revolves around ensuring the accurate encoding and decoding of Quality of Service (QoS) information, encompassing parameters like Guaranteed Flow Bit Rate (GFBR) and Maximum Flow Bit Rate (MFBR) within the Guaranteed Bit Rate (GBR) Information. This process begins with the extension of GBR information in Abstract Syntax Notation One (ASN.1) format based on the F1AP within the *BuildDRBSetup()* function, coupled with ensuring proper memory release in the *FreeDRBSetup()* function. Following the transmission of GBR information from the CU stub to O-DU High, the GBR information is interpreted within the *extractQosInfo()* function. Subsequently, the acquired QoS details are relayed from the DU APP to the scheduler (SCH) by adapting the *schLcCtxt* data structure, with necessary modifications made to functions such as *fillSchDlLcCtxt* (responsible for populating Downlink (DL) logical channel information in the UE control block) and *schSliceBasedFillLcInfoToSliceCb* (tasked with filling DL logical channel information in the slice control block).

After integrating the essential QoS information into the scheduler, a series of new functions are introduced to facilitate QoS-aware scheduling. These include *schGetResourceTypeFromFiveQI()*, *schSortLcByPriority()*, *schMFBRAlgoforLc()*, *schGFBRAlgoforLc()*, and *schQoSBasedAlgo()*. The *schGetResourceTypeFromFiveQI()* function serves to distinguish between GBR and non-GBR traffic based on their 5G QoS Identifier (5QI). Meanwhile, *schSortLcByPriority()* prioritizes traffic for scheduling based on their assigned priority levels. The *schQoSBasedAlgo()* function forms the backbone of the scheduler framework, employing multiple queues to organize GBR and non-GBR traffic lists and scheduling them according to GFBR and MFBR requirements, as illustrated in Figure 3.2-1. Additionally, *schMFBRAlgoforLc()* and *schGFBRAlgoforLc()* represent distinct scheduling algorithms, ensuring that each GBR traffic receives a minimum throughput dictated by GFBR and that both GBR and non-GBR traffic are scheduled while preventing them from surpassing their maximum throughput based on MFBR. These functions collectively enable precise and efficient QoS-aware scheduling within the O-DU High architecture.

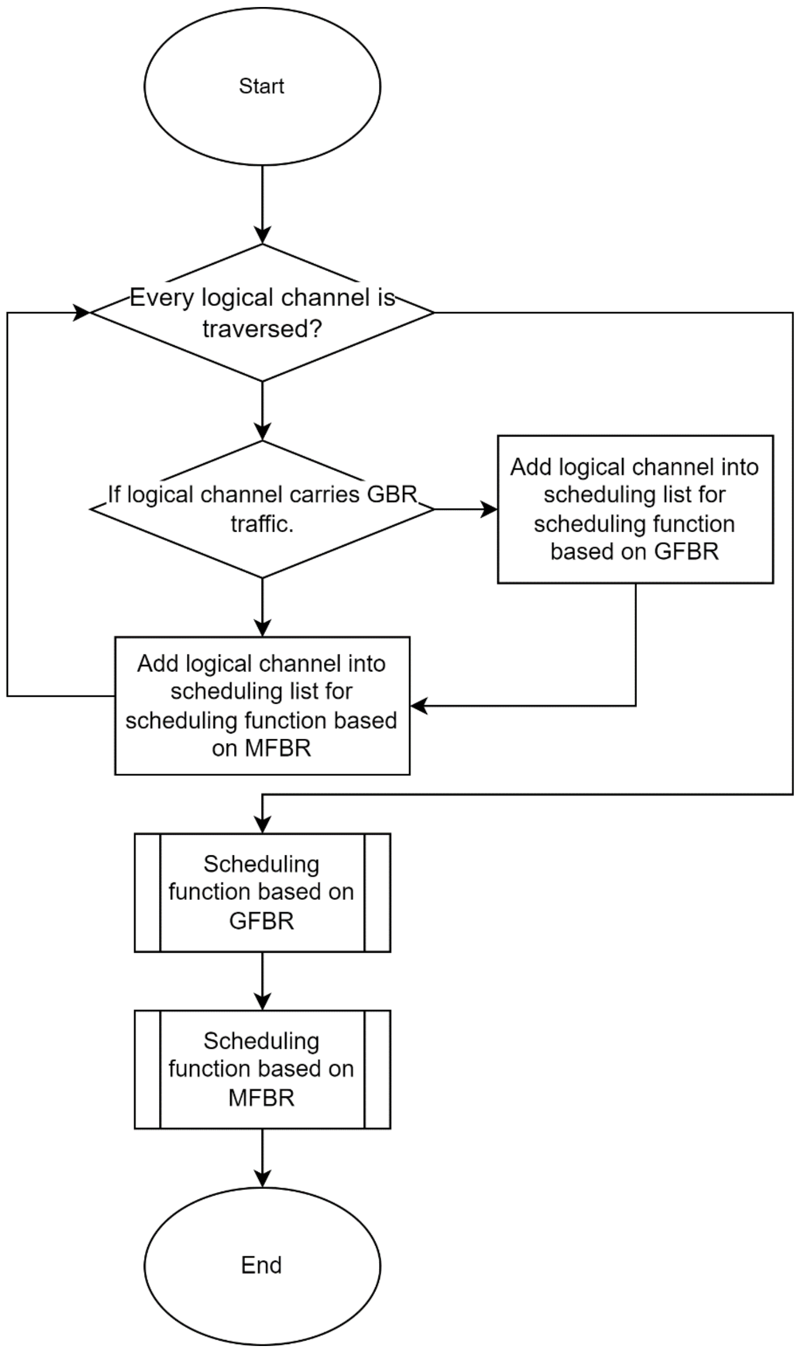

Start
Every logical channel is traversed?
If logical channel carries GBR traffic.
Add logical channel into scheduling list for scheduling function based on GFBR
Add logical channel into scheduling list for scheduling function based on MFBR
Scheduling function based on GFBR
Scheduling function based on MFBR
End

Figure 3.2-1: The flow chart of QoS-aware SCH

In implementing the scheduling algorithm for MFBR, we utilize the leaky bucket mechanism, as illustrated in Figure 3.2-2. However, the current OSC open-source codebase lacks support for this mechanism. Consequently, when attempting to limit the scheduling of Guaranteed Bit Rate (GBR) traffic in the scheduler to prevent exceeding MFBR, packet transmission still occurs at the Radio Link Control (RLC) layer, and it will cause a system crash since accumulated packets can not be consumed. To address this discrepancy, modifications are required in the *RlcProcDlUserDataTransfer()* function. Our proposed solution involves implementing a mechanism to drop packets in the function *RlcProcDlUserDataTransfer()* when GBR traffic exceeds MFBR. Another challenge arises from occasional instances where GBR traffic surpasses MFBR, thereby jeopardizing the guarantee of

Guaranteed Flow Bit Rate (GFBR). This issue stems from the asynchronous nature of timers between the scheduler and RLC layers. To mitigate this, we propose a mechanism to reset the accumulated Buffer Occupancy (BO) for GFBR and MFBR when both the RLC and SCH timers expire concurrently. Through rigorous experimentation, we validate the effectiveness of our QoS-aware scheduler, with detailed numerical results presented in Section IV for comprehensive evaluation.

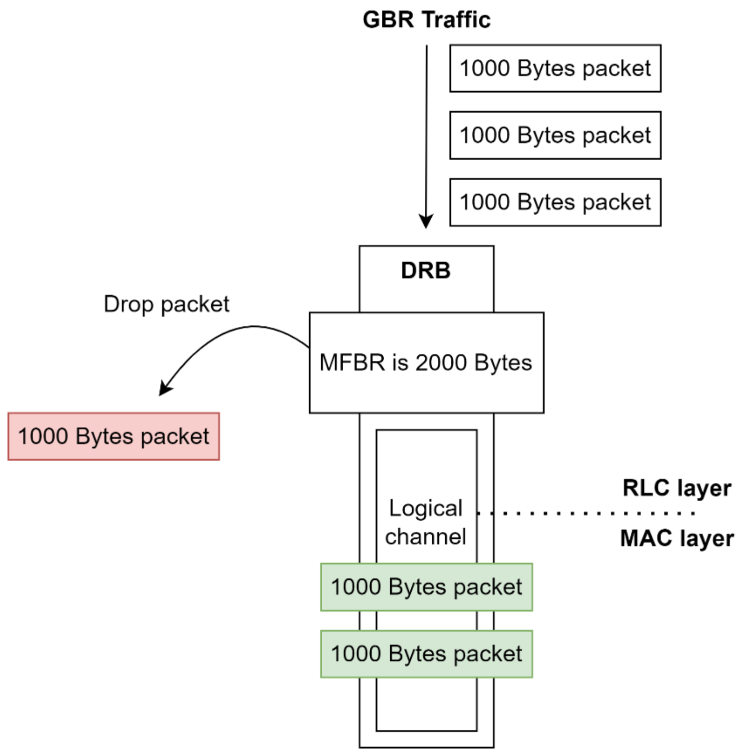


Figure 3.2-2: Illustration of leaky bucket mechanism

Furthermore, our exploration of the MFBR scheduling algorithm uncovered a memory allocation issue during the Uplink Control Information (UCI) allocation process when User Equipment (UE) exclusively handles Guaranteed Bit Rate (GBR) traffic. Specifically, when GBR traffic satisfies the Maximum Flow Bit Rate (MFBR) requirement but remains unscheduled in a given slot, the memory allocation for the *DlMsgSchInfo* data structure is inadequately released, as depicted in Figure 3.2-3. To address this, we developed a solution to ensure proper memory management and alleviate potential system instabilities.

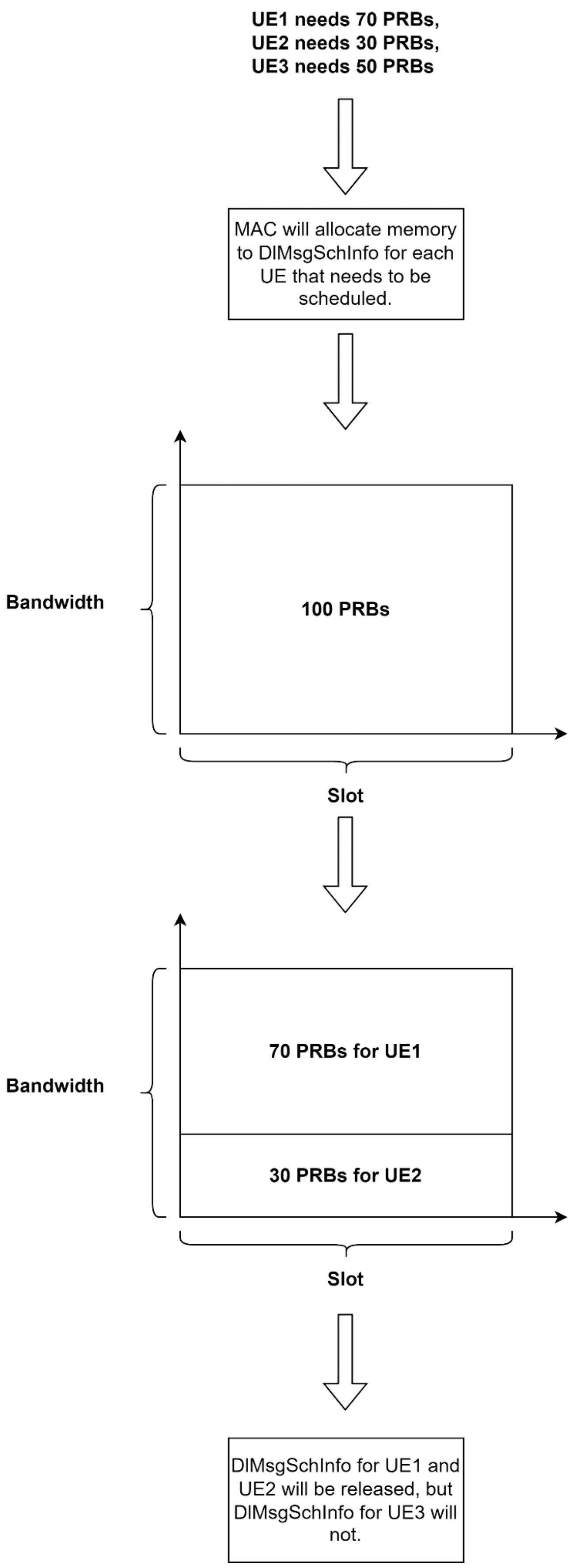

UE1 needs 70 PRBs,
UE2 needs 30 PRBs,
UE3 needs 50 PRBs
MAC will allocate memory to DlMsgSchInfo for each UE that needs to be scheduled.
Bandwidth
100 PRBs
Slot
Bandwidth
70 PRBs for UE1
30 PRBs for UE2
Slot
DlMsgSchInfo for UE1 and UE2 will be released, but DlMsgSchInfo for UE3 will not.

Figure 3.2-3: Illustration of the Memory release problem in OSC SCH

3.3 Retransmission scheduling

In the original OSC scheduler, the functionality for retransmission is not supported. To enable retransmission, we implemented HARQ failure in the *fillPucchF0F1PduInfo* function within the OSC PHY simulator to trigger retransmissions. Following this, we integrate a retransmission queue within the DL dedicated scheduling function.

In the DL dedicated scheduling function *schSliceBasedDlScheduling()*, as depicted in Figure 3.3-1, we implement a retransmission scheduling function called *schRetransmissionQueue()*. Before invoking *schRetransmissionQueue()*, we first identify which UEs require retransmission and add them to a retransmission UE list. We then prioritize these UEs based on the amount of GBR traffic they have or the number of retransmission attempts. After sorting the retransmission UE list, we traverse it to perform retransmissions using the function *schRetransmissionQueue()*.

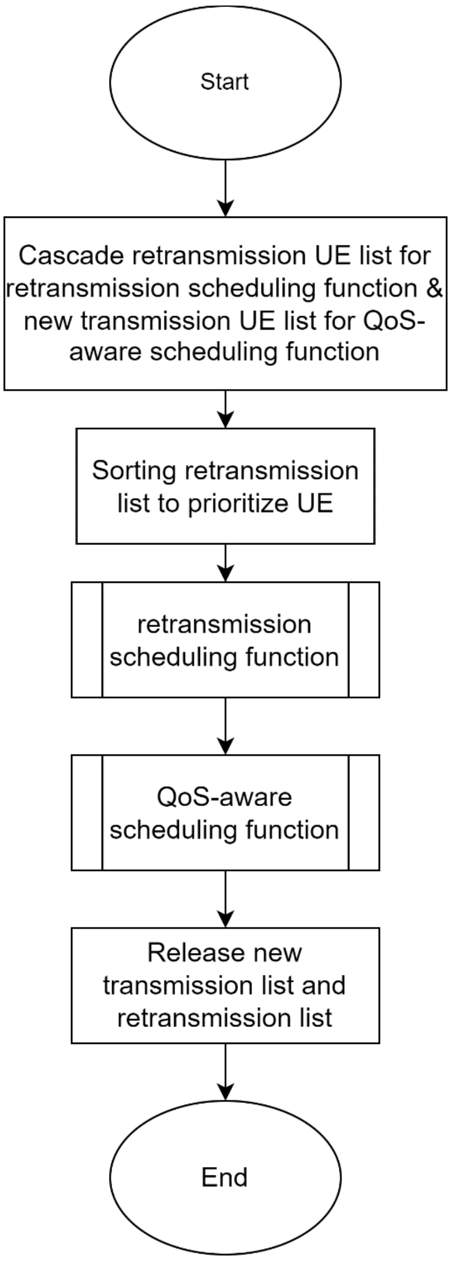

Start
Cascade retransmission UE list for retransmission scheduling function & new transmission UE list for QoS-aware scheduling function
Sorting retransmission list to prioritize UE
retransmission scheduling function
QoS-aware scheduling function
Release new transmission list and retransmission list
End

Figure 3.3-1: The flow chart of DL dedicated scheduling function

For the retransmission scheduling function *schRetransmissionQueue()*, implemented as shown in Figure 3.3-2, we first retrieve the scheduling information for the PDCCH, PDSCH, and PUCCH time slots. Next, we allocate memory for the UE DL scheduling information. Instead of recalculating the transport block size, we reuse the transport block size from the previous transmission. After verifying that the available bandwidth is sufficient for retransmission scheduling, we populate the UE scheduling information with the PDCCH, PDSCH, and PUCCH data.

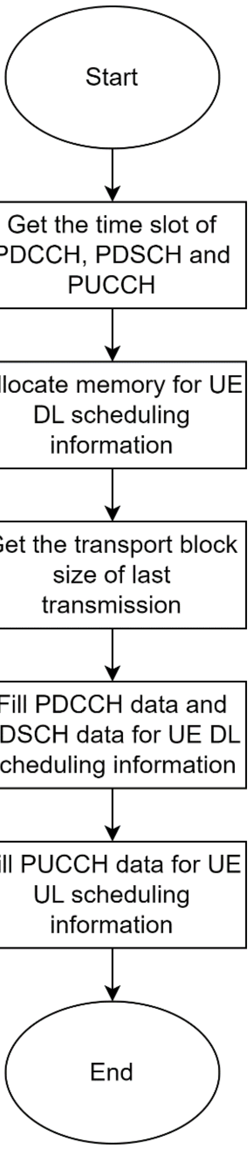

Start
Get the time slot of PDCCH, PDSCH and PUCCH
Allocate memory for UE DL scheduling information
Get the transport block size of last transmission
Fill PDCCH data and PDSCH data for UE DL scheduling information
Fill PUCCH data for UE UL scheduling information
End

Figure 3.3-2: The flow chart of the retransmission scheduling function

4 Experimental Result

In this section, we systematically present our testing methodology and evaluate the performance enhancements achieved through three phase implementations within our testing environment. We introduce the testing environment in the first subsection, providing essential details to contextualize our experiments and analyses. Subsequently, we focus on the performance improvements resulting from the extension of Single-UE (SU) per Transmission Time Interval (TTI) scheduling to support Multi-UE (MU) per TTI scheduling. Demonstrating the efficacy of this extension in improving resource utilization. Following that, we evaluate the effectiveness of our QoS-aware scheduling scheme in the next subsection, considering its impact under different input parameters and analyzing its processing time compared to the original open-source implementation, providing insights into its practical viability. Finally, we present the numerical results obtained from our evaluation of the retransmission queue, shedding light on the effectiveness of our proposed algorithm.

4.1 Testing Environment

To evaluate the system with the implementation of our work, we prepared the following test setup. This setup primarily focuses on evaluating the downlink Radio Link Control (RLC) throughput, a key performance indicator for our system. Our testing configuration involves the utilization of a CU stub, a simulator of OSC O-CU, as a substitute for the O-CU component. This CU stub serves to generate dummy data via the F1-U interface, which is then relayed to the RLC and MAC layers subsequent to the reception of scheduling result information by MAC. This scheduling result information includes the Modulation and Coding Scheme (MCS) index value alongside resource allocation details. The MCS index value guides the determination of the data transmission capacity per transmission instance. Upon receiving this information, MAC initiates a data request to RLC, which, in turn, forwards the requested downlink data to MAC and subsequently to the Physical (PHY) layer via the PHY Stub, a simulator of OSC O-RU. The quantity of data forwarded by RLC is directly ascertainable via the O-DU function. This downlink data serves as the basis for evaluating system performance across diverse channel conditions, with the scheduling task being managed by the scheduler (SCH).

In our test setup, the CU Stub generates synthetic data via the F1-U interface and QoS Information via the F1-C interface. Once MAC receives the scheduling result information, this data is directed to RLC and MAC. The scheduling result information encompasses the allocation of physical resource blocks, determining the volume of data transmissible in a single transmission. Armed with this allocation, MAC requests data from RLC, initiating the transfer of requested downlink data from RLC to MAC and then to the PHY layer (PHY Stub). Both the forwarded data from RLC and the corresponding PRB usage values are readily indicated by the O-DU. So there are two key parameters are recorded during the experiments: downlink throughput, measured in bits per second, which quantifies the rate of data transmission from the base station to the user equipment (UE), and downlink overall Physical Resource Block (PRB) usage, measured as a percentage per second, reflecting the proportion of utilized PRBs relative to the total available.

The conducted experiments aim to present a performance measurement between scenarios for the following subsections, the testing procedure is delineated as follows:

1. Cell-Up Procedure: O-DU initiates the cell-up procedure in collaboration with CU stub.
2. UE Attach Procedure: PHY stub executes the UE attach procedure, involving both O-DU and CU stub.

3. Transmission of Dummy DL User Data: CU stub transmits simulated Downlink (DL) user data to O-DU.
4. Resource Scheduling: The scheduler undertakes resource scheduling for each UE.
5. Throughput Measurement: RLC conducts an evaluation of overall throughput [25] based on the scheduler's report.
6. PRB Usage Measurement: Simultaneously, DU APP measures overall Physical Resource Block (PRB) usage [25] concurrently with the throughput assessment.
7. Repeated Evaluation: Steps 3 to 6 are iterated for 10 seconds, ensuring a comprehensive assessment of the system's performance under different UE scheduling conditions.

The illustration of our environment setup for our thesis is shown in Figure 4.1-1.

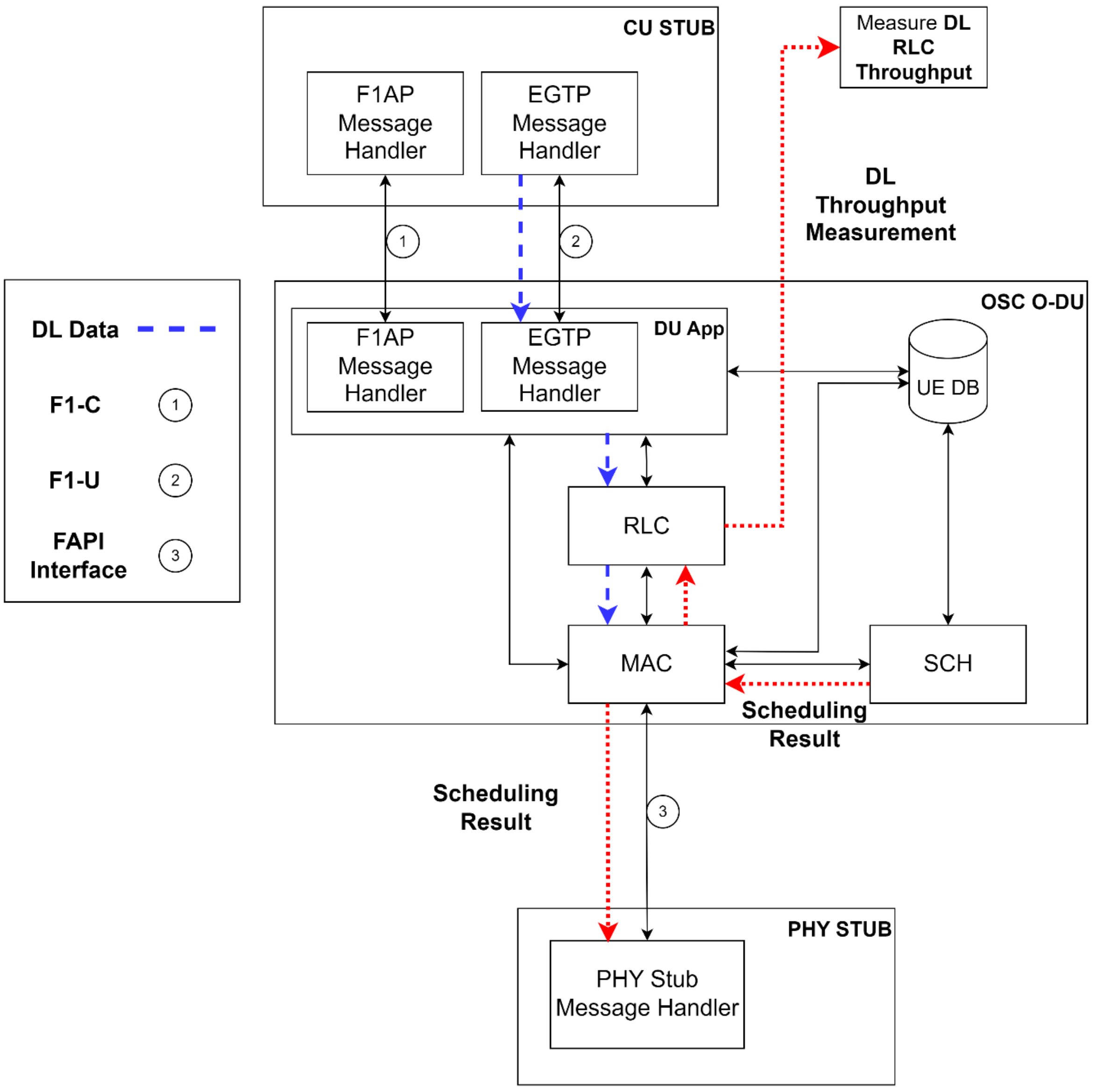
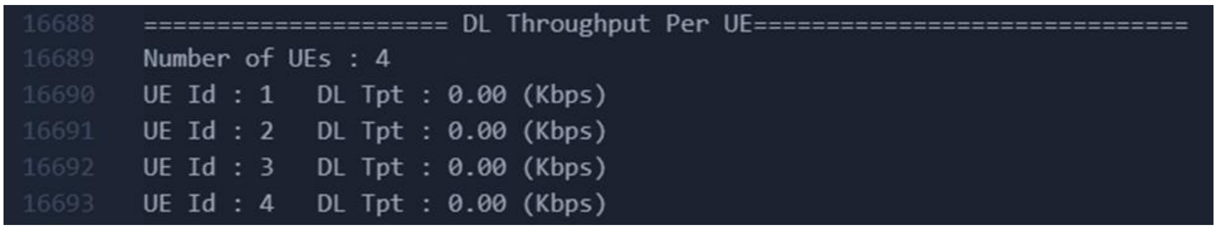


Figure 4.1-1: The testing environment for this paper

4.2 Multiple UE per TTI scheduling

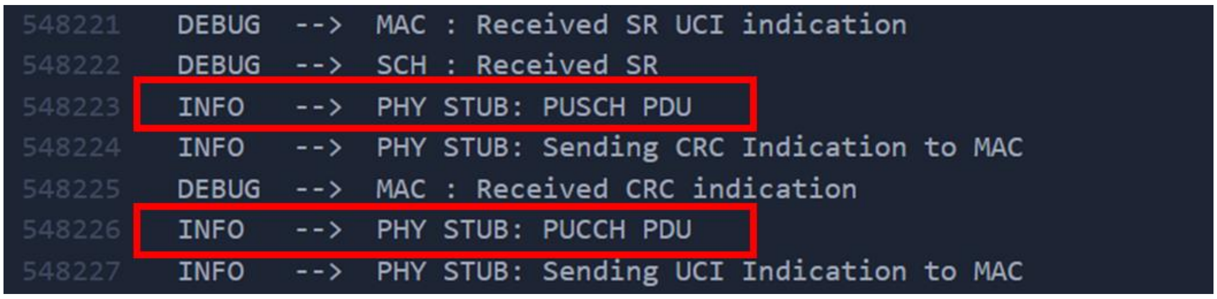
Post-modification of data structure and influenced functions, a comprehensive evaluation of the code's functionality ensues. This evaluation includes assessing the successful attachment of each UE to the RAN, as indicated by their presence on the throughput display list, illustrated in Figure 4.2-1.



```
===================== DL Throughput Per UE===============================
Number of UEs : 4
UE Id : 1   DL Tpt : 0.00 (Kbps)
UE Id : 2   DL Tpt : 0.00 (Kbps)
UE Id : 3   DL Tpt : 0.00 (Kbps)
UE Id : 4   DL Tpt : 0.00 (Kbps)
```

Figure 4.2-1: O-DU Log of UE Throughput list

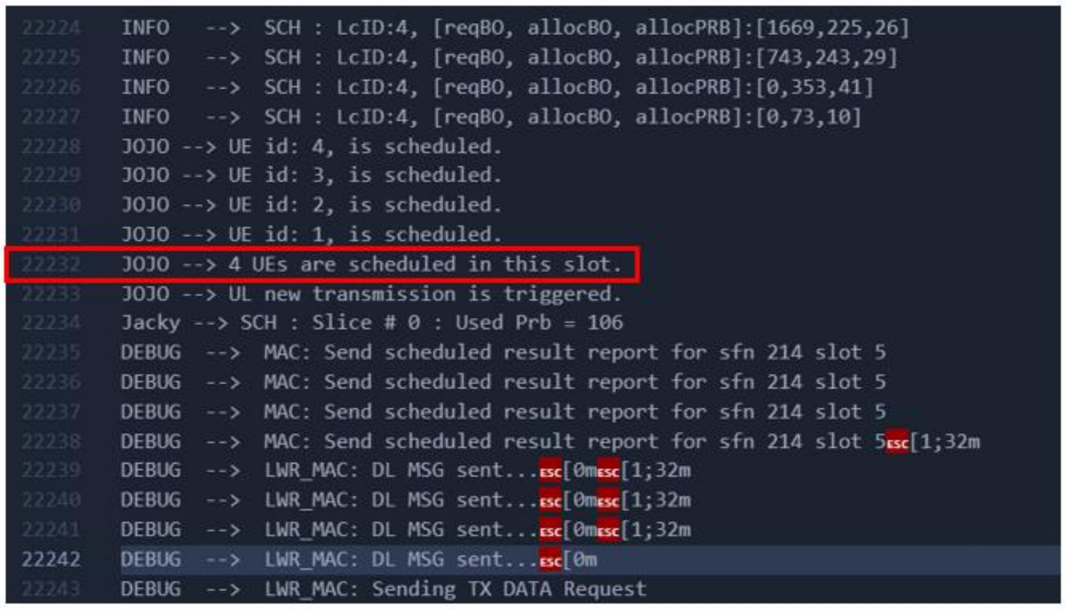
For verification of the functionality of the modified FAPI message handler, an assessment is conducted to ensure the appropriate exchange of FAPI PDUs, as visually depicted in Figure 4.2-2.



```
DEBUG  -->  MAC : Received SR UCI indication
DEBUG  -->  SCH : Received SR
INFO   -->  PHY STUB: PUSCH PDU
INFO   -->  PHY STUB: Sending CRC Indication to MAC
DEBUG  -->  MAC : Received CRC indication
INFO   -->  PHY STUB: PUCCH PDU
INFO   -->  PHY STUB: Sending UCI Indication to MAC
```

Figure 4.2-2: O-DU Log of FAPI PDUs

Upon completing the aforementioned code modifications, a rigorous verification process ensues. This involves a thorough examination of the augmented code through the scrutinization of added logs. The objective is to validate the scheduler's capability to schedule multiple User Equipments (UEs) within each time slot. The log details are presented in Figure 4.2-3, providing a transparent insight into the scheduling performance achieved by the implemented enhancements.



```
INFO   -->  SCH : LcID:4, [reqBO, allocBO, allocPRB]:[1669,225,26]
INFO   -->  SCH : LcID:4, [reqBO, allocBO, allocPRB]:[743,243,29]
INFO   -->  SCH : LcID:4, [reqBO, allocBO, allocPRB]:[0,353,41]
INFO   -->  SCH : LcID:4, [reqBO, allocBO, allocPRB]:[0,73,10]
JOJO --> UE id: 4, is scheduled.
JOJO --> UE id: 3, is scheduled.
JOJO --> UE id: 2, is scheduled.
JOJO --> UE id: 1, is scheduled.
JOJO --> 4 UEs are scheduled in this slot.
JOJO --> UL new transmission is triggered.
Jacky --> SCH : Slice # 0 : Used Prb = 106
DEBUG  -->  MAC: Send scheduled result report for sfn 214 slot 5
DEBUG  -->  MAC: Send scheduled result report for sfn 214 slot 5
DEBUG  -->  MAC: Send scheduled result report for sfn 214 slot 5
DEBUG  -->  MAC: Send scheduled result report for sfn 214 slot 5ESC[1;32m
DEBUG  -->  LWR_MAC: DL MSG sent...ESC[0mESC[1;32m
DEBUG  -->  LWR_MAC: DL MSG sent...ESC[0mESC[1;32m
DEBUG  -->  LWR_MAC: DL MSG sent...ESC[0mESC[1;32m
DEBUG  -->  LWR_MAC: DL MSG sent...ESC[0m
DEBUG  -->  LWR_MAC: Sending TX DATA Request
```

Figure 4.2-3: O-DU Log of scheduled UEs per slot

After confirming the functionality of the scheduler framework for multiple UEs per TTI scheduling, our focus shifts to measuring performance metrics. Specifically, we aim to conduct a numerical comparison between scenarios involving multiple UEs per TTI and those with a single UE per TTI. This comparison will provide valuable insights into the efficiency and effectiveness of the scheduling approach, highlighting improvements and optimizations in network resource utilization and overall performance in terms of throughput.

We configure the parameters for the simulated traffic as outlined in Table 4.2-1. To ensure UE1 will be scheduled and make it less efficient in single user per TTI scheduling, we assign the highest priority to UE1 to prioritize UE1's traffic, but for this case, multiple UE per TTI scheduling will still allocate resources in an efficient way since it can schedule remaining PRBs to other UEs. In this experimental setup, traffic is generated over ten seconds. Following this period, the transmission from UE1 ceases, while other traffic accumulates due to larger packet sizes. Despite this, MU per TTI scheduling demonstrates superior resource utilization, enabling earlier completion of transmissions. This phenomenon is illustrated in Figure 4.2-4.

UE ID	Packet size sent from UE	Transmission interval
UE#1	10 bytes	2 milliseconds
UE#2	150 bytes	2 milliseconds
UE#3	150 bytes	2 milliseconds
UE#4	150 bytes	2 milliseconds

Table 4.2-1: CU Stub DL Packet Generation Parameters for MU per TTI Testing

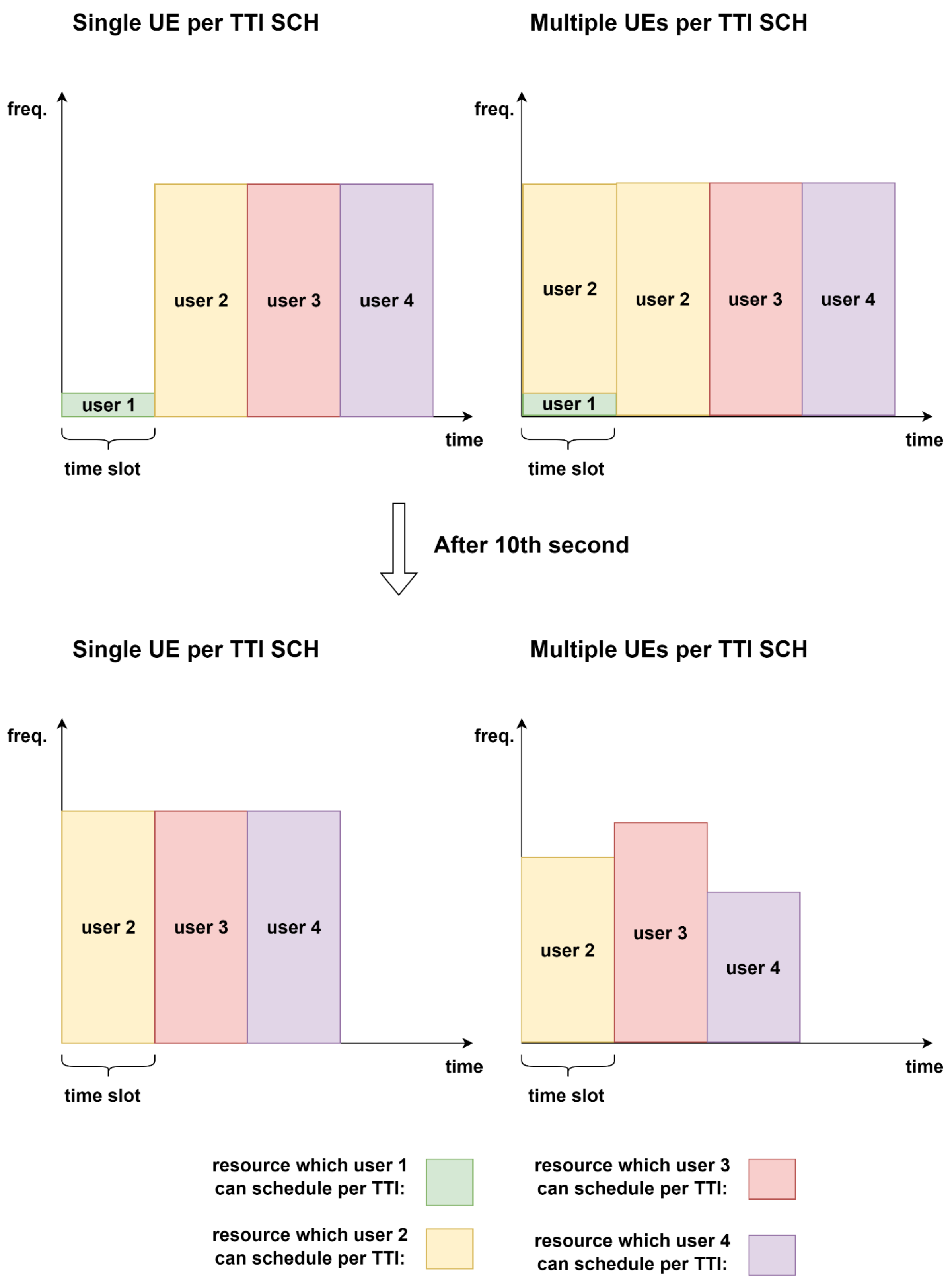


Figure 4.2-4: Illustration of the experiment for SU and MU scheduling per TTI

The resulting numerical insights are illustrated in Figure 4.2-5, and the findings reveal following observations:

1. Multiple UEs per TTI scheduling demonstrates significantly superior Physical Resource Block (PRB) utilization compared to Single UE per TTI scheduling, achieving a 20.9% improvement.
2. Following the tenth second, traffic from all sources except UE1 continues to accumulate due to larger packet sizes, as shown in Figure 4.2-4, necessitating continued transmission. However, MU per TTI scheduling outperforms SU per TTI scheduling by achieving earlier completion of transmissions.
3. The enhanced PRB usage directly contributes to improved overall throughput performance. Consequently, Multiple UEs per TTI scheduling exhibits superior throughput performance compared to Single UE per TTI scheduling, achieving a 15.1% increase in throughput.

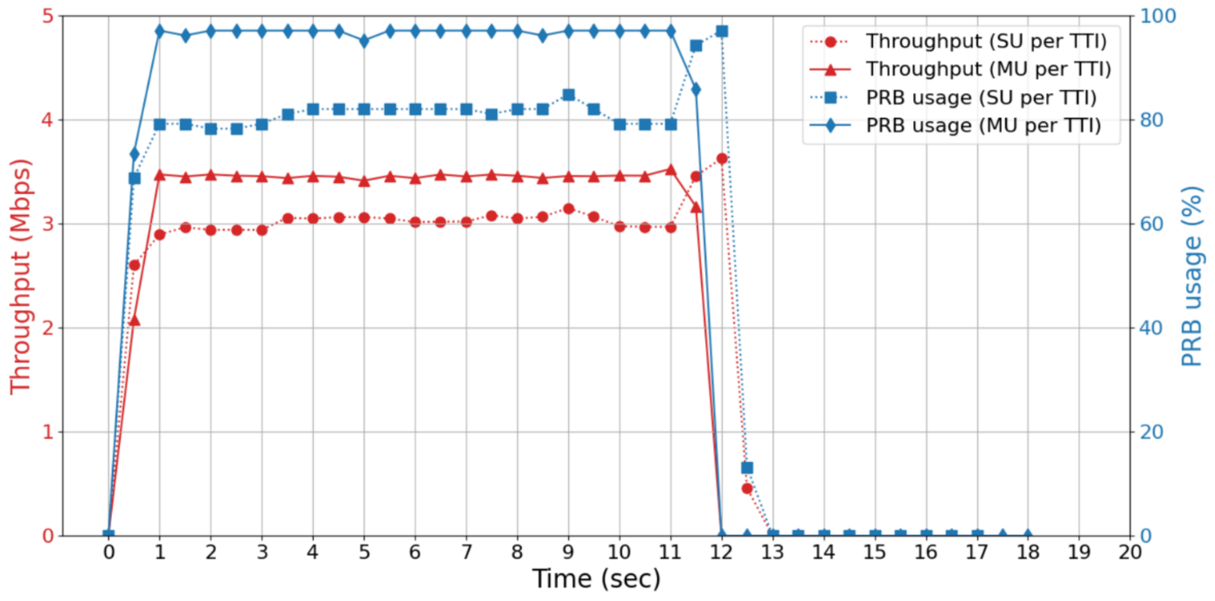


Figure 4.2-5: Downlink overall throughput and PRB usage between SU and MU scheduling per TTI

4.3 QoS-aware scheduling

The numerical results validate the successful implementation of our QoS-aware scheduler, effectively meeting the specified criteria. This scheduler ensures that each Guaranteed Bit Rate (GBR) traffic, prioritized according to its level, adheres to the Guaranteed Flow Bit Rate (GFBR), while simultaneously avoiding exceeding the Maximum Flow Bit Rate (MFBR). Additionally, we test the performance under different input parameters and the processing time compared to the original scheduling function.

4.3.1 Functionalities of GFBR & MFBR

In the first experiment assessing MFBR, the traffic configuration detailed in Table 4.3.1-1 was utilized. This setup involved three GBR (Guaranteed Bit Rate) traffic “Traffic 1, Traffic 2, and Traffic 3” each subject to GFBR and MFBR constraints, alongside a non-GBR traffic exempt from these requirements. The results, depicted in Figure 4.3.1-1, illustrate that the maximum throughput of each traffic remains below the designated MFBR threshold, and the throughput values of non-GBR traffic will not be constrained to a narrow range.

Traffic	MFBR	Priority level	Packet size	Transmission interval
Traffic#1	15000	20	200 Bytes	8 milliseconds
Traffic#2	20000	40	200 Bytes	8 milliseconds
Traffic#3	25000	7	200 Bytes	8 milliseconds
Traffic#4	None	70	200 Bytes	8 milliseconds

Table 4.3.1-2: CU Stub DL Packet Generation Parameters for MFBR Testing Experiment

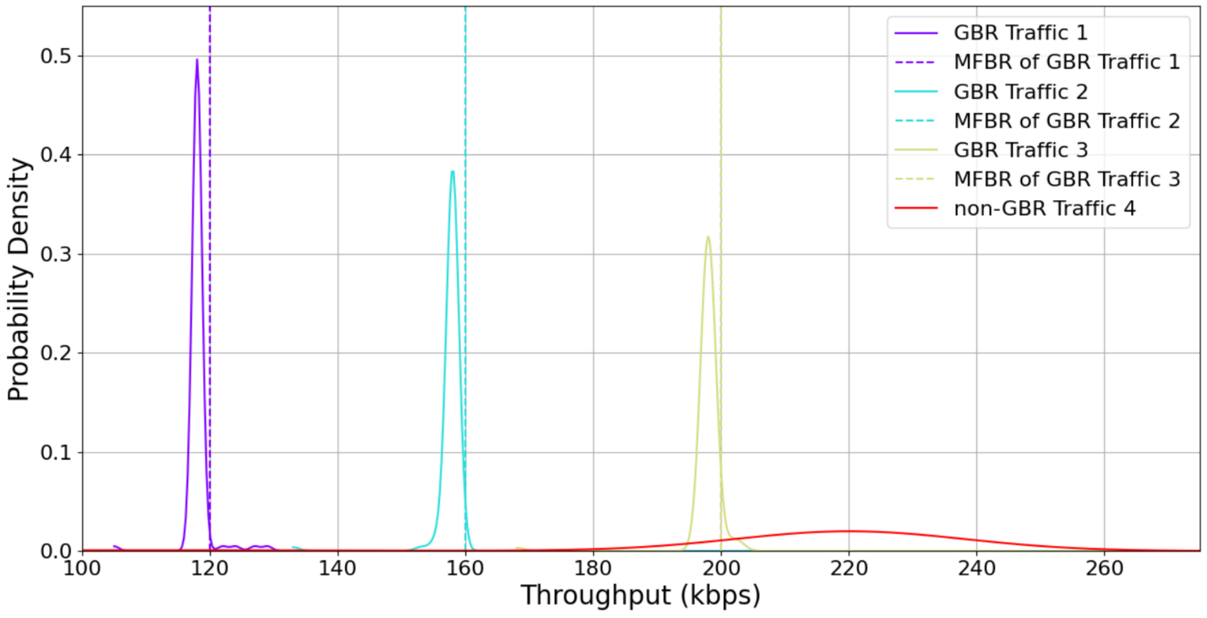


Figure 4.3.1-1: Downlink Throughput of GBR and Non-GBR Traffic for MFBR Testing Experiment

In the subsequent experiment aimed at assessing GFBR, the traffic configuration, as outlined in Table 4.3.1-2, was employed. To achieve full loading, we adjusted parameters by increasing packet size and reducing transmission intervals. As depicted in Figure 4.3.1-2, the results illustrate that once Traffic 1 and 3 fulfill their GFBR requirements, the scheduler allocates resources accordingly before switching to schedule another GBR traffic, Traffic 2.

Traffic	GFBR	Priority level	Packet size	Transmission interval
Traffic#1	150000	20	200 Bytes	4 milliseconds
Traffic#2	170000	40	200 Bytes	4 milliseconds
Traffic#3	190000	7	200 Bytes	4 milliseconds
Traffic#4	None	70	200 Bytes	4 milliseconds

Table 4.3.1-3: CU Stub DL Packet Generation Parameters for MFBR Testing Experiment

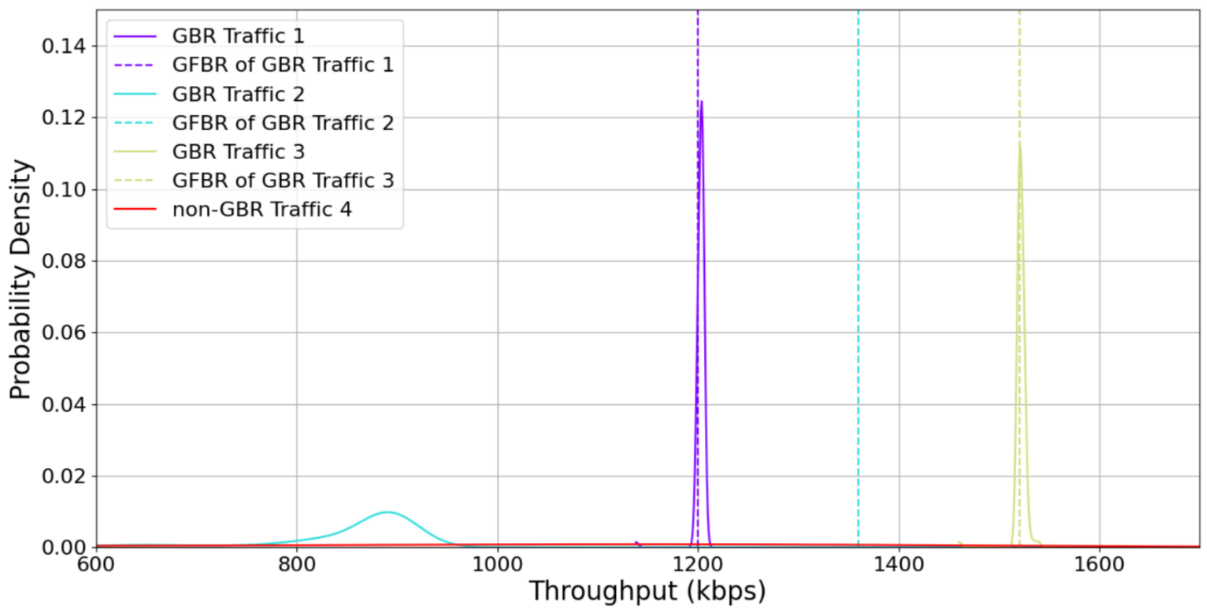


Figure 4.3.1-2: Downlink Throughput of GBR and Non-GBR Traffic for GFBR Testing Experiment

4.3.2 QoS satisfactory

In the third experiment, illustrated in Figure 4.3.2-1, the findings indicate an inverse relationship between the MCS value or Guaranteed Flow Bit Rate (GFBR) value and the overall percentage of Quality of Service (QoS) satisfactory outcomes. Specifically, as the MCS value or GFBR value increases, the proportion of instances where the GFBR of the traffic meets expectations decreases.

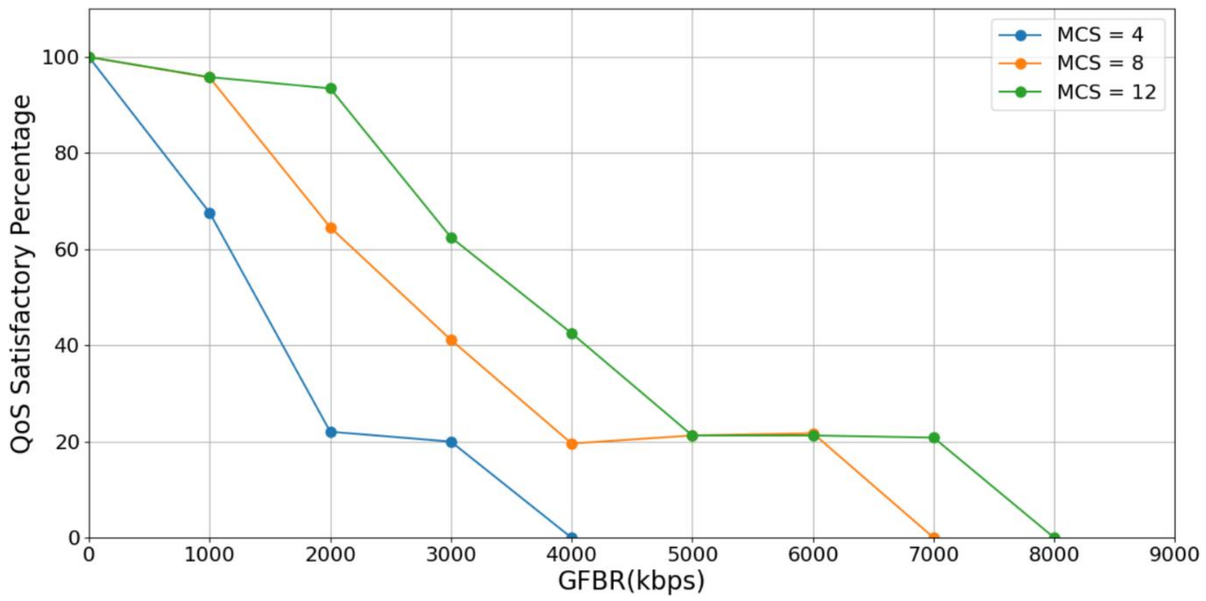


Figure 4.3.2-1: QoS satisfactory under different GFBR values

4.3.3 Effect of average window size

In the fourth experiment, depicted in Figure 4.3.3-1, the impact of varying average windows is examined. In the OSC Radio Link Control (RLC) module, throughput statistics are measured every second. A smaller average window leads to more precise control of the Maximum Flow Bit Rate (MFBR) by the scheduler. This precision enables finer control over the throughput of each traffic, resulting in a sharper Probability Density Function (PDF) curve.

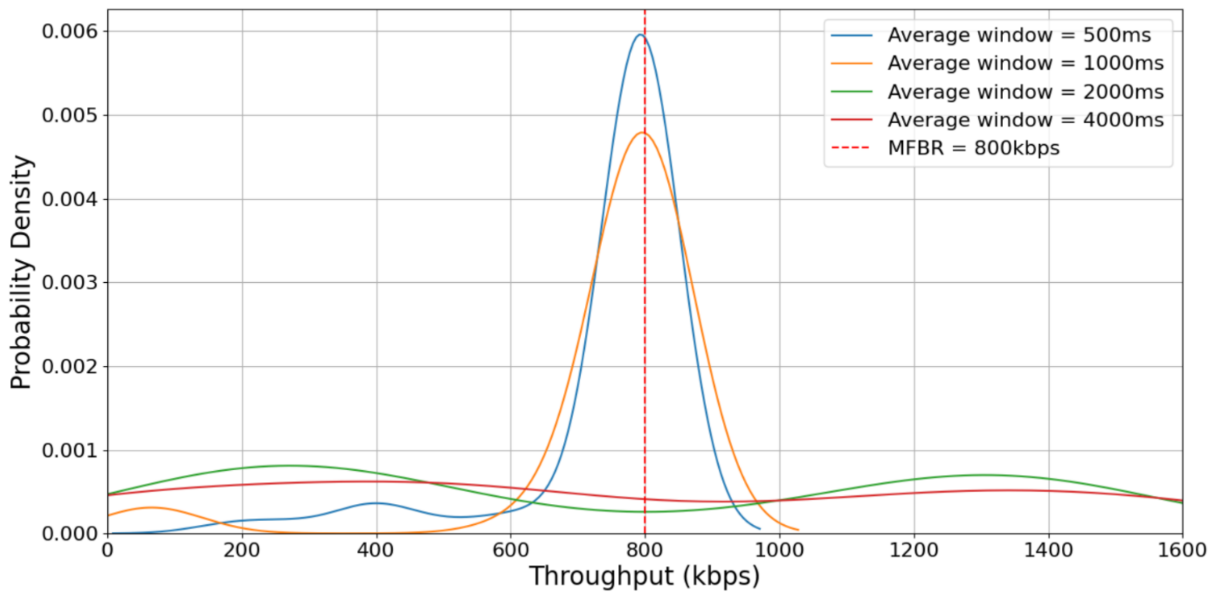


Figure 4.3.3-1: The effect of different average window values.

4.3.4 Processing time

In this section, we conduct a comprehensive evaluation of the processing time for our implemented QoS-aware scheduler. This evaluation includes a comparative analysis with two existing schedulers. The first is the original OSC scheduler, which employs a First-Come-First-Served (FCFS) pattern, prioritizing UEs with earlier traffic. The second is a slice-enabled scheduler developed by the OSC Taiwan lab [26], which supports network slicing and uses a round-robin (RR) scheduling algorithm to manage traffic.

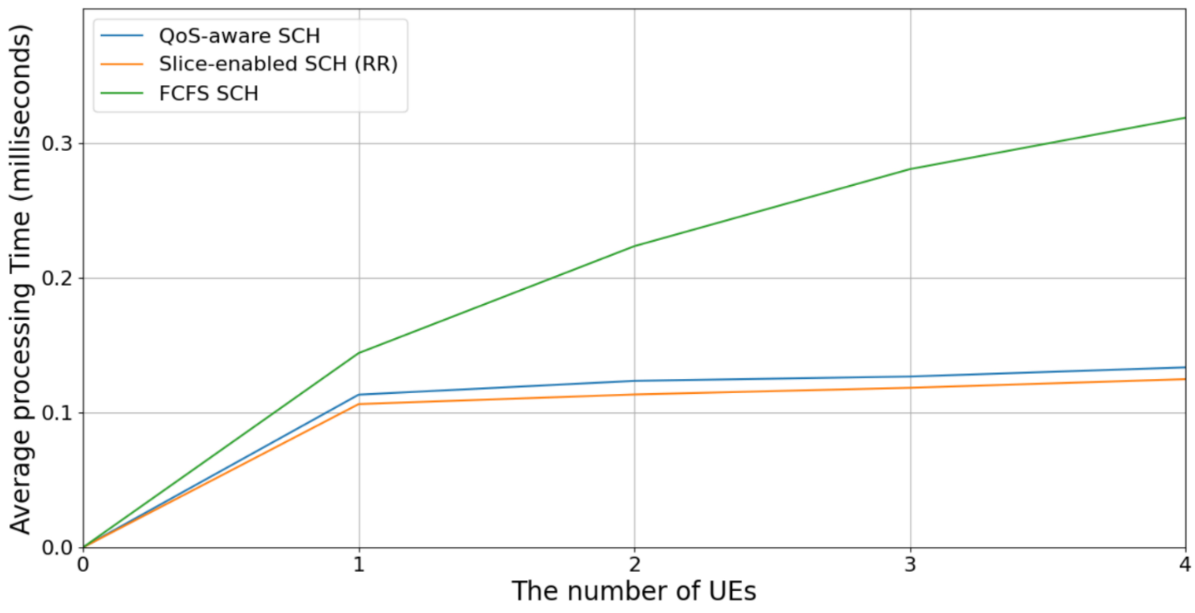


Figure 4.3.4-1: The comparison of processing time between QoS-aware SCH, FCFS SCH, and slice-enabled SCH

The processing time values reflect the efficiency of our scheduler functions during dedicated scheduling. To assess performance trends with increasing numbers of UEs in the system, we incrementally increase the total number of UEs up to four , as illustrated in Figure 4.3.4-1. Through this analysis, we observe notable differences in processing time among schedulers. Specifically, the FCFS scheduler show a significant increase in processing time with each additional UE, demonstrating a 27% longer processing time than the QoS-aware scheduler in the single UE case and a 138% longer processing time in the four UE case. In contrast, both the QoS-aware scheduler and slice-enabled scheduler exhibit only slight increases. Notably, the QoS-aware scheduler has approximately 7% longer processing times than the slice-enabled scheduler due to the additional task of accumulating the counter for the average window at the end of the scheduling function. However, regarding sorting for prioritization and resource allocation, both the QoS-aware scheduler and the slice-enabled scheduler exhibit similar time complexities.

4.4 Retransmission scheduling

For the retransmission scheduling experiment, we utilize the traffic configuration detailed in Table 4.4-1. Four UEs are configured for testing, each with different channel statuses (Block Error Rates, as defined in [27]) and different types of traffic to be transmitted. We then compare two methods of sorting UEs for retransmission, one is prioritization based on retransmission times and another is prioritization based on how many GBR traffics UE has. The results were analyzed based on the average throughput of both new transmissions and retransmissions, as well as the average throughput of retransmissions alone. This analysis provides insights into the effectiveness of different retransmission scheduling strategies.

UE ID	Traffic type	BLER	Packet size	Transmission interval
1	GBR	10%	1200 Bytes	20 milliseconds
2	GBR	30%	1200 Bytes	20 milliseconds
3	Non-GBR	50%	1200 Bytes	20 milliseconds
4	Non-GBR	50%	1200 Bytes	20 milliseconds

Table 4.4-1: CU Stub DL Packet Generation Parameters for retransmission Testing Experiment

From the results depicted in Figure 4.4-1 concerning the average throughput measurement for retransmissions, we note that prioritizing UEs for retransmission based on the number of retransmission attempts leads to higher throughput, particularly for UEs experiencing higher Block Error Rates (BLER) such as 50% error rate in our experimental setup. This outcome stems from the sorting method, which grants UEs with more retransmission attempts and increased scheduling opportunities within each time slot. Consequently, UEs with better channel status, like UE1 and UE2, exhibit little differences in throughput as their retransmissions tend to succeed before the transport block is dropped due to exceeding the maximum retransmission times. Conversely, UE3 and UE4 witness notable improvements of 5% and 19%, respectively, in throughput when employing prioritization based on retransmission times instead of prioritization based on the amount of GBR traffic.

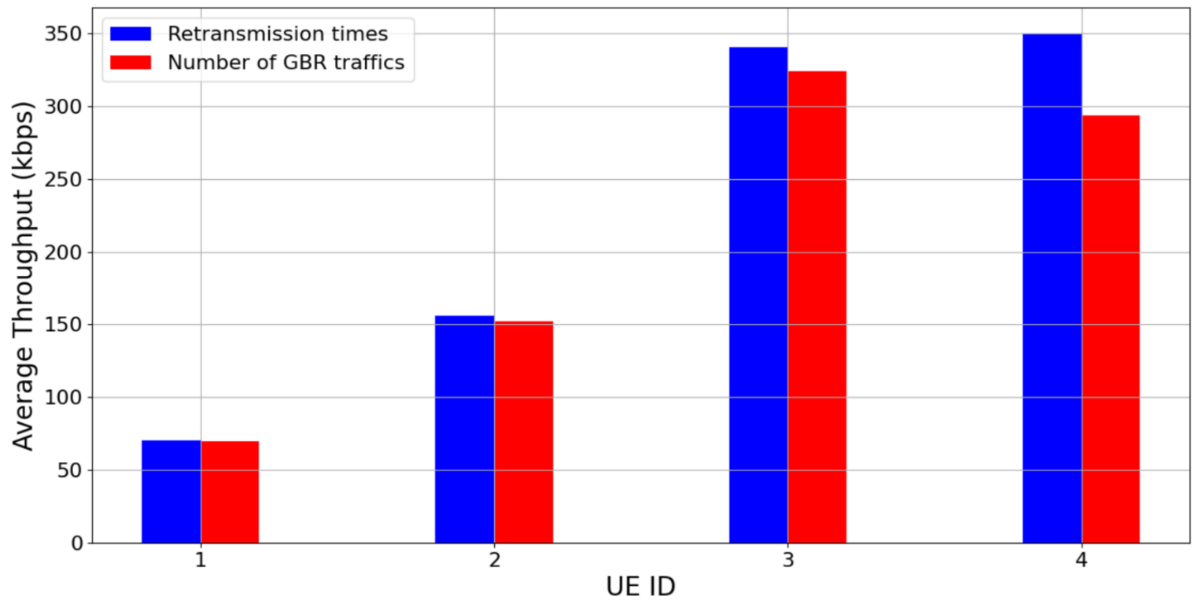


Figure 4.4-1: The average throughput of retransmission

From the results of the average throughput measurement for both retransmissions and new transmissions, shown in Figure 4.4-2, we observe that prioritizing UEs for retransmission based on the amount of GBR traffic each UE carries leads to higher overall throughput compared to prioritization based on the number of retransmission attempts, except for UE4. Specifically, using prioritization based on the GBR traffic amount results in a 2% higher overall throughput. This difference is primarily because, as indicated by the previous results in Figure 4.4-1, prioritization based on retransmission times results in a higher proportion of retransmission scheduling and it will reduce the bandwidth available for new transmissions. Consequently, allocating less bandwidth for new transmissions decreases the total amount of data that can be sent, thereby reducing overall throughput.

The efficiency gained by prioritizing based on GBR traffic highlights the importance of balancing retransmission and new transmission scheduling to optimize network performance. This approach ensures that while retransmissions are handled in a balanced way, sufficient bandwidth is also reserved for new transmissions, maximizing the overall data throughput and improving the quality of service for all users.

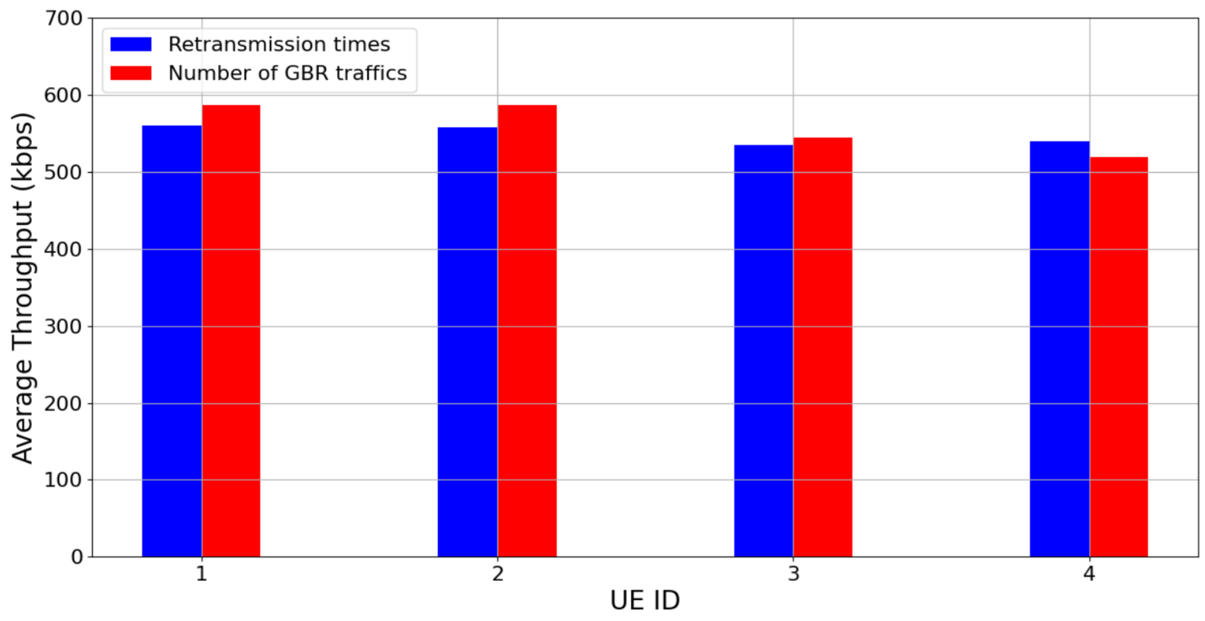


Figure 4.4-2: The average throughput of retransmission and new transmission

Therefore, it can be concluded that prioritizing based on the number of retransmission attempts allocates more bandwidth to retransmission scheduling, enhancing reliability. Conversely, prioritizing based on the amount of Guaranteed Bit Rate (GBR) traffic reserves additional bandwidth for new transmissions, resulting in improved overall throughput.

5 Conclusion

In this thesis, we confront the existing limitations of the OSC O-DU MAC scheduler, which presently confines scheduling to a single UE per TTI and lacks support for rate limiting functionality, primarily due to deficiencies within the open-source framework. To overcome this challenge, we undertake a three-phase implementation strategy aimed at enhancing the scheduler module.

We enhance the efficiency of radio resource usage by introducing a bit rate control mechanism and enabling the simultaneous handling of multiple users. By considering both GFBR and MFBR of individual users, we ensure quality service for all users based on QoS requirements. Additionally, the implemented scheduler addresses the retransmission of erroneous blocks and incorporates two methods of prioritization to ensure fairness among users. We demonstrate that our developed scheduler significantly improves both throughput and processing time compared to the traditional O-RAN scheduler, achieving a 15.1% increase in throughput and reduction in processing time ranging from 21.2% to 58%.

Leveraging the comprehensive architecture outlined in the O-RAN specification, but unfortunately, so far there are no relevant parameters that can be controlled in the resource allocation of GBR traffic and non-GBR traffic over E2 interface, future endeavors may involve the integration of a RAN intelligent controller platform, particularly the Near-RT RIC platform. With its Kubernetes-based infrastructure, the Near-RT RIC platform allows the containerization of intricate scheduling algorithms, referred to as xApps. This approach ensures easy deployment and adaptation of scheduling algorithms based on dynamic operational scenarios.

While the introduction of Near-RT RIC holds promise for enhanced system flexibility, it raises considerations regarding potential performance impacts. The additional workload imposed on the E2 Message handler in O-DU, stemming from reporting and control procedures, introduces complexities. The selection of scheduling algorithms on Near-RT RIC becomes a pivotal factor influencing overall system performance. Although uncertainties linger about performance improvement, the proposed system architecture undeniably empowers more flexible scheduling policy adjustments.

6 Reference

[1] “O-RAN: Towards an open and smart RAN,” 2018. [Online]. Available: https://www.o-ran.org/resources

[2] W. Azariah, F. A. Bimo, C.-W. Lin, R.-G. Cheng, R. Jana, and N. Nikaein, “A survey on open radio access networks: Challenges, research directions, and open source approaches,” Future Wireless Communication Networks (Volume II).

[3] O-RAN Software Community, “O-RAN Software Community (SC).”

[4] O-RAN Software Community (SC) Documentation, "O-RAN Software Community (SC) Documentation." [Online]. Available: https://docs.o-ran-sc.org/en/latest/index.html

[5] O-RAN Software Community, “Code Repositories of OSC on Gerrit.” [Online]. Available: https://gerrit.o-ran-sc.org/

[6] openairinterface5G, “Open air interface software repository on Gitlab,” 2023. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g

[7] srsRAN Project, “srsRAN Documentation,” 2023. [Online]. Available: https://docs.srsran.com/en/latest/

[8] Software Radio Systems, “SRS,” 2023. [Online]. Available: https://www.srs.io/

[9] 3GPP, “TS 38.473 - NR; F1 Application Protocol (F1AP) specification,” Apr. 2024. Rev. 17.8.0.

[10] O-RAN Software Community (SC) O-DU-L2 Documentation, “O-DU High Overview,” 2023. [Online]. Available: https://docs.o-ran-sc.org/projects/o-ran-sc-o-du-l2/en/latest/overview.html

[11] X. Wang, G. B. Giannakis, and A. G. Marques, "A Unified Approach to QoS-Guaranteed Scheduling for Channel-Adaptive Wireless Networks," 2007 Proceedings of the IEEE.

[12] ITU-T, Traffic Control and Congestion Control in B-ISDN, Recommendation I.371, International Telecommunication Union, 2004.

[13] X. Wang, D. Wang, I. Li, H. Zhuang, and S. D. Morgera, “Incorporating retransmission diversity in quality-of-service guaranteed multi-user scheduling,” 2008 42nd Annual Conference on Information Sciences and Systems.

[14] C.-Y. Chang and H.-F. Hsiao, “Priority-based retransmission scheduling for Automatic Repeat Request over wireless networks,” 2015 Seventh International Conference on Ubiquitous and Future Networks.

[15] M. Polese, L. Bonati, S. D’Oro, S. Basagni, and T. Melodia, “Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges,” 2022.

[16] O-RAN Alliance, “O-RAN ALLIANCE e.V.” [Online]. Available: https://www.o-ran.org/

[17] Small Cell Forum, “Small Cell Forum – Accelerating small cell adoption.” [Online]. Available: https://www.smallcellforum.org/

[18] O-RAN Alliance, “O-RAN-WG1-O-RAN Architecture Description—v06.00.00,” Tech. Spec.

[19] 3GPP, “TS 38.213 - v17.2.0 – 5G; NR; Physical layer procedures for control,” Jan. 2022. Rev. 17.0.0.

[20] O.-R. S. Community, “O-DU High Overview,” 2023. [Online]. Available: https://docs.o-ran-sc.org/projects/o-ran-sc-o-du-l2/en/latest/overview.html

[21] 3GPP, “TS 38.322 - NR; Radio Link Control (RLC) protocol specification,” Apr. 2022. Rev. 17.0.0.

[22] 3GPP, “TS 38.321 - NR; Medium Access Control (MAC) protocol specification,” Apr. 2022. Rev. 17.0.0.

[23] K. Juha, “Scheduling,” 2023. [Online]. Available: https://www.3gpp.org/technologies/scheduling

[24] 3GPP, “TS 38.211 – v17.4.0 – 5G; NR; Physical channels and modulation (3GPP TS 38.211 version 17.4.0 release 17),” 2023.

[25] 3GPP, “TS 28.552 – v17.10.0 – 5G; Management and orchestration; 5G performance measurements (3GPP TS 28.552 version 17.10.0 release 17),” 2023.

[26] F. A. Bimo, R.-G. Cheng, C.-C. Tseng, C.-R. Chiang, C.-H. Huang, and X.-W. Lin, "Design and Implementation of Next-generation Research Platforms," 2023 IEEE Globecom Workshops.

[27] 3GPP, “TS 38.300 – v16.4.0 – 5G; NR and NG-RAN Overall description (3GPP TS 38.300 version 16.4.0 Release 16),” 2021

[28] The GitLab repo. of the developed SCH: https://gitlab.fel.cvut.cz/mobile-and-wireless/codes/o-ran/o-du-scheduler/-/blob/main/Multiple-UE-Per-TTI-SCH-qos-aware.zip?ref_type=heads

7 Appendix

In this section, we present the modified file list and the functions that have been updated. Figure 7-1 illustrates these files. The modifications primarily extend based on the slice-based scheduler.

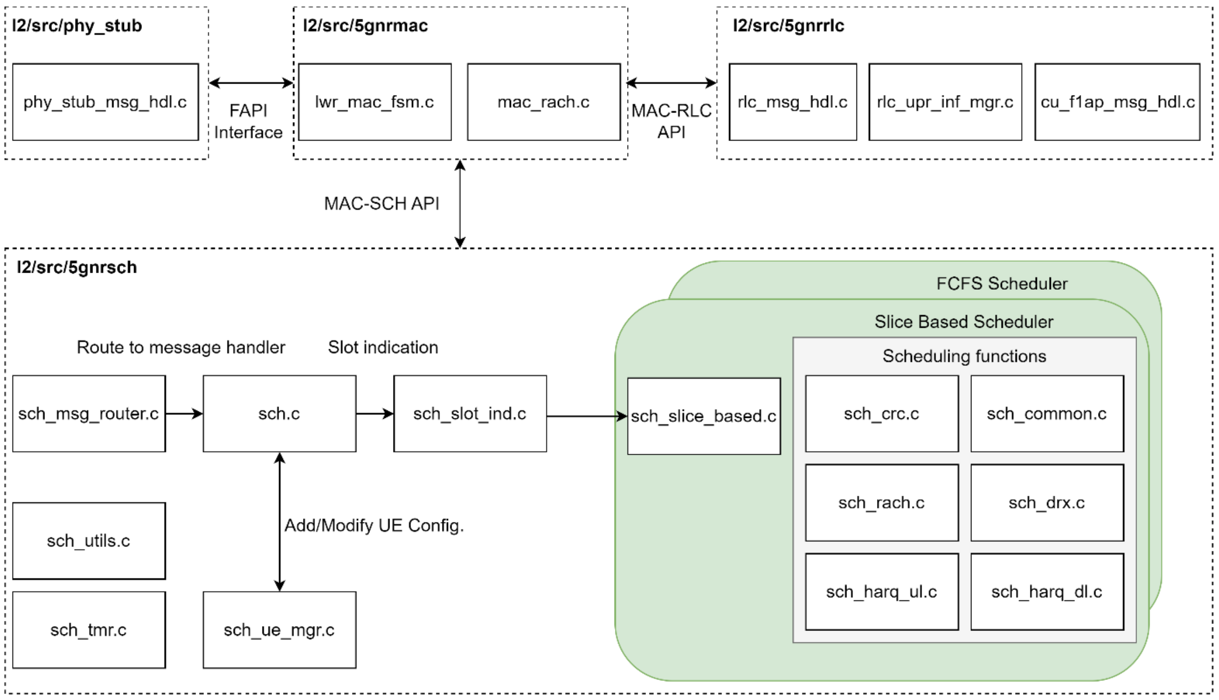


Figure 7-1: Illustration of modified files of the thesis

7.1 List of modified file list for scheduling of Multiple UE per TTI

- File *common_def.h*: Configure the maximum number of UE.
- File *phy_stub.h*: Configure the identifier of UE.
- File *phy_stub_msg_hdl.c*
  - Function *l1HdlUlTtiReq()*: Send different amounts of RACH indications based on the total number of UEs.
- File *sch.h*: Modify data structures *schDlSlotInfo*, and *schUlSlotInfo* to enable multiple UE scheduling per TTI.
- File *mac.h*: Modify data structures *MacUlSlot* to enable multiple UE HARQ per TTI.
- File *sch_utils.h*: Declare new data structure for PUCCH CS set for PUCCH resource set.
- File *sch_utils.c*: Create a new PUCCH CS set for PUCCH resource set.
- File *sch_slice_based.h*: Define parameters *isDlMsgPending*, *isDlMsgScheduled*, *isUlGrantPending*, *isUlGrantScheduled* in data structure *schSliceBasedUeCb* to trace the status of UE data scheduling.
- File *sch_slice_based.c*
  - Function *schSliceBasedDlFinalScheduling()*: Assign UE ID to array elements of PDCCH, PDSCH, and PUCCH for dedicated channel scheduling.

    - Function *schSliceBasedScheduleSlot()*: Check if there is new transmission or retransmission in only function *schSliceBasedDlScheduling()*, and traverse each UE and check if it needs to be removed from UE scheduled list or not.
    - Function *schSliceBasedDlScheduling()*: Collect the UE list for new transmissions and retransmissions that can be scheduled in this DL slot. Check if it is necessary to release the HARQ process for new transmissions and retransmissions, then release the UE list for both new transmissions and retransmissions.
- File *sch_common.c*
    - Function *schProcessMsg4Req()*: Assign UE ID to array elements of PDCCH, PDSCH, and PUCCH for common channel scheduling.
    - Function *schProcessRaReq()*: assign UE ID to array elements of PDCCH, PDSCH, and PUCCH for common channel scheduling, and disable one slot that can only be scheduled one UE.
    - Function *schMsg3RetxSchedulingForUe()*: Disable one slot can only be scheduled one UE.
    - Function *findValidK0K1Value()*: Disable one slot can only be scheduled one UE.
    - Function *schUlResAlloc()*: Allocate PUSCH, PRACH to the fixed position of the array, traverse all PUCCH scheduled for UEs, and fill in PUCCH info.
    - Function *schAllocPucchResource()*: Allocate memory to PUCCH info. and set HARQ info. for specific UE.
    - Function *schDlRsrcAllocDlMsg()*: Change CORESET1 config., assign DCI allocation for multiple UEs.
    - Function *getPucchResource()*: Calculate the resource allocation of PUCCH.
    - Function *schAllocPucchResourceMu()*: Calculate the resource allocation of PUCCH.
    - Function *fillPucchResourceInfo()*: Modify the resource allocation of PUCCH resource set from fixed allocation to dynamic allocation.
- File *mac_rach.c*
    - Function *MacProcUlSchInfo()*: Traverse UL info. for each UE.
- File *lwr_mac_fsm.c*
    - Function *getnPdus()*: Collect the number of PDUs for UL in the current UL slot from each UE by traversing UL info. for each UE.
    - Function *fillUlTtiReq()*: Handle multiple UL PDUs by traversing each UE.
    - Function *fillPrachPdu()*: Fill PRACH info. for specific UE into PRACH FAPI PDU.
    - Function *fillPuschPdu()*: Fill PUSCH info. for specific UE into PUSCH FAPI PDU.
    - Function *fillPucchPdu()*: Fill PUCCH info. for specific UE into PUCCH FAPI PDU.

7.2 List of modified file list for QoS-aware scheduling

- File *sch_slice_based.h*: Add 5QI related parameters into struct *schSliceBasedLcInfo*, and declare the functions for QoS-aware scheduling algorithm based on GFBR and MFBR.
- File *sch_slice_based.c*
    - Function *schSliceBasedFillLcInfoToSliceCb()*: Initialize LC config. in UE config.
    - Function *schGetResourceTypeFromFiveQI()*: To get the resource type of traffic carried by logical channel.
    - Function *schSortLcByPriority()*: Do the sorting for logical channels using insertion sort based on priority level.

    - Function *schQoSBasedAlgo()*: It is a QoS-aware SCH framework, that cascades GFBR LC list, and MFBR LC list, and puts them into scheduling function *schGFBRAlgoforLc()* and *schMFBRAlgoforLc()* for scheduling.
    - Function *schGFBRAlgoforLc()*: Scheduling function for the logical channels for GBR traffic based on GFBR.
    - Function *schMFBRAlgoforLc()*: Scheduling function for the logical channels for each traffic based on MFBR.
    - Function *schSliceBasedDlIntraSliceScheduling()*: Add a new if else condition for QoS scheduling type.
    - Function *schSliceBasedDlIntraSliceThreadScheduling()*: Add a new if else condition for QoS scheduling type.
    - Function *schSliceBasedDlFinalScheduling()*: Add a new if else condition for QoS scheduling type, and release data structure *DlMsgSchInfo* if there is no logical channel has been scheduled or there is no available resource block.
    - Function *schSliceBasedUpdateGrantSizeForBoRpt()*: Check if MFBR is achieved, if yes, it will not accumulate BO in the transport block.
- File *sch_utils.h*: Create 5QI table for table look up.
- File *common_def.h*: Declare the MFBR flag and a flag used to synchronize the timer in RLC and the counter in SCH.
- File *sch_utils.c*: Set the value of 5QI table.
- File *sch.h*: Create new members *gfbr*, *mfbr,* and *avgWindow* in data structure *schLcCtxt*.
- File *sch_slot_ind.c*
    - Function *SchProcSlotInd()*: Accumulate counter for average windows and reset the accumulated BO scheduled for GBR traffics.
- File *sch_ue_mgr.c*
    - Function *getQoSFlowResourceType()*: To check the resource type.
    - Function *fillSchDlLcCtxt()*: To fill GBR info. in the logical channel with GBR traffic.
- File *rlc_msg_hdl.c*
    - Function *RlcProcDlUserDataTransfer()*: To drop PDUs if MFBR is satisfied.
- File *rlc_upr_inf_mgr.c*
    - Function *rlcCalculateTputPerDrb()*: we set the flag of time synchronization as true since the function is called when the timer of throughput measurement is up.
- File *cu_f1ap_msg_hdl.c*
    - Function *BuildDRBSetup()*: Fill out ASN.1 format for GBR QoS Info. for each DRB.
    - Function *FreeDRBSetup()*: Release memory of GBR QoS Info.

7.3 List of modified file list for retransmission scheduling

- File *sch.h*: Set the maximum retransmission times of each UE.
- File *mac_slot_ind.c*
    - Function *MacProcDlAlloc()*: To avoid data pointer *txPdu* is used and causing the segmentation fault because the data pointer *txPdu* is null pointer.
- File *sch_slice_based.h*: Declare the scheduling function for the retransmission queue, and the function to prioritize the retransmission queue.
- File *sch_slice_based.c*

- Function *schSliceBasedDlScheduling()*: Traverse the UEs that need to do retransmission and release HARQ for retransmission.
- Function *schRetransmissionQueue()*: Do resource allocation for retransmission.
- Function *schSliceBasedScheduleDlLc()*: Release data structure *dlMsgAlloc*.

• File *sch_slot_ind.c*

- Function *SchProcSlotInd()*: Calculate the weight of UE retransmission queue based on how many GBR traffic UE has.

• File *phy_stub_msg_hdl.c*

- Function *fillPucchF0F1PduInfo()*: Make HARQ failure to trigger retransmission scheduling.