Enabling Experimentation and Evaluation of xApp Direct Conflict Detection and Mitigation in Testbed

Abida Sultana

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Luiz Antonio Pereira da Silva, Chair
Aloizio Da Silva, Co-chair
Adrian Kliks

May 12, 2025
Blacksburg, Virginia

Keywords: xApp, Near-real time, conflict

Enabling Experimentation and Evaluation of xApp Direct Conflict Detection and Mitigation in Testbed

Abida Sultana

Abstract

Telecommunications networks have evolved from enabling human-focused communication to supporting machine-driven interactions such as Internet of Things (IoT) communication. This shift demands increasingly intelligent and adaptive infrastructures, most notably through the virtualization of the Radio Access Network (RAN). The Open Radio Access Network (O-RAN) architecture introduces the RAN Intelligent Controller (RIC), which enables near-real-time control through modular software applications known as xApps. These xApps deliver various network functions such as mobility management, security, and radio resource management by interacting with RAN nodes via the E2 interface.

The concurrent execution of multiple xApps within the Near Real-Time RAN Intelligent Controller (Near-RT RIC) can lead to resource allocation conflicts, particularly when xApps pursue overlapping or competing control objectives. This thesis presents a complete conflict management mechanism, implemented as part of the Conflict Mitigation (CM) component in Near-RT RIC, that includes both detection and resolution of direct conflicts. This thesis enable for the first time xApp direct conflict detection and mitigation in a Open Radio Access Network (O-RAN) and Software Define Radio (SDR)-based testbed. To validate it a specific use case where multiple xApps attempt to control the same set of Physical Resource Block (PRB)s in a way that causes a conflict is deployed. Further a proposed Deep Q-Network (DQN)-based weighted priority approach to mitigate the conflict is presented. Experimental results show that the DQN agent learns optimal resource allocation strategies, achieving low standard deviation in Downlink (DL) bitrate, minimal latency, and improved reward convergence. These findings validate the feasibility and effectiveness of the proposed DQN weighted priority mechanism in enabling adaptive, conflict-aware xApp orchestration in O-RAN environments.

Enabling Experimentation and Evaluation of xApp Direct Conflict Detection and Mitigation in Testbed

Abida Sultana

General Audience Abstract

With the rapid rise of mobile applications and the Internet of Things (IoT), the way we communicate is evolving faster than ever. This thesis explores how modern technologies can improve telecommunications, particularly through the use of specialized software applications known as xApps. These applications run within a new kind of mobile network architecture called the Open Radio Access Network (O-RAN), which is designed to be more flexible and efficient than traditional systems.

As more people use high-performance applications like virtual reality or real-time video streaming, networks must respond quickly and reliably. To help meet this demand, this work focuses on a critical part of the O-RAN architecture known as the Radio Intelligent Controller (RIC). The RIC uses xApps to manage network resources and improve functions such as performance, reliability, and security.

However, one of the key challenges is conflict resolution—when multiple xApps try to control the same part of the network at the same time. This research presents a solution through a Central Controller Agent (CCA), which is designed to detect and resolve these conflicts in real time. By doing so, it ensures that different applications can work together without degrading network performance.

Overall, this thesis provides practical tools and insights for improving the way future mobile networks operate, making them smarter, more efficient, and better suited to support the growing needs of digital communication.

Dedication

Acknowledgment

I want to sincerely thank everyone who helped and encouraged me along the way. At first, I am grateful to Almighty Allah for guiding and providing me with the strength to undertake this challenging journey.

I am sincerely grateful to my advisor, Dr. Aloizio da Silva, for his guidance, encouragement, and insightful feedback. I also appreciate the support of my fellow colleagues and friends, whose assistance made this experience both enjoyable and enriching.

Furthermore, I would like to acknowledge the resources and facilities provided by my institution, which played a crucial role in my research endeavors. Lastly, I am grateful to my beloved family for their unwavering support and encouragement, which has been my backbone throughout this journey. Thank you all for your contributions and belief in my ability to succeed.

Table of Contents

List of Figures

List of Tables

Acronyms

API Application Protocol Interface.

ASN.1 Abstract Syntax Notation One.

CCA Central Controller Agent.

CCI Commomnwealth Cyber Initiative.

CM Conflict Mitigation.

CMF Conflict Mitigation Framework.

Dbaas Database as a service.

DL Downlink.

DMS Deployment Management Services.

DQN Deep Q-Network.

E2AP E2 Application Protocol.

E2E End-to-End.

E2SM E2 Service Model.

E2SM-RC E2 Service Model RAN Control.

eMBB enhanced Mobile Broadband.

gNB gNodeB.

IE Information Element.

IoT Internet of Things.

JSON Java Script Object Notation.

KPI Key performance Indicators.

KPM Key Performance Measurement.

MIMO Multiple Input Multiple Output.

ML Machine Learning.

MNO Mobile Network Operators.

Near-RT RIC Near Real-Time RAN Intelligent Controller.

NN Neural Network.

Non-RT RIC Non Real-Time RAN Intelligent Controller.

O-RAN Open Radio Access Network.

O-RAN SC O-RAN Software Community.

OSC O-RAN Software Community.

OTA Over-The-Air.

PLMN Public Land Mobile Network.

PR Priority Resolution.

PRB Physical Resource Block.

QoE Quality of Experience.

QoS Quality of Service.

RAN Radio Access Network.

RF Radio Frequency.

RIC RAN Intelligent Controller.

RL Reinforcement Learning.

RMR RIC Message Router.

SCTP Stream Control Transmission Protocol.

SD Standard Deviation.

SDL Shared Data Layer.

SDR Software Define Radio.

SM Slice Management.

SMO Service Management and Orchestration.

srsRAN Software Radio Systems RAN.

TBR Time Based Resolution.

TS Traffic Steering.

UE User Equipment.

UL Uplink.

USRP Universal Software Radio Peripheral.

VM Virtual Machine.

xApp Extended Application.

Chapter 1

Introduction

1.1 Research Motivation

The rapid evolution of mobile communication services, driven by the increasing demand for new use case such as virtual reality and massive machine-type communications, has resulted in an unprecedented increase in data traffic. These emerging applications demand higher Quality of Service (QoS) that is dynamic, intelligent, and flexible. To meet these demands while minimizing capital and operating expenses, the RAN must undergo a significant architectural transformation toward openness and intelligence. Future RAN architectures, such as O-RAN, offer a

layered and modular approach that enables flexibility and cost-efficiency. O-RAN achieves this through disaggregation, virtualization, and support for multi-vendor ecosystems, while introducing the concept of xApps—third-party applications for network control and optimization.

The Near-RT RIC, offers standardized interfaces and hardware support to guarantee interoperability, making it a reliable and safe platform for xApps. However,several xApps created by various vendors run simultaneously, conflicts can occur because of overlapping objectives or incompatible decision-making procedures. Network performance and stability may be hampered by these conflicts, which can be subtle, intricate, and challenging to identify.

Effective CM techniques are desperately needed, as evidenced by the increasing complexity of commercial O-RAN deployments. The following primary motivations are the focus of this work, which attempts to address these issues: Addressing the Knowledge Gap: There is a lack of established methods for evaluating and comparing CM measures in O-RAN environments. This gap in the literature underscores the need for practical, real-world validation of CM.

Real-World Validation: While much prior research has relied on simulations or emulated environments without Over-The-Air (OTA) Radio Frequency (RF) transmission, this work seeks to conduct the real-world assessment of a CM using an actual testbed. This approach enhances the credibility and applicability of the findings to real O-RAN deployments.

Improving Testing and Evaluation of xApp Conflict Mechanism: By implementing conflict detection and resolution mechanisms in a real O-RAN compliant testbed (as opposed to a simulation environment), this research aims to demonstrate a more accurate network stability evaluation and ensure QoS in increasingly complex and heterogeneous network environments.

In conclusion, this work contributes to advancing the experimental and evaluation of xApp conflict mechanisms in O-RAN by addressing interoperability challenges, validating CM methods in real-world environments, and paving the way for more reliable and effective network solutions. This aligns with the overarching goal of enabling agile, intelligent, and flexible RAN

architectures to support the growing demands of modern mobile communication services.

1.2 Research Objectives

Enable for the first time xApp direct conflict detection and mitigation in a O-RAN and SDR-based testbed. Validate it by deploying a specific use case where multiple xApps attempt to control the same set of PRBs in a way that causes a conflict. Present a proposed DQN-based weighted priority approach to mitigate the direct conflict.

To achieve this, xApps were developed and deployed on a SDR-based testbed using srsRAN and Open5GS, allowing for realistic experimentation of conflicting control actions. The proposed conflict mitigation mechanism is implemented as part of the CM component of the Near-RT RIC, incorporating both real-time detection of direct conflicts and a DQN-based dynamic resolution strategy. To validate the effectiveness of the proposed approach, the system is evaluated across multiple network performance metrics such as UE-level downlink throughput, latency, and standard deviation.

Furthermore, a comparative analysis is conducted against proposed DQN mechanism and O-RAN Alliance-recommended resolution approaches—Time-based and Priority-based —to highlight the benefits and trade-offs of AI-driven conflict mitigation over traditional static mechanisms. To our knowledge, this is the first time that O-RAN Alliance-specified conflict mechanisms are implemented in a testbed.

Ultimately, this research contributes to improve operational stability, fairness, and resource optimization in O-RAN environments, contributing valuable insights toward future open and intelligent RAN deployments.

1.3 Research Contributions

The research presented in this work aims to enable experiment-driven analysis and evaluation of xApp direct conflict detection and mitigation strategies. The following are the key contributions of this research.

Contribution 1: First time demonstration and experimentation of xApp direct conflict detection and mitigation in a O-RAN compliant and SDR-based testbed.

Contribution 2: A new DQN-based weighted priority approach for xApp direct conflict mitigation that overcome and complement O-RAN Alliance specified approaches.

Contribution 3: Comparison analysis of the proposed DQN-based weighted priority mechanism against two O-RAN Alliance specified approaches (time-based and priority-based implemented for the first time in a SDR-based testbed) .

1.3.1 Published Work

Along with the aforementioned contributions, this thesis also discusses the following associated research extensions:

1. "A. Tripathi, J. S. R. Mallu, M. H. Rahman, **Abida Sultana**, A. Sathish, A. Huff, M. Roy Chowdhury, and A. P. Da Silva, "End-to-End O-RAN Control-Loop For Radio Resource Allocation in SDR-Based 5G Network", IEEE MILCOM, 2023, pp. 253–254" [1].

2. "**Abida Sultana**, F. Bashar, M. R. Chowdhury, and A. P. Da Silva, "A software-defined radio based o-ran platform for xapp conflict detection and mitigation", IEEE Military Communications Conference (MILCOM), 2024, pp. 686–687" [2].

3. "**Abida Sultana**, C. Adamczyk, M. R. Chowdhury, A. Kliks, and A. P. Da Silva, "Experimental Evaluation of xApp Conflict Mitigation Framework in O-RAN: Insights from Testbed Deployment in OTIC", IEEE International Conference on Computer Communications (IEEE INFOCOM), London, United Kingdom, May 19–22, 2025" [3].

1.4 Related Works

The idea of CM is still in its initial phases within the O-RAN framework but has recently gained significant attention as advancement of O-RAN architecture. The O-RAN Alliance Work Group 3 has released a specification which highlights the challenges and possible solutions for CM in the Near-RT RIC [4]. While this report provides valuable insights regarding CM in O-RAN, it does not include any validation. Resource conflicts among xApps in Near-RT RIC are a critical challenge in Open RAN, especially as independently developed applications simultaneously control overlapping network parameters such as PRB allocation, handovers, and QoS tuning. Several research efforts have proposed detection and mitigation strategies, but most are limited to simulation environments and do not address the complexities of real-world deployment, xApp heterogeneity, or integration challenges.

Zhang et al. [5] published a paper on a related topic. They presented a Deep Q-learning-based team learning algorithm that requires two xApps to collaborate in a O-RAN setting. The mentioned work suggests a practical method for lowering the conflicts through the use of a cooperative ML technique. This approach showed that xApps learn more quickly and converge faster when working together. However, their work remains simulation-based and does not explore the practical integration of collaborative agents in real environments with independently deployed xApps.

Another work by Cezary, [6] describes a Conflict Mitigation Framework (CMF) integrated into the current O-RAN architecture to enable the Conflict Mitigation technique in O-RAN's Near-RT RIC to identify and mitigate all types of O-RAN specified conflict. A simulation of an O-RAN network is used to demonstrate the suitability of the suggested CMF. The simulation's results demonstrate that, with a minor decrease in network reliability, turning on CMF enables balancing the conflicting xApp' network control abilities greatly enhance the network performance. The trade-offs involved in prioritizing one xApp's control decisions over another could

lead to unintended performance degradation in certain scenarios. The implementation of the framework and the resolution methods may face challenges in real-world networks due to the complicated architecture of RAN environments and potential integration issues with various xApps from different providers

Del Prever et al. [7] describe a framework for CM called PACIFISTA. This work bridges the topic of CM and O-RAN testbeds, as this framework is another solution for O-RAN tested using Colosseum wireless network emulator, providing pre-deployment insights to mitigate conflicts and optimize xApp/rApp deployment. In contrast to conventional configurations, Colosseum RF channels and uses RF cables for communication rather than OTA RF transmission between the RAN and UEs. A profiling pipeline conducts sandbox tests, generating statistical profiles to predict and assess conflicts, identifying affected KPMs and control parameters.

Another work by Arshia et al. [8], introduces a data-driven method for detecting and managing conflicts caused by third-party applications (xApps) within the network. The proposed GNN-based approach has been validated primarily using simulated datasets derived from conflict models in the literature. While these simulations demonstrate promising results, they may not capture the full complexity, variability, and unpredictability of actual operational environments in live networks.

The scheduler-based conflict mitigation framework [9] for O-RAN environments introduces a novel, intent-driven approach that significantly advances the management of multiple pre-trained xApps without requiring retraining after deployment. At the heart of this contribution is a dynamic scheduler leveraging the Advantage Actor-Critic (A2C) reinforcement learning algorithm, which intelligently selects the most suitable xApps based on contextual network conditions and target KPM. This method demonstrates improved coordination and performance in scenarios involving indirect or overlapping control logic, outperforming naive or independent xApp deployments. The framework is based on controlled testbed simulations rather than

real-world deployments, leaving potential implementation challenges unaddressed.

The work [10] introduces a novel Zero-Touch Management (ZTM) framework for conflict mitigation in O-RAN using Multi-Agent Reinforcement Learning (MARL), enhanced with a Graph Convolutional Network (GCN)-based attention mechanism. The proposed system enables efficient, scalable resource allocation by allowing selective communication among xApps managing different network slices. This reduces overhead and enhances performance, particularly in large-scale scenarios, showing improved slice satisfaction and throughput distribution compared to traditional MARL approaches. Nevertheless, the framework assumes idealized coordination across slices and lacks validation on real RAN platforms. The effectiveness of this framework may vary under dynamic network conditions and real-world complexities not fully captured in simulations.

While the aforementioned works offer valuable insights into conflict mitigation strategies in O-RAN, many of them are either limited to simulations, rely on static configurations, or lack real-time adaptability in dynamic environments. In contrast to these prior works, our approach focuses on direct conflict resolution among concurrently operating xApps using a DQN-based learning agent. Unlike simulation-driven studies, we deploy and evaluate the framework in a live O-RAN testbed using srsRAN and Open5GS. Our system not only detects and resolves PRB conflicts in real time but also adapts to dynamic radio conditions without requiring pre-coordination or retraining of xApps. Unlike prior simulation-based approaches [5], [11], or profiling-based methods [7], this work actively learns optimal PRB allocation strategies through trial-and-error interactions within a live network, adapting to conflicting control decisions in real time. Moreover, it does not assume cooperation among xApps [5], but instead introduces a centralized decision-making component that dynamically adjusts resource allocation based on observed system performance. This offers a practical step forward in building intelligent and resilient orchestration mechanisms for future disaggregated RAN deployments.

1.5 Thesis Structure

This thesis is outlined into six chapters. Chapter 2 provides the background of the Near-RT RIC architecture, detailing its sub-components and the related interface to communicate with E2 node. Chapter 3 presents problem statement of this work in an O-RAN use case.

Chapter 4 illustrates the methodology of the work, providing details of resource allocation xApps and their routing mechanism in the Near-RT RIC. Chapter 5 provides the overall experimental setup along with results and evaluation of the proposed methodology. Additionally the chapter presents the research challenges encountered during the real-world implementation and the lessons learned from deploying and evaluating xApp conflicts in a live testbed.

Lastly, Chapter 6 completes up the thesis by providing an overview of the research findings and results, and outlining potential directions for further contributions.

Chapter 2

Near-RT RIC and srsRAN Communication

This chapter explores the key components of O-RAN, focusing on the RIC platform, which enables network automation, intelligent control, and orchestration of network functions. This chapter also outlines two foundational elements of this thesis: (1) the O-RAN architecture, including the RIC platform and its applications (xApps), and (2) the challenges and opportunities in conflict mitigation within O-RAN, crucial for ensuring optimal network performance and reliability.

2.1 Background

In wireless communication, O-RAN architecture represents a revolutionary method of building and managing RAN. O-RAN seeks to transform conventional RAN architectures by embracing the concepts of openness, virtualization, and interoperability, allowing for increased flexibility, scalability, and efficiency in network deployments. With the help of cloud-native technologies,

open interfaces, and the disaggregation of RAN components, this novel architecture makes multi-vendor deployments and dynamic network optimization possible. Using distributed architecture and open interfaces, O-RAN is a more open and interoperable RAN concept. The RIC platform is essential to the O-RAN architecture because it facilitates network automation, intelligent control mechanism implementation, and network function orchestration. It lowers the cost of deployment and management by enabling various vendors to create compatible components. Near-RT RIC use dynamic adaptation to the current state to monitor, predict, and optimize network behavior.

Applications on the RIC platform fall into two categories: near-real time xApp and non-real time (rApps), each of which performs a different role in network optimization and management. In response to shifting network conditions and user demands, the RIC platform synchronizes dynamic network functions and services. O-RAN has numerous advantages, but it also brings with it a host of new difficulties because of disputes between different agents that are in charge of the RAN. By properly configuring the RAN control agents, the majority of these conflicts can be avoided completely or at least minimized in advance.But xApp and rApps can be created and implemented quickly in both Near-RT RIC and Non-RT RIC, when they are implemented widely, new kinds of unforeseen conflicts might appear. Network performance may decline as a result of RAN control conflicts that arise during network operation. Therefore, a key component of guaranteeing the dependable and effective operation of future RANs is conflict mitigation in O-RAN.

2.2 Near-RT RIC Architecture

Near-RT RIC consists of numerous subcomponents to allow near real-time (10 ms-1s) operation and optimization of E2 Node resources to enable detailed data measurements over E2 interface [12]. Figure 2.1 represents the O-RAN Software Community (OSC) Near-RT RIC

internal architecture. All the components of RIC architecture deployed as a micro services and is managed as Kubernetes cluster. According to the architecture, it provides northbound Application Protocol Interface (API)(O1 and A1) and southbound API (E2) to communicate with external interface. The E2 interface is used by the Near-RT RIC's xApps to gather data in near real-time and offer additional functionality. To establish End-to-End (E2E) communication, Near-RT RIC platform include the following components:

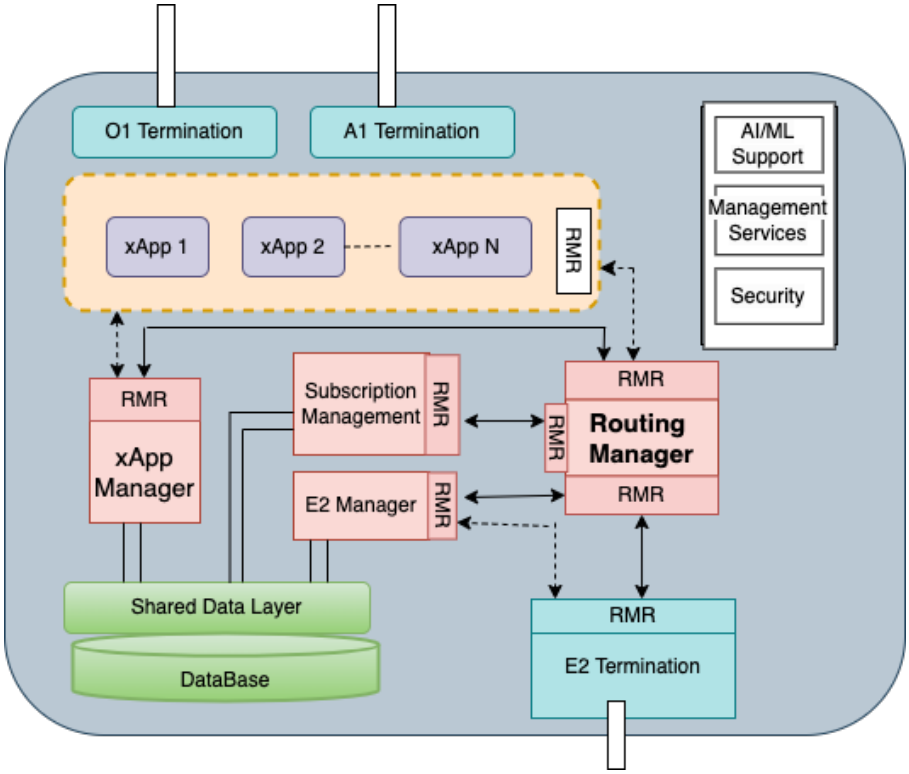

Figure 2.1: Architecture of Near-RT RIC

2.2.1 Routing Manager

One of RIC's fundamental platform services is Routing Manager. It is in charge of allocating routing policies to the various xApps and platform components. Inter-platform messaging within Near-RT RIC is made possible by Routing Manager. It generates and provides routing policies to all deployed xApps and other sub-components. The internal messaging infrastruc-

ture, known as the RIC Message Router (RMR), uses these routing policies to identify the appropriate messaging endpoint.

2.2.2 RIC Message Router

The RIC Message Router (RMR) functions as a peer-to-peer communication library, enabling applications to exchange messages. The library manages flow of messages and destination selection based on the message type. The RMR serves as a message network within the platform and is utilized by all components in Near-RT RIC, as it operates in a decentralized manner rather than as a central platform component. The sender and recipient message endpoints are configured at designated ports by the RMR on the sub-modules inside Near-RT RIC. With RMR, an application can communicate with other peer applications and send and receive messages with little effort on its part. RMR handles all endpoint data, connection details, and routing data required to initiate and sustain communication in order to accomplish this goal. According to the application, sending a message only requires allocating a message buffer (through RMR), adding the data, and configuring the format of message [13]. Upon arrival of a message, the function returns a data buffer; the framework only requires using the receiver function to obtain a message.

2.2.3 E2 Terminator

In order to utilize RIC features and functionality through E2 interface, all E2 Nodes use the E2 Terminator as their gateway. It is designed to monitor all newly introduced E2 setup interaction from E2 Nodes using a Stream Control Transmission Protocol (SCTP) link. In order for the Routing Manager to configure routing policies that permit RMR messages to reach the E2 Node, the E2 Terminator records the E2 Node's authentication information in the database during the E2 connection setup. The E2 Terminator routes messages between the Near-RT

RIC and RAN nodes while maintaining E2 interface connectivity between them.

Abstract Syntax Notation One (ASN.1)-decoded messages are routed to the desired address via the RMR library, using the routes advertised by the Routing Manager. It communicates with the E2 Nodes via the E2 interface by sending E2 Application Protocol (E2AP) messages to them and receiving ASN.1 messages to them. Using the routes advertised by the Routing Manager, the RMR library routes the ASN.1-decoded messages to an intended destination for internal communication within the Near-RT RIC.

2.2.4 E2 Manager

One Near-RT RIC platform can manage several E2 node instances through the use of the E2 Manager, which serves as a scalable architecture. The functionality of the connections between multiple E2 Terminator instances and E2 Nodes is verified. Following any modification in network, such as an established connection or disconnection of RAN, the Routing Manager updates all internal components and deployed xApps with the newly attached E2 Node endpoint. The Routing Manager updates all internal components and deploys xApps with the newly attached E2 Node endpoint after any configuration updates, such as the E2 Node being connected or disconnected to a managed E2 Terminator instance.

2.2.5 Subscription Manager

In RIC, subscription manager is a fundamental platform service. xApps' E2 subscriptions to the E2 Node are managed by it. xApp can use subscription manager to subscribe to E2 Node. From the E2 Node, xApp can subscribe to services of the REPORT, INSERT, CONTROL, and POLICY types. E2 Node returns E2 Indication messages to xApp for a few of those subscription types. A route between the E2 Termination and the xApp is established for those

messages. E2 subscriptions are managed by subscription manager, which also handles message request routing from Routing Manager for the subscribed messages. When two xApps start the same subscription after one has already been made, subscription manager merges the two xApps internally, asks routing manager to create a route for the second xApp, and replies to both xApps. When an xApp submits an accurate subscription request, the subscription manager produces an individual subscription ID and allocates it that xApp. After a successful subscription responding from the E2 Node, the routing manager adds the assigned subscription ID to the routing table to allow additional communication between the xApp and RAN node.

2.2.6 xApp Manager

The life-cycle management of xApps is under the control of the xApp Manager. Working Group (WG) 3 specifications map the functions of this component to the management API for the Near-RT RIC [14]. These specifications state that xApp deployment and management will be handled by the northbound Service Management and Orchestration (SMO) entity. According to the OSC Near-RT RIC, xApp deployments are currently initiated by deploying them onto a Deployment Management Services (DMS) agent and running the installed xApp as a docker service. The Routing Manager notify all platform components that the authorized xApp has been implemented into RIC network and updates the routing table after a successful registration. The xApp Manager is responsible for this.

2.2.7 RIC Database

The RIC database repository, which contains all the components required to deploy Database as a service (Dbaas) to docker container, is where the Near-RT RIC platform keeps RAN data. A single container running a Redis database is used to implement the Dbaas service. The database is set up to be neither redundant nor persistent. To improve service delivery, these

databases keep track of the RAN data that xApps use. Details like RAN Node address, its matching Public Land Mobile Network (PLMN) number, and its connection information, for instance, can be included in the RIC database.The SDL API is used to abstract the databases in order to guarantee high availability, scalability, load balancing, and database integrity and monitor-ability. All Near-RT RIC components and approved xApps have access to this API.

2.3 Near-RT RIC Functions

Near-RT RIC is able to facilitate the creation and administration of E2 interface connection along with transport network layer according to these functional procedures. These features are used by the E2 Terminator to carry out a number of E2 operations, including E2 Setup Request, E2 Setup Response, E2 Removal, and connection status. Figure 2.2 represents the flows of communication between E2 node and Near-RT RIC. The E2AP specifications [15] and the E2 Setup procedure, for instance, describe the workings of the other previously mentioned services. The E2 Setup procedure must be done between Near-RT RIC and E2 Node to ensure the functionality, and other support procedures to guarantee the interface's scalable and resilient operation over a longer duration of time.

2.3.1 Near-RT RIC Support Services

In order to optimize RAN functionality, the E2 node must be monitored and controlled by xApps. Near-RT RIC services enable xApps on E2 Nodes to execute configurable service logic using global procedures. The RIC Services specified by the E2AP specifications are available for xApps to register to within the E2 Node.

REPORT Service: Enables E2 Node to forward data to Near-RT RIC with information from the E2 Node. Since the Near-RT RIC does not respond, this service uses asynchronous

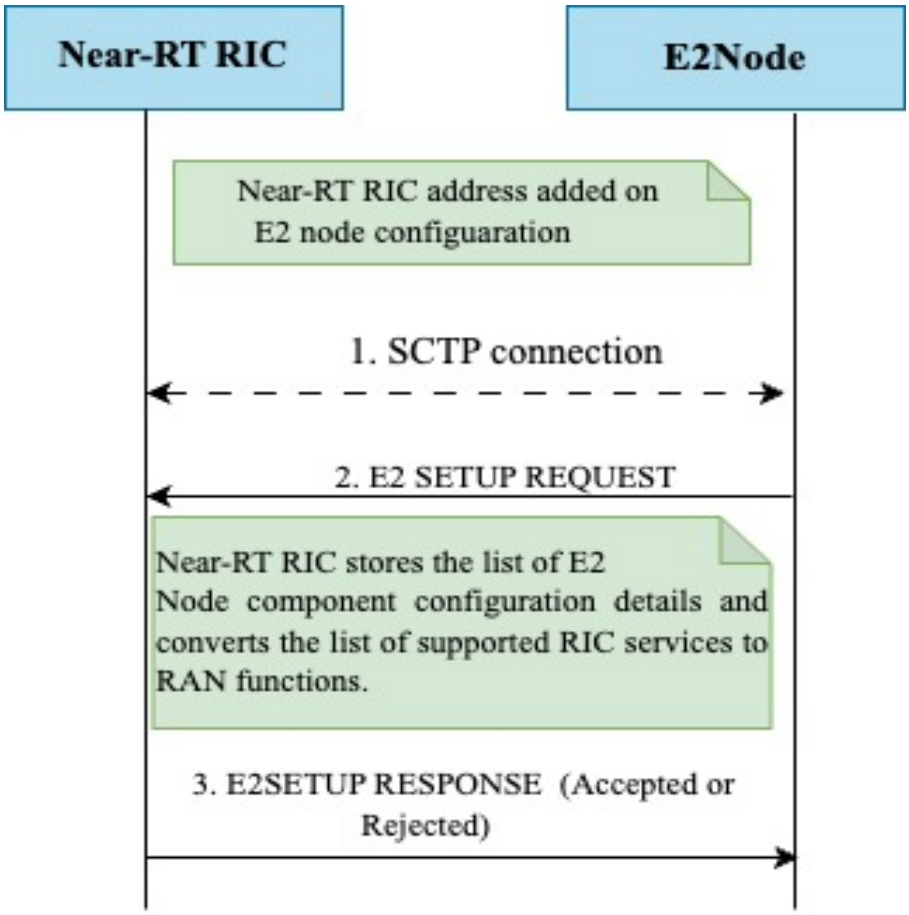

Figure 2.2: E2Setup Process in Near-RT RIC and E2 node

messaging.

INSERT Service: Permits the E2 Node to request that Near-RT RIC command assistance and pause signal processing for a while. Synchronous messaging is used in this service, and when a Time to Wait timer expires, a predetermined subsequent action is automatically carried out.

CONTROL Service: Support Near-RT RIC to initiate a related procedure in E2 Node. This service uses simultaneous messaging while Near-RT RIC demands acknowledgment of the E2 Node's control confirmation.

POLICY Service: Ensure the E2 Node to perform a particular decision dynamically when the event trigger occurs.

Before an xApp can use a RIC Service, the Near-RT RIC must perform one or more E2AP procedures. In order to onboard a RIC Service on E2 node, the E2 manager follows a set of basic E2AP steps, as depicted in Figure 2.3 [14]. The listed below E2AP functions are defined in O-RAN by WG3 [15].

The RIC Subscription creates a subscription for both the E2 Node and the xApp.

Subscription Delete terminates an active E2 Node subscription.

RIC Indication: According to the subscription, the E2 Node sends reports to Near-RT RIC.

A specific E2 Node function can be started or continued with the help of RIC Control.

The Near-RT RIC and RAN node must exchange a series of distinct E2AP messages in order to complete a RIC procedure. For instance, a RIC SUBSCRIPTION REQUEST message is transmitted to the E2 Node via the E2 interface to perform the RIC Subscription procedure showed in Figure 2.3. A contract-like payload in the message notifies the RIC service that the xApp has requested. It carries Information Element (IE)s that authorize the entity, such as

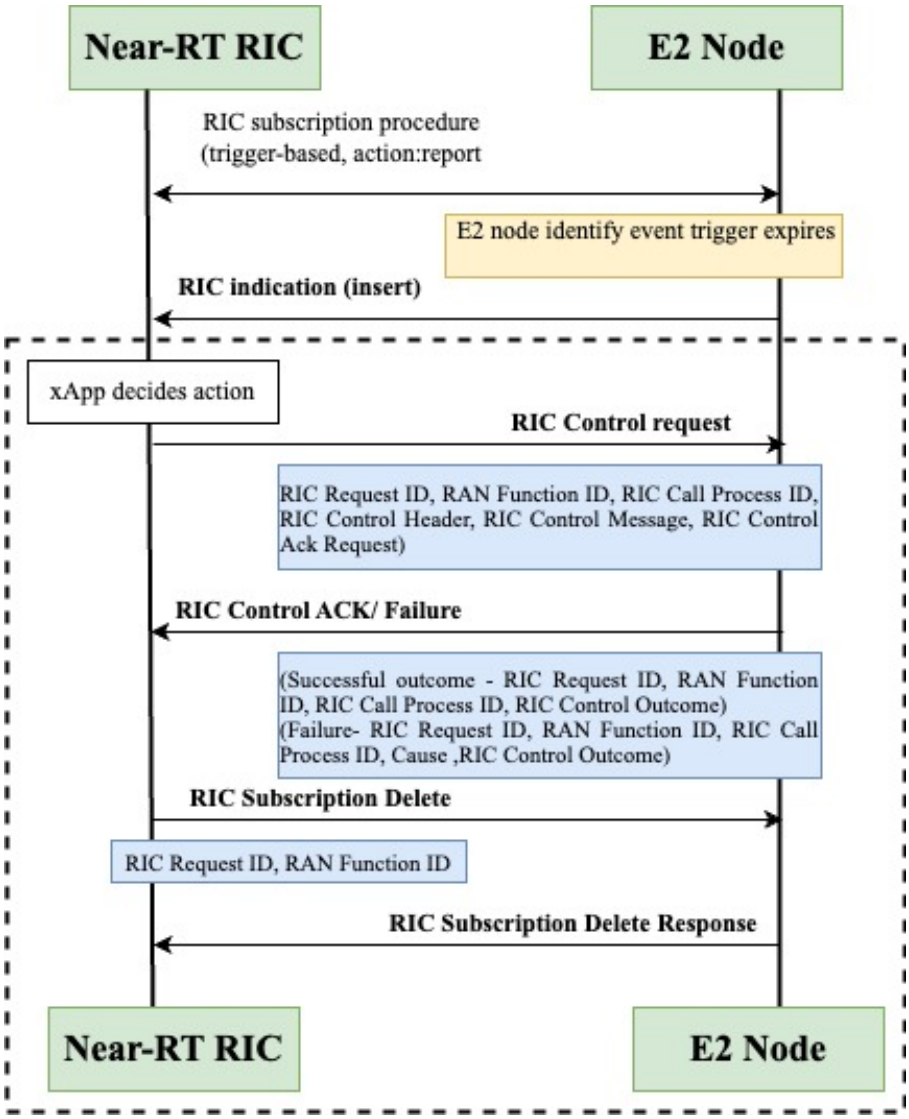

Figure 2.3: RIC Services, Procedure in Near-RT RIC with E2 node

an xApp asking RIC Service and associated RAN function ID on the E2 node that requires subscription. The E2 Node replies with a RIC SUBSCRIPTION RESPONSE message after attempting to provision its sub-components to satisfy the subscription request's requirements. A RIC SUBSCRIPTION FAILURE response is returned to Near-RT RIC in that the subscription cannot be completed. Based on application, an E2 Service Model (E2SM) can be created by combining one or more RIC Services. This model can optimize and lead RAN action toward a target scenario or state [15]. As we will discuss in Chapter 4, xApps use E2SM to handle RAN-specific use cases.

2.4 xApp Framework Operation

An xApp is technically an entity that carries out a specific policy. An xApp requires an xApp descriptor Java Script Object Notation (JSON) that outlines its configuration parameters in order for it to be deployable. Furthermore, the RIC platform must be configured for the xApp. A JSON schema for the descriptor also has to be supplied by the xApp writer. The complete functionality of xApp is shown in Figure 2.3 that involve the following steps:

1. xApp Development: xApp development involves creating applications for the RIC platform using RIC utility libraries. For these applications to be deployed, a valid Docker image need to be created, stored in an accessible Docker registry, and an appropriate xApp descriptor must be supplied. The xApp descriptor, which is structured in JSON, is crucial for managing the xApp life-cycle on the RIC platform. It includes essential configuration details such as data flow definitions for northbound and southbound traffic, parameters for Helm chart generation, and optional custom parameters specific to the xApp's functionality. To ensure compatibility and correctness, the descriptor must pass validation against the xApp JSON schema. This descriptor plays a critical role in defining how the xApp integrates with the RIC platform and operates effectively.

2. Registration : An xApp must first register with the xApp Manager after being deployed within the Near-RT RIC. Sharing a configuration file containing information about the xApp, along with its name, type of message, and version it plans to forward inside RIC platform, is part of registration process. To allow xApp to communicate with the remaining sub-components, xApp Manager verifies the authentication process and initiates a routing table notification. Following a successful registration, the xApp carries out service discovery, which finds various Near-RT RIC services with which it has authorization to communicate. The RMR messaging infrastructure is then used by the xApp to establish communications. It then sends a query to the SDL to see if any E2 Nodes are connected. When a target is found that matches, the xApp notifies the Subscription Manager of the subscription request along with an installation request for the RIC Service. After verifying the subscription request, the Subscription Manager gives it a new subscription ID. After that, it sends the request to the E2 Terminator, which uses the E2 interface to send it to the E2 Node.

3. Reconfiguration and Management: The xApp starts exploring service databases to find the Near-RT RIC services it is permitted to communicate with after successfully registering. After that, it connects using the RMR messaging framework and asks the SDL for a list of connected E2 Nodes. When the xApp finds a good target, it submits a subscription request, along with a request to install the RIC Service, to the Subscription Manager. After verifying the request and assigning a distinct subscription ID, the Subscription Manager sends it to E2 Terminator, which uses E2 interface to forward it to the network. Subscription Manager links the subscription ID to the requesting xApp and instructs the Routing Manager to create modified routing policies upon receiving subscription response along the return path.

4. De-registration: After gracefully ending the microservice, the xApp Manager manually deactivate the xApp. The xApp deletes all of its subscriptions with the E2 Nodes through the Subscription Delete Procedure when xApp Manager initiates de-registration for an active xApp. The xApp Manager then asks the Routing Manager to configure the routing table to

eliminate all links directing to the deactivated xApp.

The following chapters expand the topics covered in this chapter to implement application involving the deployment of numerous xApps for concurrent RAN control and to identify conflicts in E2 node.

Chapter 3

Problem Statement

This chapter presents the research problem statement, with an emphasis on the difficulties in implementing CM in O-RAN. The chapter starts by outlining the types of conflict that arise when several xApps operate simultaneously within the RIC. These problems frequently result in conflicting control decisions, poor network performance, and damaged QoS. Moreover, most prior conflict mitigation strategies are limited by being validated only in simulations, lacking reliability in real-world deployments with hardware, RF, and latency challenges. This chapter sets the foundation for the subsequent investigation into the implementation and effectiveness of CM in testbeds and real-world O-RAN environments, aiming to bridge the gap between theoretical CM strategies and their practical application.

3.0.1 xApp Driven RAN Control

With xApps installed in Near-RT RIC, intelligent RAN control is carried out in O-RAN's near real-time control loop. Specific network optimization tasks, such as traffic steering, energy efficiency, slice control, anomaly detection, and other duties, are carried out by each xApp. Mobile Network Operators (MNO)can choose xApps within a variety of external providers or create those in-house to make sure network performance is tailored to their unique operational needs. The integration of Slice Management (SM) and Traffic Steering (TS) is an illustration

of RANs optimization powered by multiple xApps. A TS xApp, for example, might route the an SM xApp sets up resource allocation for slices to guarantee sufficient QoS is preserved for preferred services, while traffic in the RAN is designed to keep equal load throughout cells. To prevent disputes between TS and SM control decisions, cooperation between the two xApps is necessary.

3.0.2 Types of Conflict

The RAN architecture with multiple network control agents is made possible by O-RAN's RICs. Conflicts between xApps' reconfiguration choices are likely to occur because different third-party vendors may co-exist with xApps. These disputes have the potential to seriously impair network performance, resulting in erratic behavior and a decline in confidence in the network service. The following classification of control conflicts is provided by O-RAN specification [12].

1. **Implicit Conflict**: Implicit conflicts occur when xApps have conflicting assumptions or dependencies on shared resources. For instance, if one xApp assumes that a certain resource will always be available, while another xApp depends on that same resource being released at a certain time, this could result in an implicit conflict.

2. **Indirect Conflict**: Indirect conflicts occurs with control decisions influencing the same network area. Indirect conflict cannot be observed immediately. Indirect conflicts can occur when xApps are not aware of each other's actions or do not have a way to coordinate their activities. This can lead to inconsistencies, errors, and inefficiencies in the RAN.

3. **Direct Conflict**: When two or more xApps wish to modify the same control parameter, a direct conflict arises.. Two or more xApps request different settings for the very same configuration of one or more parameters of a Control Target. The new request from an xApp may conflict with the running configuration resulting from a previous request

of another or the same xApp. The total requested resources from different xApps may exceed the limitation of the RAN system.

3.0.3 Problem

In the proposed system model, the O-RAN testbed architecture consists of two network slices (Slice A and Slice B) managed by distinct xApps. Each xApp is responsible for optimizing the allocation of PRBs to its respective slice while serving different user groups: Slice A prioritizes high-priority UEs, and Slice B handles regular traffic. This allocation process aims to enhance service quality and ensure efficient resource use across the slices.

While prior research has proposed conflict detection and mitigation strategies for such scenarios, the majority of these solutions are developed and validated only in simulated environments. This limits their applicability and reliability in real-world deployments, where hardware variability, Radio Frequency (RF) conditions, and control latencies introduce additional complexity.

To address this gap, this thesis focuses on enabling a more realistic and practical conflict management study by implementing and evaluating xApp conflict scenarios on a real-world testbed.

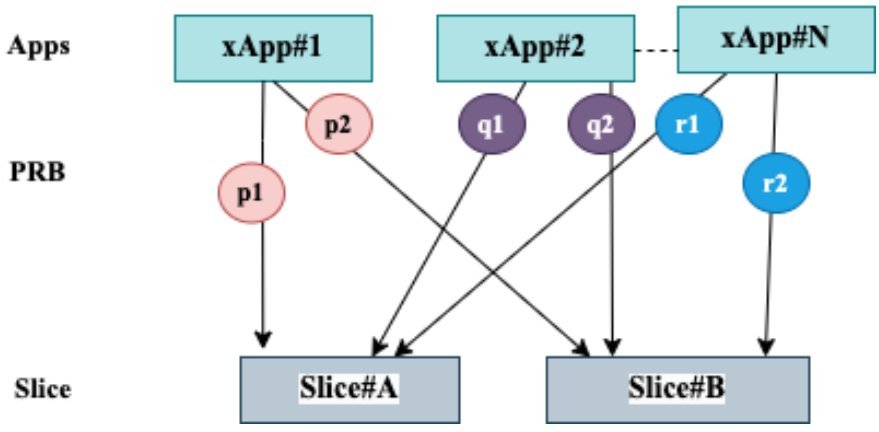

Figure 3.1: Direct Conflict Scenario

A direct conflict occurs when xApps attempt to adjust the PRB allocations for the same resource targets (i.e., slice-level PRBs) shown in Figure 3.1. For instance, if xApp #1, responsible for optimizing service for high-priority UEs in Slice A, decides to increase its allocation of PRBs to maximize throughput for these users, and xApp #2 concurrently tries to reallocate some of these same PRBs to enhance performance for regular UEs in Slice B, a direct conflict arises. This conflict not only results in potential instability and unpredictable network performance but also risks degrading the overall QoS for both slices. Such conflicts necessitate a robust Conflict Mitigation to detect the inconsistencies between the xApps' control decisions and to implement a resolution strategy that optimizes the needs of high-priority traffic while ensuring effective resource management across the network.

This work focuses on a specific class of conflicts within the Near-RT RIC environment—namely, direct conflict over PRB allocation by concurrently running xApps. While general conflict detection and mitigation in RIC can take many forms (e.g., indirect conflicts, implicit conflict), this study confines its scope to overlapping PRB control commands that lead to measurable performance degradation.

3.0.4 Case Study Scenario

In RAN, multiple xApps within the Near-RT RIC are responsible for managing PRB allocation across various network slices, each catering to a specific group of UEs with distinct performance requirements. Conflicts can occur when different xApps implement contrasting resource allocation strategies simultaneously. For instance, one xApp may prioritize a particular slice based on the number of connected UEs, allocating PRBs using a weighted combination of UE ratio and average PRB ratio to favor that slice—potentially at the expense of others. Meanwhile, another xApp might adopt a fairness-based approach, distributing PRBs equally across all active slices without consideration for individual slice demands.

Conflict Scenario

When two xApps operate simultaneously: xApp #1 seeks to allocate a larger PRB share to its prioritized slice, disrupting xApp #2's fairness-driven approach. xApp #2 enforces equal or proportional distribution, potentially limiting PRBs for the slice prioritized by xApp #1.

This leads to: Instability in network performance, with frequent PRB adjustments causing fluctuations in slice throughput and UE experience. Figure 3.2 represents the network metrics in a Grafana dashboard [16]. It is showing the number of cell, connected UEs, and downlink throughput of the network. When there are just two UEs connected, the network operates in a generally stable manner at first. After the first minute, when another UE joins in slice A, notable variations in UE-level DL throughput are seen for every UE. In particular, UE #2's DL throughput fluctuates between roughly 10 and 28 MB/s. Erratic QoS in user devices is indicated by similar disruptions seen for other UEs. The conflict between xApps that routinely carry out conflicting control decisions is the cause of unmitigated conflict in the network.

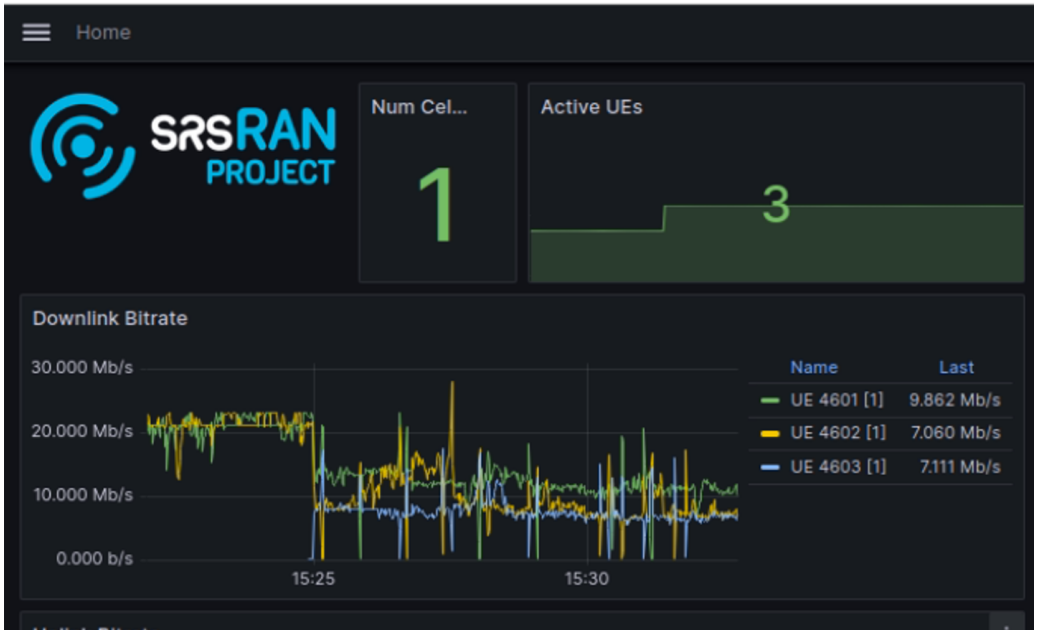

Figure 3.2: Network Fluctuations in Direct Conflict Scenario

Suboptimal PRB utilization, as the conflicting allocation strategies prevent efficient distri-

bution, negatively impacting overall network efficiency. These conflicting approaches create inefficiencies, highlighting the need for a coordination mechanism to harmonize PRB allocation between xApps.

The problem involves resolving xApps conflicts with different PRB allocation strategies in a RAN slicing scenario. The goal is to ensure stable, fair, and efficient PRB allocation while minimizing deviations from target allocations and maximizing network performance. The solution requires conflict detection, mitigation, and optimization techniques that are discussed in the following chapter.

Chapter 4

Methodology

This chapter presents the system model utilized to enable and implement CM scheme in an O-RAN-compliant environment (CCI xG Testbed) (Contribution 1). The Near-RT RIC deployment is described, which involves xApps with different PRB allocation strategies, resulting in a direct conflict in resource allocation. The process for detecting and resolving conflicts is also described, emphasizing the roles of the CCA in detecting and resolving conflicts (Contribution 2). The specifics of these components' implementation and how they interact within the testbed will be covered in detail in the sections that follow.

4.1 Proposed System Model

This work's primary objective is to use actual hardware and software to experimentally validate the considered direct CM scheme. The testbed used for the experimental setup is composed of O-RAN-compliant components.

Figure 4.1 displays the deployment's system diagram. The configuration of RAN setup made of a gNodeB (gNB) software from the Software Radio Systems RAN (srsRAN) project [17]. Two network slices, A and B, are hosted by the gNB and are both responsible for managing eMBB traffic. On the frequency 3600 MHz, a single cell provides access to both slices. Slice

A is designed to serve high-priority user equipment UEs, whereas Slice B manages traffic from standard, non-prioritized UEs.

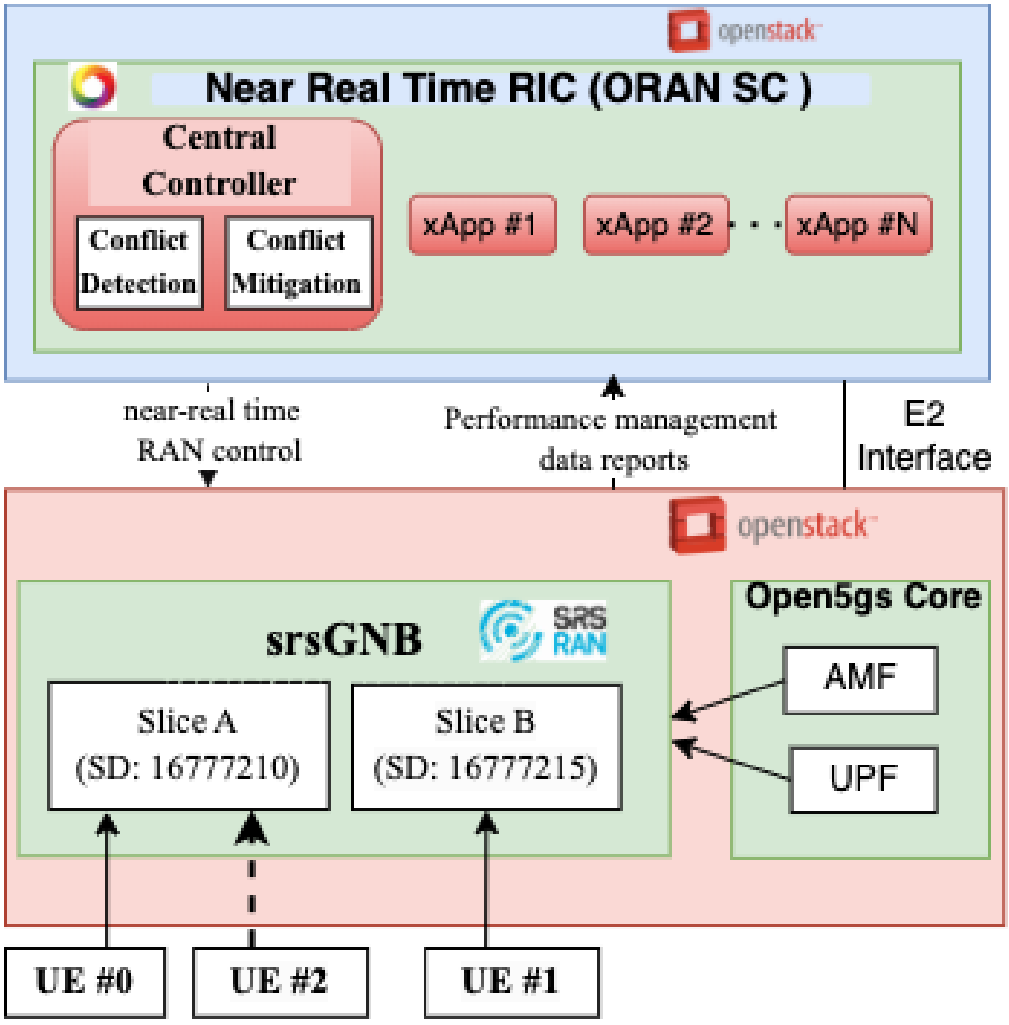

Figure 4.1: System diagram for the deployment

The open source O-RAN Software Community (O-RAN SC) Near-RT RIC software serves as the foundation for the deployed RIC (release I). It contains xApps that use action 6 from style 2 of CONTROL services in E2SM-RC [18] to optimize slice-level PRB allocations. Although all deployed xApps modify slice-level PRB allocations, their optimization goals differ. From an explanatory point of view, we use two xApps (#1 and #2) to describe our methodology. While the math formulation is generic and adapt to N numbers of xApps.

Emphasizing traffic from prioritized UEs connecting to slice A is the goal of xApp #1. In order to accomplish this, xApp #1 gives slice A a proportionately larger share of the available PRBs. gNB serves total U number of UEs, total slices in gNB is S , the available PRBs in the cell P, and the available UEs assigned in slice A within gNB U_A. To regulate the trade-off between strict prioritization and fair resource sharing across slices, we introduce a tunable fairness modulation parameter, denoted as β. This parameter controls the additional baseline PRB allocation that each slice receives, ensuring that even non-prioritized slices maintain a minimum level of resource availability. As shown in (4.1), xApp #1 determines how many PRBs $p_{1,A}$ are assigned to slice A, and how many are assigned to non-prioritized slices except A $p_{1,s}$ in accordance with (4.2). r_A serves as high priority PRB ratio and r_s serves the non-priority PRB ratio makes foundation for the computations.

The percentage of resource allotted depends on the user device connected to the emphasized slice is shown by r_A. Afterwards resources have been allocated to the designated slice, the remaining PRBs are split equally on other non-prioritized slices, as indicated by the ratio r_S. Generally speaking, xApp #1 distributes PRBs by averaging two factors: a variable inversely correlated to the amount of slices ($1/S$) and the proportion associated to the slice type (r_A or r_s) shown in equation 4.1 and 4.2. This guarantees that the logic strikes a balance between overall fairness and slice priority ratios.

$$r_A = \frac{U_A}{U} \qquad p_{1,A} = P \cdot (r_A + \frac{1}{S}) \cdot \frac{1}{\beta} \tag{4.1}$$

$$r_s = \frac{1 - r_A}{S - 1} \qquad p_{1,s} = P \cdot (r_s + \frac{1}{S}) \cdot \frac{1}{\beta} \tag{4.2}$$

xApp #2 seeks to prioritize a balanced distribution of PRBs across the network, whereas xApp #1 concentrates on slice A. Equation (4.3) is used to allocate PRBs for all slices p_2 that are

performed by xApp #2.

$$p_2 = P/S \tag{4.3}$$

The relevant xApp logic is triggered simultaneously and both xApps are operating concurrently in the network. The xApps create and transmit the appropriate E2 CONTROL messages to apply the determined allocations following each computation of the PRB allocations.

4.1.1 Proposed Conflict Detection Mechanism

In the considered test scenario, the control conflict arises directly within all deployed xApps. The slice-level PRB allocation is altered by both xApps, but their goals are distinct, resulting in a direct conflict. The conflict detection mechanism, involves a Central Controller Agent (CCA), deployed in Near-RT RIC monitoring all control decisions made by the xApps. It verifies whether any decisions target the same control parameter. If conflicting control decisions are identified, the CCA forwards the conflict information to the CM Agent for resolution. This process is showed in Figure 4.2

4.1.2 Proposed Conflict Mitigation Mechanism

In open RAN environments, conflicts in xApps lead to suboptimal performance, reduced network efficiency, and unfair resource utilization, ultimately degrading the QoS for users. Resolving this conflict is essential to ensure balanced resource allocation while meeting the diverse network requirements. To address this, we employ a Deep Q-Network (DQN) method Figure 4.3, which dynamically learns from conflict scenarios and optimizes PRB allocation by minimizing a loss function that quantifies resource contention. Unlike static rule-based conflict resolution, DQN continuously adapts to network conditions, making intelligent decisions that

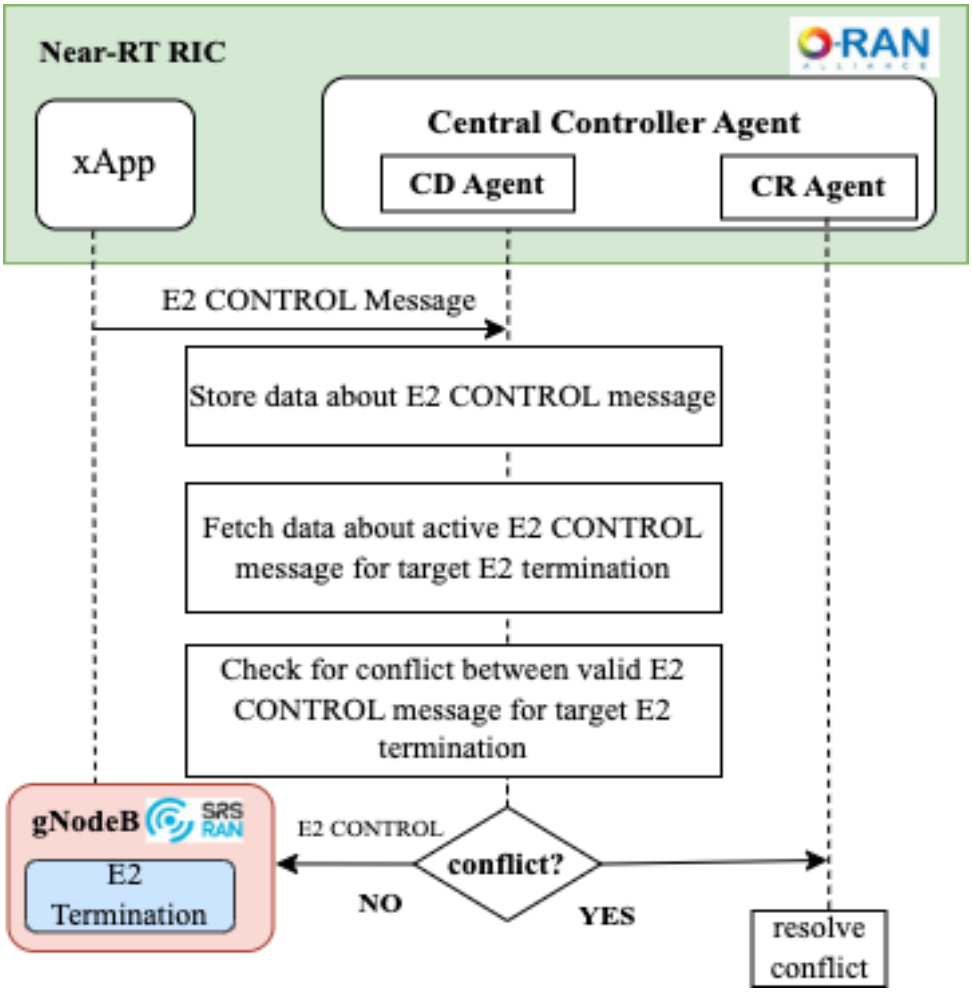

Figure 4.2: Conflict Detection Procedure

reduce interference and improve overall system efficiency. DQN dynamically adjusts allocation strategies by learning optimal actions that balance these conflicting objectives, avoiding the limitations of static rule-based approaches.

The choice of DQN stems from its ability to handle complex, high-dimensional decision spaces and generalize well across different conflict scenarios. Traditional methods often fail to adapt to rapid changes in network conditions. DQN employs an exploration-exploitation tradeoff to continuously refine resource allocation strategies, ensuring adaptability to real-time traffic variations. By storing and reusing previous allocation scenarios, DQN's experience replay mechanism improves decision-making over time.

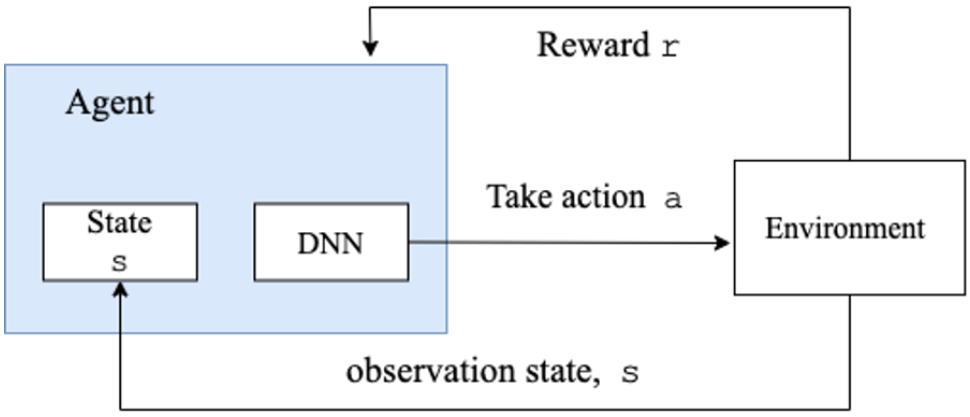

Figure 4.3: DQN Control Loop

This feature enhances the model's capacity to generalize to novel conflict scenarios and aids in managing unforeseen network conditions. The purpose of the reward function is to penalize over-allocation and encourage actions that align resource distribution with an optimal state shown in Figure 4.4. The optimal policy will feed into the network to allocate the resource accordingly. This results in an efficient and fair allocation policy without requiring manual intervention.

In the resolution mechanism we adopt a weighted priority approach using pre-defined alpha values to resolve resource contention. This method assumes prior knowledge of xApp importance or operator policy, and serves as a starting point for practical, policy-driven conflict mitigation. In future iterations, these weights can be learned or dynamically adapted based on performance feedback or operator-defined service level agreement. This approach does not yet generalize to all xApp conflict types or fully autonomous priority assignment, which remains an open challenge in conflict resolution systems.

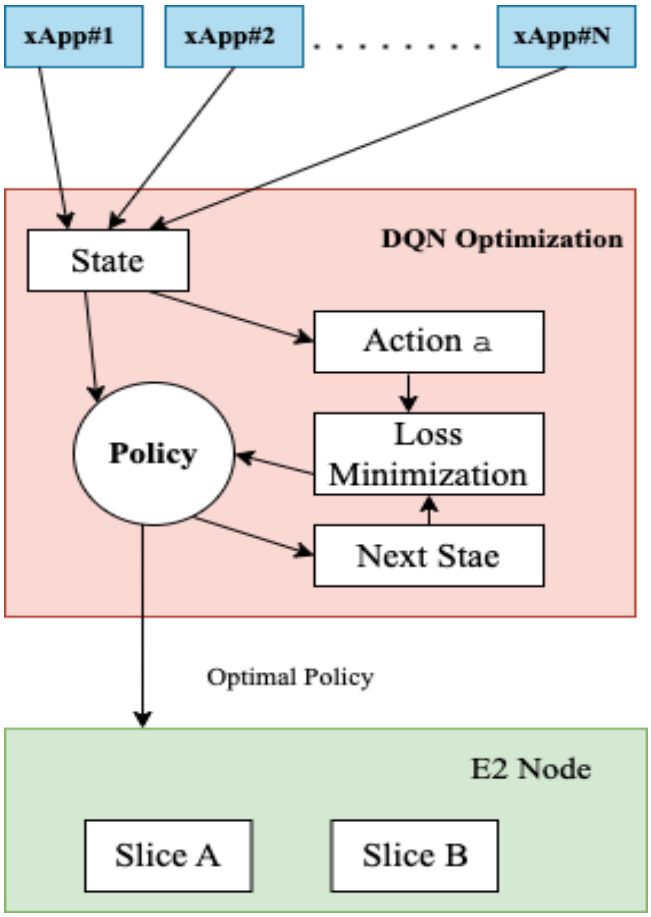

Figure 4.4: System Design For Proposed Methodology

4.2 Mathematical Formulation For Optimal Policy

Let S be the set of network slices, where each slice s has an associated xApp responsible for managing PRB allocation. The total available PRBs in the system is:

$$P_{\text{total}}$$

The problem is modeled as a Markov Decision Process (MDP), defined by:

- **State** (s): The state represents the current PRB allocation across all N xApps:

$$s = [P_{x1}, P_{x2}, ..., P_{xN}] \tag{4.4}$$

  Each xApp x_i has a target PRB allocation:

$$P_{xi} = [P_{i,A}, P_{i,s}] \tag{4.5}$$

  where $P_{i,A}$ and $P_{i,s}$ represent PRB allocations for different traffic types within a slice.

- **Action** (a_t): The PRB adjustment action proposed by the DQN agent. a_t represents the action taken by the DQN agent at time step t. The agent adjusts PRB allocation based on the learned policy. Actions represent incrementing or decrementing PRBs for each xApp:

$$a_t = \{\Delta p_1, \Delta p_2, ..., \Delta p_N\} \tag{4.6}$$

  where:

$$\Delta p_i \in \{0, P_{\text{total}}\} \tag{4.7}$$

- **Reward** (R): The reward is a numerical term that gives feedback to the agent about how good or bad its action was in a given state. The goal of the DQN is to maximize the cumulative reward over time by learning the best actions to take in each state. In this optimization technique the reward is calculated as negative of the calculated loss in Equation 4.10

- **Transition**: The next state is updated based on the action taken.

Due to different goal of xApps, the allocations may conflict, requiring a resolution mechanism. The goal of the DQN agent is to determine an optimal PRB allocation $P^* = [p_1^*, p_2^*, ..., p_N^*]$, while minimizing the deviation from xApp targets. The loss function capturing this deviation is defined as:

$$L = \sum_{i=1}^{N} \alpha_i ||P_{x1} - P_{xi}|| \tag{4.8}$$

where α is a weight factor balancing the importance of each xApp's priority.

A penalty term X is introduced to ensure that total PRBs allocated do not exceed P_{total}:

$$\text{Penalty} = -X \times |P_{\text{total}} - \sum_{i=1}^{N} p_i| \tag{4.9}$$

The magnitude of the penalty (X) determines how strongly the agent is discouraged from violating the constraint. A larger penalty would make the agent prioritize satisfying the constraint more aggressively. Based on the observance if the agent is frequently violating the constraint, we can increase the penalty to forces the DQN model to prioritize constraint satisfaction by strictly adhering to the total PRB limit. A lower coefficient (e.g., 1 or 2) would allow for more flexibility in allocation.

Thus, the reward function given by R is formulated as:

$$R = -L \tag{4.10}$$

where a higher reward corresponds to a PRB allocation that minimizes conflicts and ensures efficient resource distribution.

The optimization process is modeled using DQN, where the Q-function is approximated by a Neural Network (NN):

$$Q(s_t, a_t; \theta) = \max_a \mathbb{E}[R + \gamma \max_{a'} Q(s_{t+1}, a'; \theta)] \tag{4.11}$$

where:

- s_t represents the state of the environment at time step t.
- $Q(s_t, a_t; \theta)$ is the predicted Q-value for state s_t and action a_t.
- γ is the discount factor.
- θ are the NN weights.

4.3 DQN Training Process

4.3.1 Experience Replay

A memory buffer stores past transitions (s, a, r, s_t) to break correlation in training data and improve learning.

Table 4.1: DQN-Based PRB Allocation Algorithm Steps

Step	**Description**
1	Initialize replay buffer $\mathcal{D}$, Q-network, and target Q-network.
2	For episode = 1 to M:
2a	Initialize state s.
2b	For $t = 1$ to T:
2b(i)	Select action a using ϵ-greedy strategy.
2b(ii)	Execute a, observe reward r and next state s_t.
2b(iii)	Store transition $(s, a, r, s_t, \text{done})$ in $\mathcal{D}$.
2b(iv)	Sample a mini-batch from $\mathcal{D}$.
2b(v)	Compute target Q-values.
2b(vi)	Update Q-network using gradient descent.
2b(vii)	Update $s = s_t$.
2c	Decay ϵ.
3	Trained Q-network compute optimized PRB allocation.

4.3.2 Target Network

A separate target network is used to stabilize training, where Q-values are updated as:

$$Q(s_t, a_t) = R + \gamma \max_{a'} Q_{\text{target}}(s_{t+1}, a') \tag{4.12}$$

4.3.3 Exploration-Exploitation Tradeoff

The agent balances between:

- **Exploration**: Selecting random actions (ϵ-greedy approach).
- **Exploitation**: Choosing actions based on learned Q-values.

The exploration probability (ϵ) decays over time:

$$\epsilon = \max(\epsilon_{\min}, \epsilon \cdot \text{decay factor}) \tag{4.13}$$

4.4 Optimal Policy

The agent learns an optimal PRB allocation policy after training. The optimal distribution is calculated as:

$$P^* = [p_1^*, p_2^*, ..., p_N^*] \tag{4.14}$$

with a final reward of:

$$R_{\max} \tag{4.15}$$

4.5 Step-by-Step Breakdown of DQN Algorithm

The DQN algorithm follows a structured approach to learn optimal resource allocations while resolving conflicts between xApps. Table 4.1 includes the notations and the algorithm steps for the optimization. The training process involves interaction with the environment, experience replay, and iterative learning.

4.5.1 Initialize Components

- Set up the replay buffer D to store past experiences.
- Initialize the Q-network (used for decision-making) and the target Q-network (used for stable learning).

4.5.2 Training Over Multiple Episodes

- Repeat the learning process for M episodes to ensure proper training.

4.5.3 Within Each Episode

1. Start in an initial state (e.g., current PRB allocations).

2. For each time step t, perform the following actions:

   (a) **Select an Action**

   - Use an ϵ-greedy strategy:
     - With probability ϵ, pick a random action (exploration).
     - Otherwise, pick the action with the highest Q-value (exploitation).

   (b) **Execute the Action**

   - Apply the selected PRB allocation and observe:
     - Reward r (how good/bad the action was).
     - Next state s_t (updated network condition).

   (c) **Store Experience**

   - Save the transition $(s, a, r, s_t, \text{done})$ in the replay buffer D.

   (d) **Sample Mini-Batch from Replay Buffer**

   - Randomly sample past experiences to break correlation in training data.

   (e) **Compute Target Q-Values**

   - Use the Bellman equation to update expected future rewards:

$$Q_{\text{target}} = R + \gamma \max Q_{\text{target}}(s_t, a')$$

   (f) **Update the Q-Network**

   - Use gradient descent to reduce the deviation between predicted and target Q-values.

(g) **Update the State**

- Move to the next state s_t.

4.5.4 Reduce Exploration Rate ϵ Gradually

- This shifts the model from exploration (trying random actions) to exploitation (selecting the best action).

Once training is complete, the learned Q-network generates optimized PRB allocations that minimize deviation and improve performance.

This process ensures the agent learns optimal PRB allocations, adapting to network conditions dynamically.

Chapter 5

Experimentation and Evaluation

This chapter outlines the considered scenario of the experimental setup and includes the metrics used to evaluate the performance of the DQN-based weighted priority CM model. To validate **Contribution 1**, the entire conflict detection and mitigation workflow is deployed and tested in an O-RAN-compliant and SDR-based testbed. In addition to the proposed approach, we also implement and evaluate two O-RAN Alliance-specified conflict mitigation mechanisms—time-based and priority-based—within the same testbed environment. This enables a comparative performance analysis, validating **Contribution 3** and demonstrating the practical applicability and benefits of the proposed method.

5.0.1 Experimental Setup

The system configuration is implemented in CCI xG Testbed [19]. On the CCI xG Testbed Figure 5.1, Near-RT RIC from O-RAN SC is installed in an Open-Stack Virtual Machine (VM) with Ubuntu 20.04 OS, 8 GB of RAM, 8 CPU, and 512 GB of storage. With the

required interfaces to communicate with RAN node, the Near-RT RIC is installed as a Docker container. Additionally, the xApps are running within the RIC platform as a Docker containers, with the required service model and all associated sub-components compatible with the RAN environment.

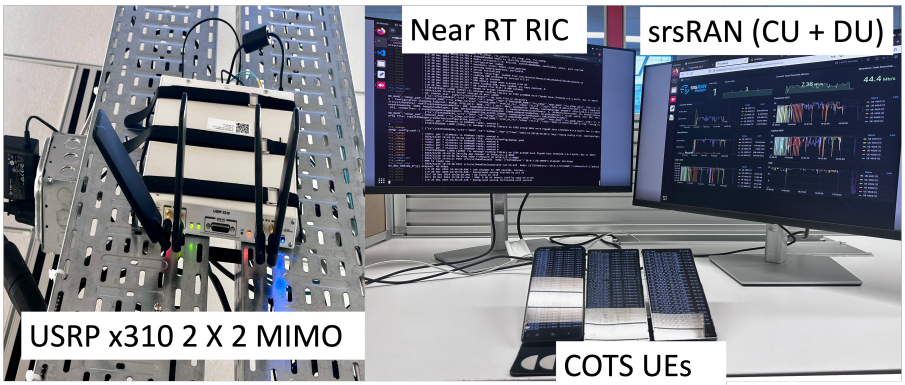

Figure 5.1: Experimental Setup

Together with Open5GS Core, the srsRAN 5G gNodeB is set up in an additional Open-Stack VM. A Universal Software Radio Peripheral (USRP) x310 (20 MHz bandwidth) in the Radio Ceiling Grid (an indoor component of the CCI xG Testbed) serves as the RF front-end for the RAN's RF function. It is a 2 Tx and 2 Rx antenna that can operate as a 2x2 Multiple Input Multiple Output (MIMO) system. It connects to the VM via a 10G Ethernet cable. Three Samsung phones equipped with programmable SIM cards served as the UEs. Figure 5.1 displays the implementation's experimental setup.

Figure 5.2 shows that the DQN model successfully converged during training conducted involving three UEs and two concurrently operating xApps with differing optimization goals. One xApp prioritized resource allocation to a specific slice based on UE distribution, while the other aimed for a fair or proportional distribution of resource across slices. This setup introduced real-time conflict in PRB allocation decisions, creating a dynamic and challenging environment for the DQN agent to learn from. The total reward steadily increased in the early episodes, indicating that the agent was learning effective strategies to resolve these conflicts and opti-

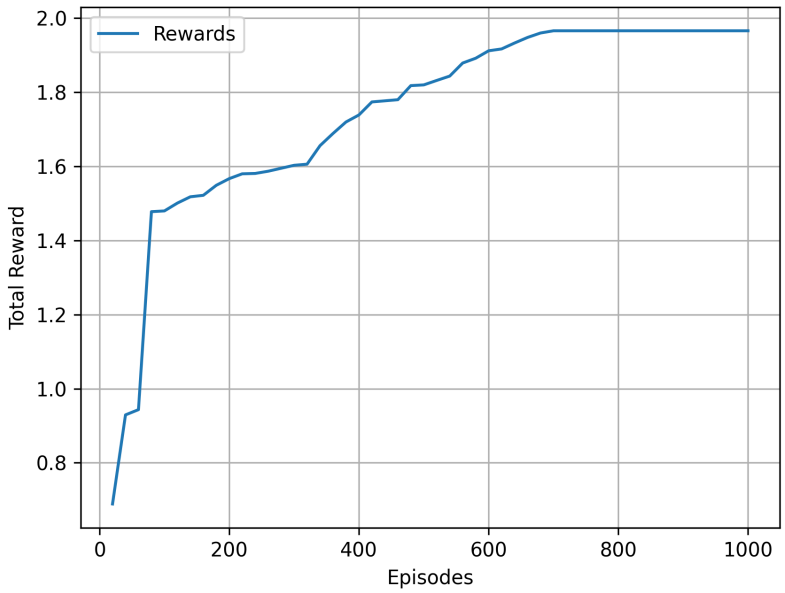

Figure 5.2: DQN Convergence Plot

mize performance. Around episode 800, the reward curve stabilized, suggesting convergence to a near-optimal policy. This demonstrates the model's capability to adapt to conflicting xApp behaviors and deliver efficient PRB allocations in complex, multi-UE scenarios typical in systems.

Table 5.1 includes the details of DQN parameters. The model is designed to maximize the reward while minimizing the deviation of DL throughput. Later, the trained model integrated into the network to evaluate its performance that is described in the following chapter.

5.0.2 Evaluation Methodology

The scenario for evaluating the efficiency of implemented mechanism measures includes two phases. Initially, one UE is connected to each of slices A and B, and both slices are enabled. One minitue later when another UE joins slice A, the resource allocations logic in xApps are altered as a result. A direct control conflict results from this situation because of the distinct

Table 5.1: Hyperparameter used for DQN training

Parameter	**Value**
Number of Hidden Layers	2
Neurons per Hidden Layer	32
Memory Size	10,000
Discount Factor (γ)	0.98
Initial Epsilon (ϵ)	1.0
Minimum Epsilon	0.01
Epsilon Decay	0.999
Learning Rate	0.001
Target Update Rate (τ)	0.01
Batch Size	1024
Total Episodes	1000
Maximum Steps per Episode	100
Activation Function	ReLU
Optimizer	Adam

resource allocation logic used by the xApps. In any event, UE-level DL throughput is tracked and recorded to allow for the assessment of network performance while the experiment is being conducted.

5.0.3 Experimental Scenarios

In order to assess the efficacy of conflict resolution techniques in a an O-RAN-compliant environment, the following four scenarios were defined and analyzed:

1. **Baseline (No Conflict) Scenario**: In this scenario, only a single xApp is active and responsible for resource allocation decisions across slices. No conflicting control actions are issued, ensuring stable and predictable resource allocation behavior. This serves as a reference point to measure the impact of conflicting actions in subsequent scenarios.

2. **With Conflict (No Resolution) Scenario**: Multiple xApps operate concurrently with overlapping control domains, issuing independent PRB allocation decisions. No conflict mitigation is applied, allowing the full effect of resource contention and inconsistency in RAN control

to be observed. This scenario highlights the need for conflict resolution mechanisms.

Figure 5.3 shows the downlink throughput in case of conflict. Initially the network throughput is stable when there is only two UE connected. After one minute when another UE joins in slice A, the network starts fluctuating due to the overlapping of xApps decisions on each other. This fluctuation leads to unstable QoS in the connected device as xApps are deviating from their goal as a result of conflicting decisions in the network. This scenario highlights the need for conflict resolution mechanisms to ensure a stable network.

3. **Time-Based Resolution Scenario**: Conflicts are resolved using a time-based mechanism. The first xApp to issue a control command is granted control, while subsequent conflicting requests are delayed for a time window. This approach follows a first-come, first-served principle and helps mitigate conflicts based on a timely manner. Figure 5.4 shows the downlink throughput when time-based resolution approach is enabled. In this method the CCA check which xApps decision arrives first in the network. Then it execute the first xApp and hold the other xApps for a certain period of time. After that time period, the CCA again execute the next xApp in the queue and hold the other xApps. The Figure 5.4 represents two xApps decisions are being executed within 10 s time window (a tunable parameter). This method [2] ensures orderly execution of xApp logic within non-overlapping time windows, thereby avoiding simultaneous control and enabling fair resource governance. The decision of which xApp runs in a given window follows a periodic alternation strategy. This ensures that all participating xApps receive equal opportunities to execute their logic over time, while avoiding contention. While this method guarantees non-overlapping execution, it does not consider the urgency or context of each xApp's control decisions. For instance, if an xApp needs to react to a sudden spike in traffic but is inactive during its time slot, latency may be introduced, which can be seen in Table 5.2.

4. **Priority-Based Resolution Scenario**: In this scenario, xApps are assigned fixed priority

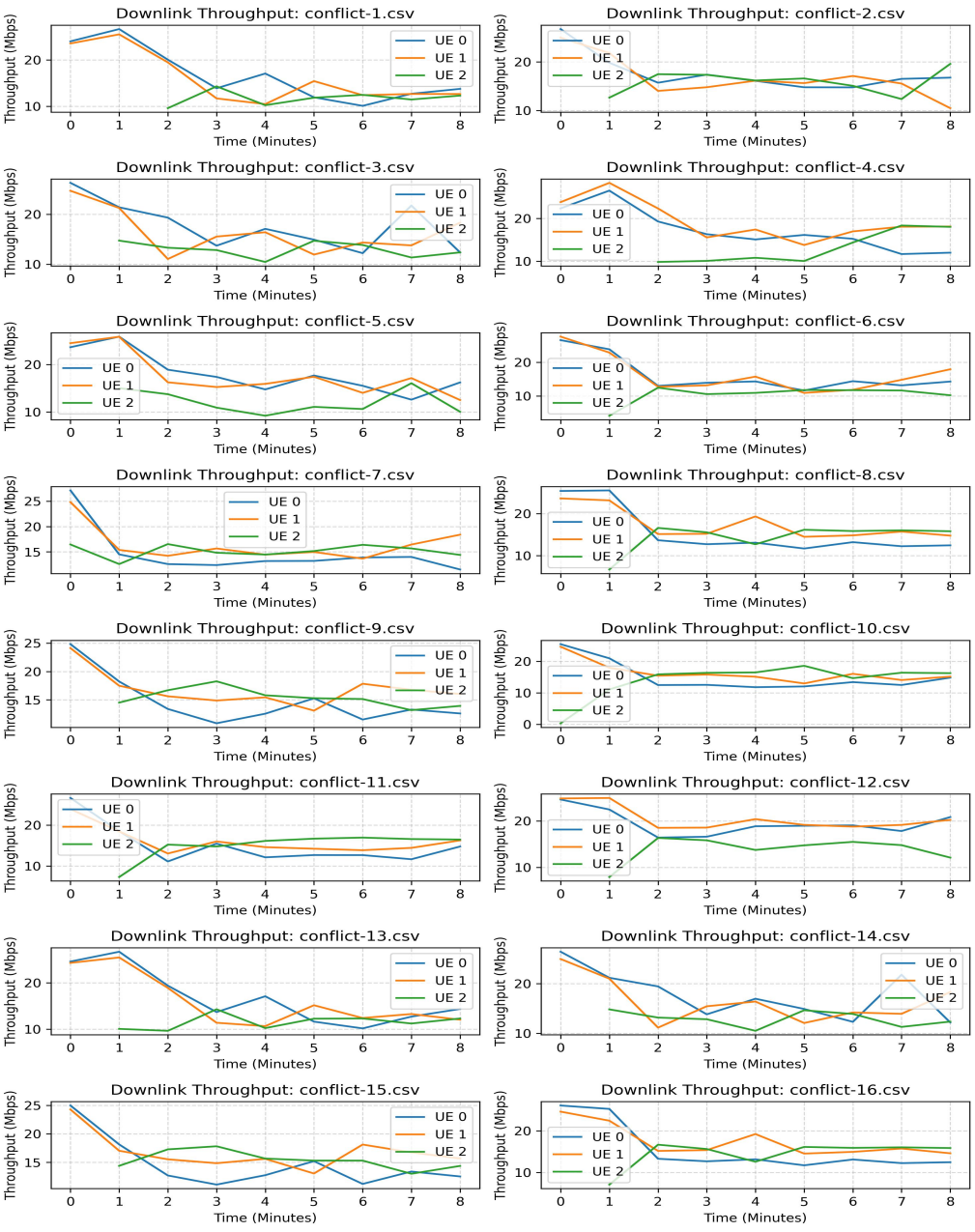

Figure 5.3: DL Throughput measurements in conflict scenario without Conflict Mitigation

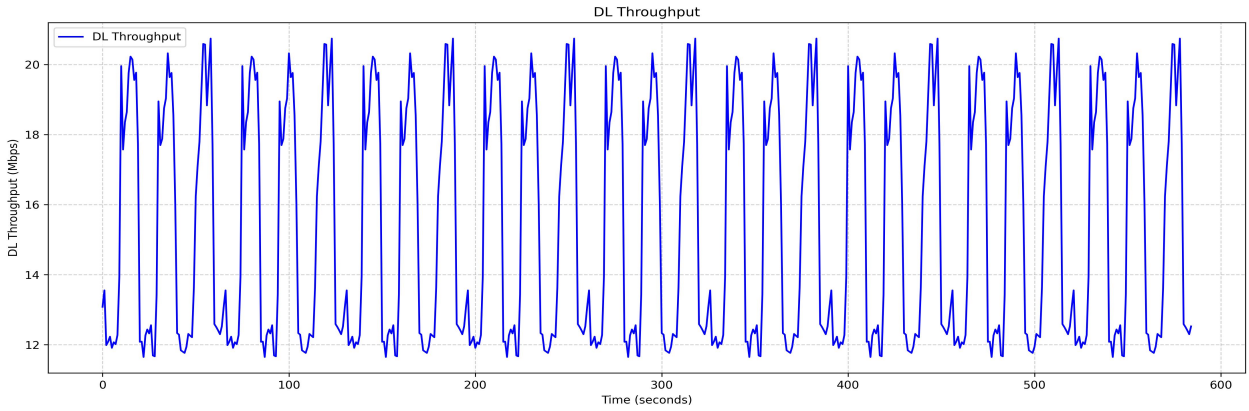

Figure 5.4: DL Throughput measurements in Time based resolution scenario

levels. When conflicts occur, the system allows control actions from higher-priority xApps to take precedence, discarding or overriding commands from lower-priority xApps. This approach ensures that critical xApps maintain influence over RAN behavior during contention.

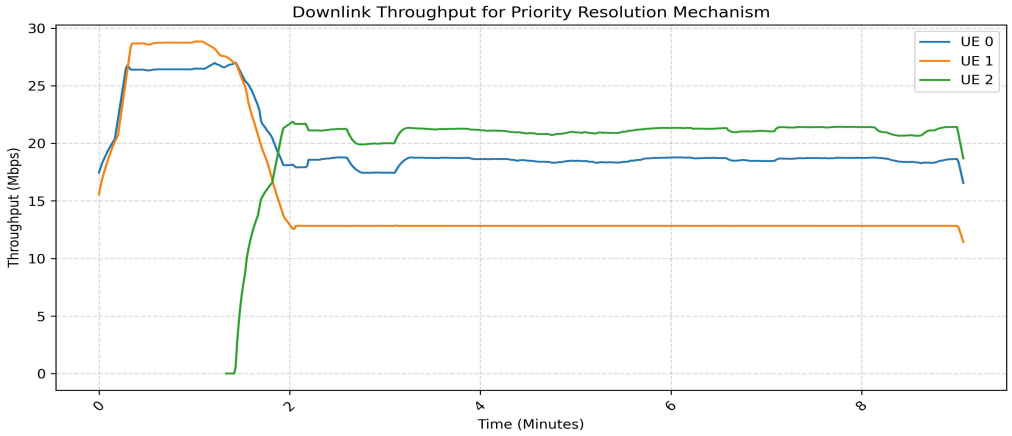

Figure 5.5: DL Throughput measurements in Priority based resolution scenario

Priority based resolution mechanism prioritizes [3] those that are deemed more critical for maintaining overall network stability and quality of service. For example, a higher priority might be given to xApps servicing high-priority traffic, ensuring that their control decisions are executed. Figure 5.5 shows the down link bit rate measurements when priority resolution is enabled. This method involves only executing one xApp and discarding other existing xApps's

decisions to emphasize the high priority user device. In this method the DL bitrate is seen stable and no fluctuation or overlapping is observed across the user devices. Though this method performs well over time based resolution but the latency is still very high compared to RL mechanism.

5. **Proposed DQN-Based Resolution Scenario**: This scenario implements a DQN-based learning model as the conflict mitigation strategy. The model learns from historical xApp behavior and real-time network metrics to dynamically resolve resource allocation conflicts. It optimizes resource distribution by minimizing a loss function that captures both fairness and performance objectives, enabling adaptive decision-making beyond static rules.

In the run with proposed weighted priority resolution mechanism enabled integrating the trained DQN model to the network, the DL throughput adapts the optimized PRB allocation by the trained model. Figure 5.6 shows the downlink throughput when the DQN resolution is enabled. Initially the the bit rate is stable when there is only two UEs in the network. After connecting the third UE to the slice A, the network adapts the optimized allocation by the trained model. It can be seen that the the throughput is stable for each device in the network. Compared to the with conflict scenario, it is clear that the proposed mechanism performs well and reduce the standard deviation of DL throughput and improve the latency which is reflected in Table 5.2.

5.1 Performance Metrics

To assess the effectiveness of the implemented solution, several metrics (downlink throughput, uplink throughput, average latency and standard deviation) were collected as outlined in Table 5.2 with no conflict, with conflict, with time based resolution, with priority based resolution and finally DQN based resolution enabled. The testbed setup remained consistent across all runs. Data collection for each experiment was one minute prior to the third UE connected to

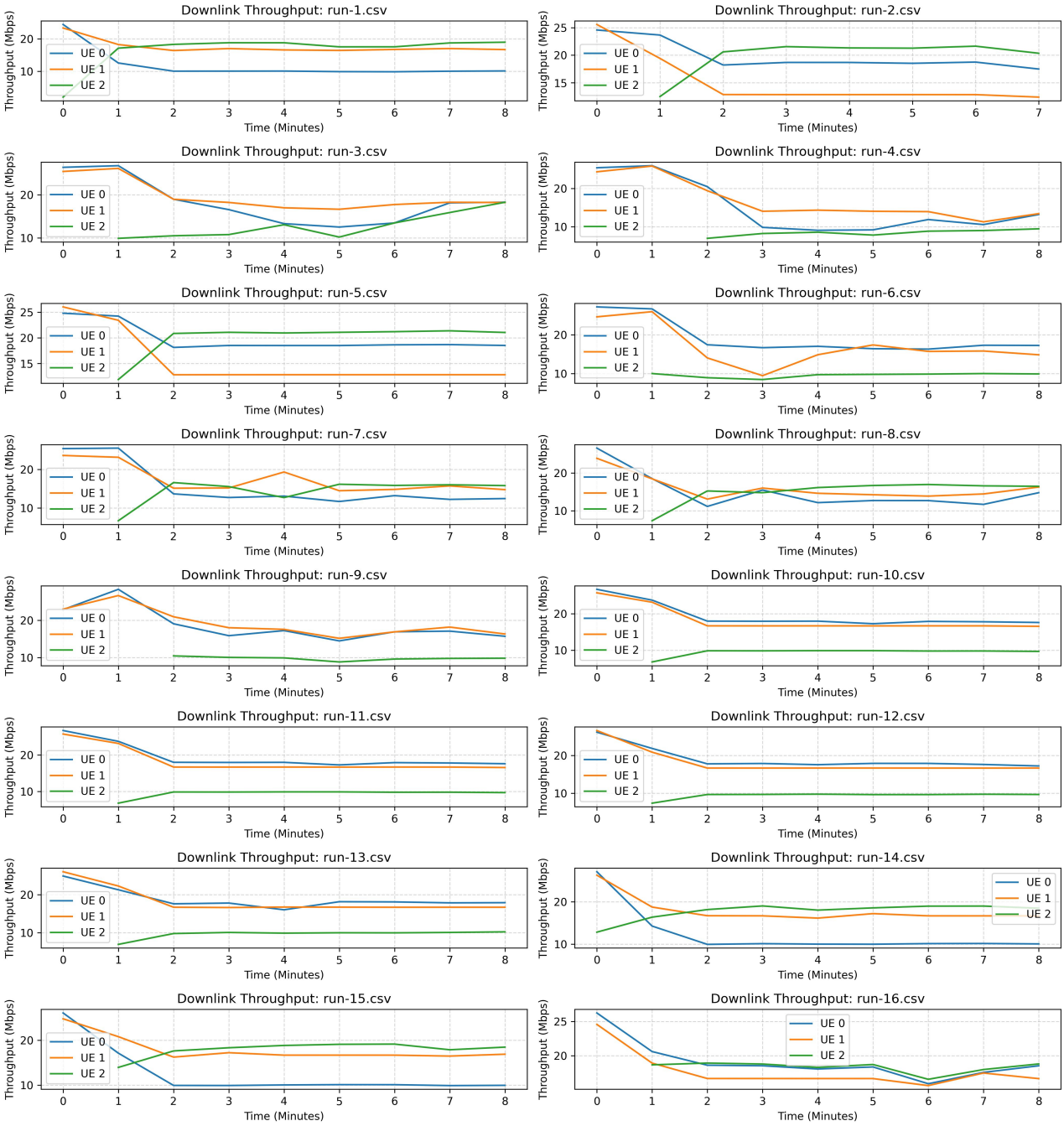

Figure 5.6: DL Throughput measurements in Proposed DQN based resolution scenario

slice A and extended around six minutes afterward. For each experiment, the DL throughput of each UE and Uplink (UL) throughput was recorded and plot over time. The performance metrics used in Table 5.2 are defined as follows:

5.1.1 Downlink (DL) and Uplink (UL) Bitrate

The average bitrate is a key performance indicator that reflects the overall data transmission rate over a network during a specific measurement period. This equation computes the mean value of the transmission rate by averaging the instantaneous bitrates over time. It provides a holistic view of how efficiently data is being transferred in the downlink (from base station to user equipment) and uplink (from user equipment to base station) directions. A higher average bitrate typically implies better performance and user experience. The average bitrates for DL and UL are calculated as:

$$\text{Avg. Bitrate} = \frac{1}{T}\sum_{t=1}^{T} R(t) \tag{5.1}$$

where $R(t)$ is the instantaneous bitrate at time t, and T is the total duration of measurement.

5.1.2 Latency

Latency is a measure of the time delay experienced in the network when transmitting data packets from source to destination. The latency of an xApp typically refers to the end-to-end processing time taken by the xApp to receive, process, and respond to events such as measurements or control triggers from the RAN via the E2 interface, computed as: xApp latency = Time taken from receiving an input (e.g., E2AP/KPM report) $\rightarrow$ to sending a corresponding control action or response (e.g., E2 Service Model (E2SM)-Control message)

$$\text{Latency} = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{5.2}$$

where L_i is the measured latency for the i-th packet and N is the total number of packets.

5.1.3 Standard Deviation (SD)

SD is a statistical metric that measures the variability of a set of values from their mean. In this experiment SD of DL throughput is considered consistent a parameter to quantify how stable the network is. The standard deviation of the DL throughput is given by:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{5.3}$$

where x_i represents each individual measurement, μ is the mean value, and N is the total number of samples.

A low standard deviation implies that the measurements are closely clustered around the mean, indicating stable and predictable performance. A high standard deviation, on the other hand, suggests large fluctuations, which can point to network instability due to the impact of competing xApps.

5.1.4 Standard Deviation Reduction

To quantify the effectiveness of a mitigation strategy, we compute the percentage reduction in SD compared to the baseline **With Conflict** scenario:

$$\text{SD Reduction (\%)} = \left(\frac{\sigma_{\text{conflict}} - \sigma_{\text{CM}}}{\sigma_{\text{conflict}}}\right) \times 100$$

- σ_{conflict} is the standard deviation of metrics in the presence of conflict.
- σ_{CM} is the standard deviation of the metrics with conflict mitigation enabled.

A higher SD reduction percentage indicates improved throughput stability and consistency, thus reflecting a more effective conflict mitigation approach.

5.2 Result Evaluation

To evaluate the performance of the implemented mechanism, it is compared with other existing conflict mitigation mechanism [2] and [3]. Table 5.2 shows the metrics for five different scenarios: No Conflict, With Conflict, Time Based Resolution (TBR), Priority Resolution (PR)[20] and finally proposed DQN resolution. Average DL bit rate, UL bit rate, latency of each mechanism and SD of DL throughput, UL throughput and latency were calculated.

Table 5.2: Comparison of Performance Metrics Under Different Scenarios with two xApps

Metric	**No Conflict**	**With Conflict**	**TBR**	**PR**	**Proposed DQN**
Avg. DL Bitrate (Mbps)	15.702	12.738	12.862	15.053	15.142
Avg. UL Bitrate (kbps)	211.515	180.662	188.800	191.753	205.526
Avg. Latency (ms)	61.190	85.850	79.350	76.389	67.520
Avg. SD of DL Bitrate (Mbps)	0.000	2.964	2.840	0.649	0.560
Avg. SD of UL Bitrate (kbps)	0.000	30.853	22.715	19.762	5.989
Avg. SD of Latency (ms)	0.000	24.660	18.160	15.199	6.330

Table 5.3: Comparison of Performance Metrics Under Different Scenarios With Three xApps

Metric	**No Conflict**	**With Conflict**	**TBR**	**PR**	**Proposed DQN**
Average Downlink Bitrate (Mbps)	15.702	12.393	12.817	14.996	15.075
Average Uplink Bitrate (kbps)	211.515	186.000	189.100	192.849	209.473
Average Latency (ms)	61.190	96.623	92.385	85.762	79.907
Avg. SD of DL Bitrate (Mbps)	0.000	3.309	2.885	0.706	0.627
Avg. SD of UL Bitrate (kbps)	0.000	25.515	22.415	18.666	2.042
Avg. SD of Latency (ms)	0.000	35.433	31.195	24.572	18.717

Table 5.3 illustrates the impact of introducing a third xApp into the system and compares five experimental scenarios: **No Conflict**, **With Conflict**, **Time-Based (TB) Resolution**, **Priority-Based (PR) Resolution**, and the **Proposed DQN**-based Conflict Mitigation approach.

The third xApp significantly increases contention with the other two xApps, which target static slice optimization and fairness-based distribution. As seen in the table, the **With Conflict** scenario exhibits sharp degradation in performance, with the average downlink bitrate dropping to **8.393 Mbps**, uplink bitrate to **186.00 kbps**, and latency rising to **96.623 ms**. Standard deviation values for all metrics also increase, indicating unstable performance.

The **Time-Based (TB)** and **Priority-Based (PR)** mechanisms provide moderate improvements. However, the **Proposed DQN** xApp consistently achieves performance close to the **No Conflict** baseline, maintaining a high downlink bitrate of **15.075 Mbps** and reducing latency to **79.907 ms**. Notably, it achieves the lowest variance in uplink bitrate and latency, indicating stable and efficient conflict resolution.

Figure 5.7 presents a comparative analysis of the average DL bitrate alongside its SD for five different scenarios. The "No Conflict" case yields the highest DL bitrate (15.702 Mbps) with a standard deviation of 0, serving as the ideal baseline with stable throughput and no control interference. In contrast, the "With Conflict" and TBR scenarios experience significant reductions in both throughput (12.738 Mbps and 12.862 Mbps, respectively) and stability, as evidenced by their elevated SD values (2.964 Mbps and 2.840 Mbps). This reflects the negative impact of uncoordinated or weakly enforced conflict mitigation strategies, resulting in erratic control decisions and volatile resource allocation.

The PR method improves performance by enforcing deterministic control dominance, achieving a higher DL bitrate (15.053 Mbps) and a noticeably reduced SD (0.649 Mbps). Most notably, the Proposed DQN-based mechanism achieves both high throughput (15.142 Mbps) and the

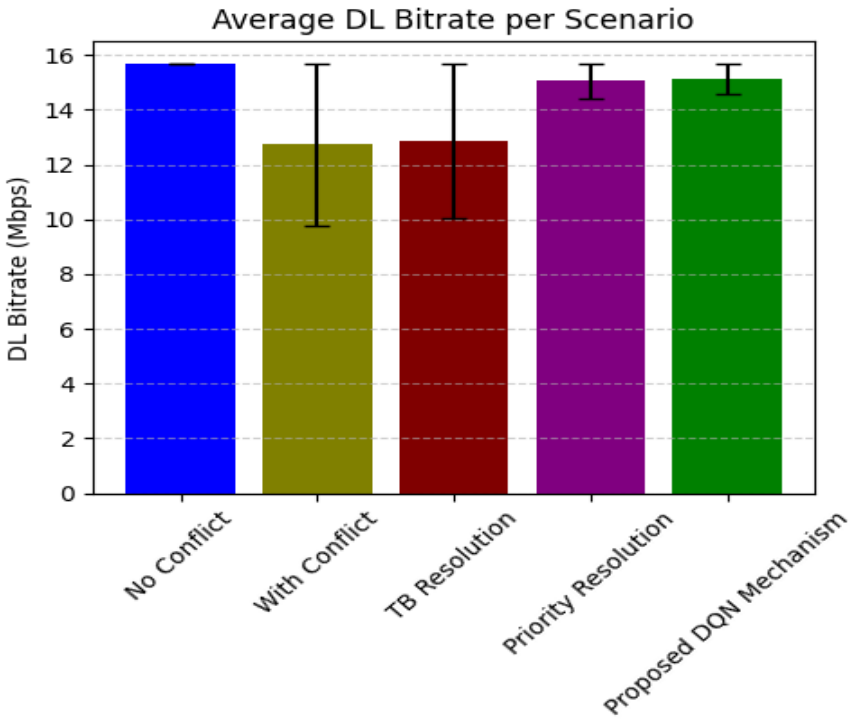

Figure 5.7: Average DL Throughput measurements for Each scenario

lowest variability among conflict scenarios (SD = 0.560 Mbps), demonstrating its ability to intelligently adapt to dynamic network states and efficiently mitigate control conflicts. These results collectively highlight that the proposed learning-based weighted priority approach not only maximizes average throughput but also ensures consistency in performance, making it highly effective for real-time RAN optimization under conflicting xApp behavior.

Figure 5.8 presents UL bitrate performance across the same five scenarios with the deviation from the baseline scenario. The No Conflict setup again sets the standard (211.5 kbps), demonstrating the best-case scenario. In contrast, the With Conflict scenario exhibits a substantial drop in average UL bitrate to 180.6 kbps, with a relatively high SD of 30.85 kbps, highlighting significant variability in uplink performance due to unmanaged xApp interference. The TBR and PR strategies offer modest improvements with average UL bitrates of 188.8 kbps and 191.8 kbps, and reduced SDs of 22.71 kbps and 19.76 kbps, respectively. These results suggest

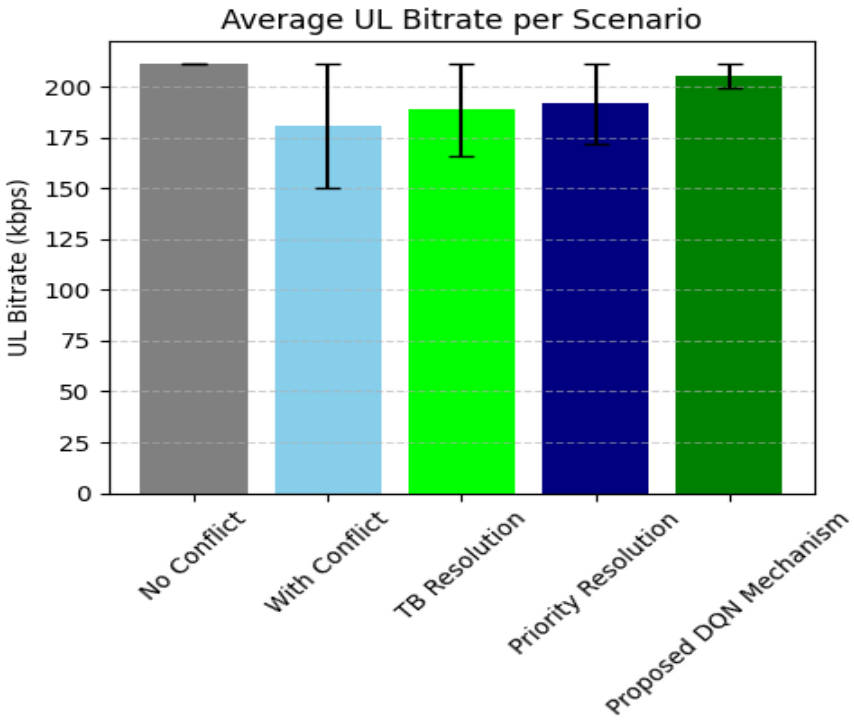

Figure 5.8: Average Uplink Throughput measurements for each scenario

that both deterministic conflict mitigation approaches help reduce fluctuations but still suffer from inconsistencies.

The Proposed DQN mechanism, however, not only recovers most of the lost performance—achieving an average UL bitrate of 205.5 kbps—but also maintains a low SD of just 5.99 kbps, underscoring its ability to intelligently manage resource conflicts and ensure consistent uplink throughput even under dynamic RAN conditions.

Figure 5.9 illustrates the latency performance — a crucial metric for real-time applications. The No Conflict case shows the lowest latency (61.2 ms), highlighting a smooth operation without interference. With Conflict case not only results in a significantly higher average latency (85.9 ms) but also exhibits the highest variability (SD = 24.66 ms), suggesting unpredictable and unstable behavior due to competing xApp control actions that disrupt scheduling efficiency and resource allocation.

The TBR strategy lowers the latency to 79.35 ms, with a reduced SD of 18.16 ms, indicating some improvement but still reflecting considerable fluctuation. The PR further improves stability, achieving an average latency of 76.4 ms and a smaller SD of 15.2 ms, suggesting more consistent handling of control conflicts.

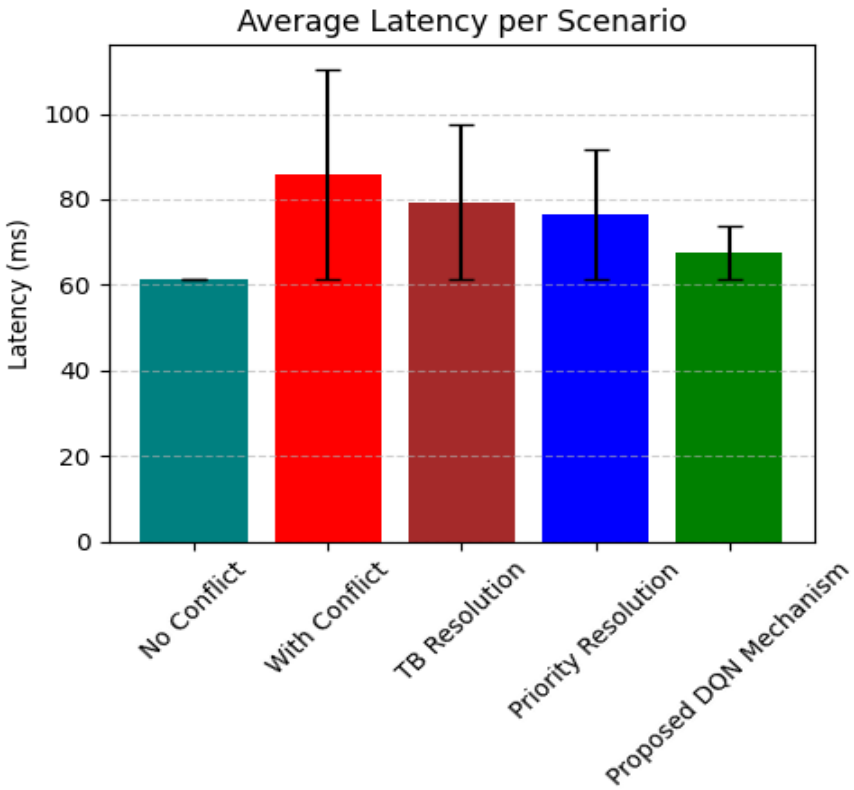

Figure 5.9: Average Latency measurements for each scenario

Notably, the proposed DQN mechanism outperforms the conflict mitigation baselines, bringing the latency down to 67.5 ms with a low SD of only 6.33 ms, demonstrating not just better average delay but also significantly more predictable and reliable system responsiveness. This highlights the learning-based method's effectiveness in stabilizing control decisions under dynamic network conditions.

The performance comparison reveals that network conflicts significantly degrade throughput and increase latency. The No Conflict scenario achieves the highest down link (15.702 Mbps)

and uplink (211.515 kbps) bitrates with the lowest latency (61.19 ms). However, introducing conflicts reduces throughput (12.738 Mbps DL, 180.662 kbps UL) and increases latency (85.85 ms).

Conflict mitigation techniques such as TBR and PR improve performance but do not fully restore optimal conditions. Proposed DQN mechanism proves to be the most effective, achieving near No Conflict performance with 15.142 Mbps DL bitrate, 205.526 kbps UL bitrate, and 67.52 ms latency. Additionally, standard deviation values indicate that DQN provides more stable throughput, demonstrating its effectiveness in mitigating performance degradation due to resource conflicts.

Table 5.4: Improvement Over Conflict Scenario Across Resolution Mechanisms

Resolution Mechanism	**DL Bitrate (%)**	**UL Bitrate (%)**	**Latency Reduction (%)**
TBR (Time-Based Resolution)	0.97%	4.50%	7.57%
PR (Priority-Based Resolution)	18.17%	6.14%	11.02%
Proposed DQN	18.87%	13.76%	21.35%

Table 5.4 presents the percentage improvements achieved by each conflict mitigation strategy over the conflict scenario baseline. The DQN-based solution demonstrates the most consistent and significant enhancements across all performance metrics: an 18.87% increase in DL bitrate, a 13.76% increase in UL bitrate, and a 21.35% reduction in average latency. In contrast, the PR offers a comparable boost in DL throughput (18.17%) but only modest latency and UL gains. The TBR shows the least improvement, highlighting its limited effectiveness in highly dynamic environments. These results validate the efficacy of the proposed DQN model in resolving multi-xApp conflicts more holistically.

Based on the observation DQN mechanism nullify the negative impact of direct conflict by reducing DL bitrate deviation by 81.10%. On the other hand time based resolution mechanism reduces this variability only 4% and priority based mechanism reduces the variability by 78%.

These results demonstrate the effectiveness of the proposed DQN-based weighted priority ap-

proach in managing dynamic and competing control logic among multiple xApps, preserving both throughput and latency performance.

These results illustrate that the proposed DQN weighted priority method effectively balances throughput and latency while maintaining system stability, making it the most promising candidate for practical deployment in multi-xApp RAN environments.

5.3 Discussion

5.3.1 Challenges

1. Testbed Constraints:

   Running experiments on the SDR-based testbed (srsRAN + Open5GS) involved limitations on available UEs, channel noise, and system restarts—factors not present in simulations.

2. Real-World Integration Complexity:

   Integrating DQN-based control logic with the Near-RT RIC required ensuring compatibility with the E2 interface and E2SM/KPM protocols. Handling the strict timing constraints of near-real-time control added complexity not present in simulation environments.

3. Coordination Across xApps:

   With multiple independently developed xApps operating concurrently, detecting and resolving overlapping control attempts (e.g., on the same PRBs) required continuous monitoring and decision-making.

4. Data Collection and Labeling:

Collecting representative KPM data for training the DQN agent in a live testbed environment was time-consuming and subject to environmental variability. Balancing training accuracy with real-time responsiveness was a key challenge.

5. Hyperparameter Tuning in Real-Time Settings:

   Finding the right batch size, exploration rate, and target update rate required extensive testing to ensure convergence and stability during operation.

5.3.2 Lessons Learned:

1. Real-World Testing Is Invaluable:

   Implementing the solution on a live testbed uncovered practical deployment challenges (timing, synchronization, interoperability) that simulations often overlook.

2. Modular Design Enables Conflict Mitigation:

   Designing the Conflict Mitigation Framework (CMF) as an independent xApp made it easier to monitor and intervene without modifying existing xApps directly.

3. Prioritization (Alphas) Is a Practical Starting Point:

   While not adaptive, setting pre-configured weights for xApp importance allowed deterministic resolution in early deployments, providing a baseline for more intelligent approaches like DQN.

4. Need for Better xApp Coordination Standards:

   The work highlights the need for standard APIs or metadata in O-RAN that allows xApps to declare their control scopes and potential conflicts upfront.

5. Monitoring and Visualization Tools Accelerate Debugging:

Building lightweight monitoring scripts and CSV-based logs significantly sped up detection of anomalies, conflicts, and DQN behavior during testbed evaluation.

Chapter 6

Conclusion and Future Work

This chapter summarizes the thesis contributions along with the scope of future work.

6.1 Conclusion

In conclusion, this thesis advances conflict management in O-RAN by addressing direct resource allocation conflicts between concurrently running xApps within the Near-RT RIC. xApps with conflicting resource allocation objectives were tested, enabling direct observation and detection of conflict patterns at runtime. These xApps served as the foundation for direct conflict detection within the RIC framework. Building on this, a DQN-based weighted pririty conflict mitigation mechanism was designed and integrated into the CM component of the Near-RT RIC. This learning-based approach dynamically optimized PRB allocation by adapting to real-time network states and xApp behavior, effectively resolving resource contention.

Extensive experimentation was conducted across five scenarios—ranging from baseline (no con-

flict) to conflict resolution using traditional methods (time-based, priority-based) and the proposed DQN-based scenario. The DQN model demonstrated clear convergence during training in the presence of conflicting xApps, with total rewards steadily increasing and stabilizing around optimal behavior.

The results reveal that the proposed DQN mechanism outperforms all other scenarios in terms of DL/UL bitrate, latency, and consistency, as evidenced by the lowest standard deviation in throughput. This confirms the DQN model's ability to not only resolve conflicts but also intelligently balance network goals, leading to improved stability and performance in the RAN.

This work has enabled for the first time an implementation and testing on a real O-RAN compliant and SDR-based testbed using O-RAN SC Near RT RIC, srsRAN and Open5GS. Unlike simulation-based approaches, this work surfaces practical challenges, such as handling real-time E2SM/KPM data, message encoding/decoding, xApp execution latency, and system state monitoring. These implementation details are crucial for moving from theoretical proposals to deployable RAN intelligence solutions.

Overall, this work validates the effectiveness of proposed DQN based weighted priority mechanism for adaptive and intelligent RAN control, offering a scalable and conflict-aware solution within O-RAN-compliant Near-RT RIC frameworks.

6.1.1 Future Work

Future research in conflict mitigation can significantly enhance the adaptability and efficiency of current solutions. Promising directions include the Double DQN, Dueling DQN, or Proximal Policy Optimization (PPO)—could be considered to further improve decision stability and convergence speed under complex and dynamic conflict scenarios.

Additionally, integrating federated learning would allow distributed xApps to collaborate in

learning optimal policies while preserving privacy and reducing communication overhead. This can be especially valuable in multi-operator environments or deployments across geographically distributed RAN nodes. Future work can scale the setup to include larger numbers of UEs, slices, and diverse xApp strategies, making the system more representative of real-world RAN deployments. To further improve the scope of this work, more diverse conflict types, indirect conflict and implicit conflict can be incorporated, allowing the model to generalize across various xApp interactions. Incorporating user-centric metrics such as QoE, rather than solely relying on system-level KPIs like bitrate and latency, would support more personalized and service-aware PRB allocation strategies.

Finally, Additionally, others can extend this work by applying multi-agent reinforcement learning (MARL), where each xApp is modeled as an independent agent learning to cooperate or compete. This can reflect more realistic RAN scenarios involving heterogeneous objectives. The framework can also be extended to operate in heterogeneous networks (HetNets), involving macro, micro, and small cells, or even across multi-vendor RAN ecosystems, increasing robustness and interoperability. Finally, researchers can explore the impact of policy constraints, such as service level agreements(SLA) or network slicing isolation requirements, to develop a more policy-aware conflict resolution strategy.

Bibliography

[1] A. Tripathi, J. S. R. Mallu, M. H. Rahman, A. Sultana, A. Sathish, A. Huff, M. Roy Chowdhury, and A. P. Da Silva, "End-to-End O-RAN Control-Loop For Radio Resource Allocation in SDR-Based 5G Network," in *IEEE MILCOM*, pp. 253–254, 2023.

[2] A. Sultana, F. Bashar, M. R. Chowdhury, and A. P. Da Silva, "A software-defined radio based o-ran platform for xapp conflict detection and mitigation," in *IEEE Military Communications Conference (MILCOM)*, pp. 686–687, 2024.

[3] A. Sultana, C. Adamczyk, M. R. Chowdhury, A. Kliks, and A. D. Silva, "Experimental evaluation of xapp conflict mitigation framework in o-ran: Insights from testbed deployment in otic," 2025.

[4] O-RAN Working Group 3, *Near-Real-time RAN Intelligent Controller Architecture*. O-RAN Alliance, October 2022. v03.00.

[5] H. Zhang, H. Zhou, and M. Erol-Kantarci, "Team learning-based resource allocation for open radio access network (o-ran)," *CoRR*, vol. abs/2201.07385, 2022. [Online]. Available: https://arxiv.org/abs/2201.07385.

[6] C. Adamczyk and A. Kliks, "Conflict Mitigation Framework and Conflict Detection in O-RAN Near-RT RIC," *IEEE Communications Magazine*, vol. 61, pp. 199–205, December 2023.

[7] P. B. del Prever, S. D'oro, L. Bonati, M. Polese, M. Tsampazi, H. Lehmann, and T. Melodia, "Pacifista: Conflict evaluation and management in open ran," *ArXiv*, vol. abs/2405.04395, 2024.

[8] A. Zolghadr, J. F. Santos, L. A. DaSilva, and J. Kibiłda, "Learning and reconstructing conflicts in o-ran: A graph neural network approach," 2025.

[9] I. Cinemre, T. Mahmoodi, and A. Farzaneh, "xapp conflict mitigation with scheduler," arXiv preprint arXiv:2504.06867, 2025.

[10] S. Bakri, I. Dey, H. Siljak, M. Ruffini, and N. Marchetti, "Mitigating xapp conflicts for efficient network slicing in 6g o-ran: a graph convolutional-based attention network approach," arXiv preprint arXiv:2504.17590, 2025.

[11] A. Wadud, F. Golpayegani, and N. Afraz, "xapp-level conflict mitigation in o-ran, a mobility driven energy saving case," in *Proceedings of the IEEE International Conference*, (TBD), IEEE, 2025.

[12] O-RAN Working Group 3, *Near-Real-time RAN Intelligent Controller Architecture*. O-RAN Alliance, June 2024. v06.00.

[13] "O-ran rmr user guide." https://docs.o-ran-sc.org/projects/o-ran-sc-ric-plt-lib-rmr/en/latest/user-guide.html. Last Accessed: 02/28/2023.

[14] O-RAN Working Group 3, *Near-Real-time RAN Intelligent Controller: Near-RT RIC Architecture*. O-RAN Alliance, October 2022. Technical Specification (TS) O-RAN.WG3.RICARCH-v03.00, Last Accessed: 02/28/2023.

[15] O-RAN Working Group 3, *Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles*. O-RAN Alliance, July 2022. Technical Specification (TS) O-RAN.WG3.E2GAP-v02.02, Last Accessed: 07/31/2022.

[16] srsRAN Project, "Grafana gui documentation," 2024. Accessed: March 11, 2025.

[17] srsRAN Project, *O-RAN gNB Overview: srsRAN Project gNB*, 2024. [Accessed: Nov. 2024].

[18] O-RAN Working Group 3, *Near-Real-time RAN Intelligent Controller, E2 Service Model (E2SM), RAN Control*. O-RAN Alliance, June 2023. v03.00.

[19] A. da Silva, M. R. Chowdhury, A. Sathish, A. Tripathi, S. F. Midkiff, and L. A. da Silva, "CCI xG Testbed: An O-RAN Based Platform for Future Wireless Network Experimentation," *IEEE Communications Magazine*, 2025. to appear.

[20] O-RAN Working Group 3, *Near-Real-time RAN Intelligent Controller and E2 Interface, Conflict Mitigation*. O-RAN Alliance, July 2024. v00.00.06.

Acronyms

API Application Protocol Interface. 12, 15, 16, 61

ASN.1 Abstract Syntax Notation One. 14

CCA Central Controller Agent. 29, 32, 47

CCI Commomnwealth Cyber Initiative. 43

CM Conflict Mitigation. i, 2, 3, 6, 7, 23, 29, 32, 43, 63

CMF Conflict Mitigation Framework. 6

Dbaas Database as a service. 15

DL Downlink. i, 27, 45, 46, 50, 52–55, 59, 64

DMS Deployment Management Services. 15

DQN Deep Q-Network. i, 3, 4, 8, 32, 33, 36–38, 40, 43–45, 50, 54, 55, 57–64

E2AP E2 Application Protocol. 14, 16, 18, 52

E2E End-to-End. 12

E2SM E2 Service Model. 20, 52

E2SM-RC E2 Service Model RAN Control. 30

eMBB enhanced Mobile Broadband. 29

gNB gNodeB. 29, 31

IE Information Element. 18

IoT Internet of Things. i

JSON Java Script Object Notation. 20

KPI Key performance Indicators. 65

KPM Key Performance Measurement. 7, 52, 61

MIMO Multiple Input Multiple Output. 44

ML Machine Learning. 6

MNO Mobile Network Operators. 23

Near-RT RIC Near Real-Time RAN Intelligent Controller. i, 2, 3, 6, 9, 11–16, 18, 20, 21, 23, 26, 29, 30, 32, 43, 44, 60, 63, 64

NN Neural Network. 38

Non-RT RIC Non Real-Time RAN Intelligent Controller. 11

O-RAN Open Radio Access Network. i, 1–4, 6–11, 18, 23–25, 29, 43, 46, 61, 63, 64

O-RAN SC O-RAN Software Community. 30, 43

OSC O-RAN Software Community. 11, 15

OTA Over-The-Air. 2, 7

PLMN Public Land Mobile Network. 16

PR Priority Resolution. 54–56, 58, 59

PRB Physical Resource Block. i, 3, 6, 8, 25–32, 36–38, 40–42, 44–46, 50, 63, 65

QoE Quality of Experience. 65

QoS Quality of Service. 1, 2, 6, 23, 24, 26, 27, 32, 47

RAN Radio Access Network. i, 1–3, 6–8, 10, 11, 14–16, 18, 20, 22–26, 28, 29, 32, 44, 46, 49, 52, 56, 57, 60, 64, 65

RF Radio Frequency. 2, 7, 23, 25, 44

RIC RAN Intelligent Controller. i, 10–16, 18, 20, 21, 23, 24, 26, 30, 44, 63

RL Reinforcement Learning. 50

RMR RIC Message Router. 13, 14, 21

SCTP Stream Control Transmission Protocol. 13

SD Standard Deviation. 53–58

SDL Shared Data Layer. 16, 21

SDR Software Define Radio. i, 3, 4, 43, 60, 64

SM Slice Management. 23, 24

SMO Service Management and Orchestration. 15

srsRAN Software Radio Systems RAN. 29