FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Multimodal O-RAN Framework for Intelligent Control of Mobile gNBs

Pedro Rafael Rocha Duarte

M.SC. DISSERTATION

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Manuel Ricardo, PhD

Second Supervisor: André Coelho, PhD

July 24, 2025

Resumo

Historicamente, as redes móveis tradicionais basearam-se em arquiteturas monolíticas e fechadas, que limitam a flexibilidade, restringem a interoperabilidade e abrandam o ritmo da inovação. Com o aumento das exigências nas comunicações sem fios, as gerações futuras de telecomunicações estão a transitar para arquiteturas de redes de acesso rádio (RANs) abertas e modulares. Um desses paradigmas é a *Open RAN* (O-RAN), que introduz interfaces normalizadas, como a interface E2, e *RAN Intelligent Controllers* (RICs), permitindo o controlo inteligente e em tempo real do comportamento da RAN através de aplicações modulares designadas por *xApps*.

Com estes avanços, prevê-se que as futuras RAN operem em ambientes cada vez mais dinâmicos, onde a mobilidade dos utilizadores e as obstruções da Linha de Vista (LoS) comprometem a estabilidade de ligações de alto desempenho. Esta realidade motiva a investigação em nós de rádio (*gNBs*) móveis, capazes de se reposicionar em resposta às condições do ambiente, bem como a integração de sistemas auxiliados por perceção, que proporcionem consciência do espaço físico envolvente.

A presente dissertação, desenvolvida no âmbito do projeto *Horizon Europe CONVERGE*, aborda este desafio propondo uma estrutura O-RAN multimodal para o reposicionamento inteligente do gNB na presença de obstruções da LoS. A solução introduz novos Modos de Serviço E2 para a transmissão de informação espacial e visual através da interface E2 da O-RAN. Estes dados multimodais são processados por uma *xApp* inovadora, que integra um modelo *Deep Q-Network* (DQN) para inferir e executar decisões de mobilidade do gNB em tempo real.

Um contributo adicional desta dissertação é o *CONVERGE Chamber Simulator* (CC-SIM), um ambiente de simulação 3D desenvolvido de raiz para suportar a experimentação de redes orientadas por perceção. O CC-SIM suporta três modos operacionais: i) *treino*, para desenvolvimento de agentes de *Reinforcement Learning*; ii) *simulação*, para inferência e validação em modo *offline*; e iii) *teste em tempo real*, para integração direta com a estrutura O-RAN. O CC-SIM é integrado com uma rede 5G autónoma, tirando partido do simulador de rádio frequência (*RFsimulator*) da *OpenAirInterface* (OAI), garantindo um funcionamento realista.

Para validar a estrutura O-RAN proposta, o CC-SIM foi utilizado para treinar com sucesso um agente de *Reinforcement Learning* (RL) capaz de ajustar autonomamente a posição do gNB em resposta ao movimento dinâmico de um obstáculo e de um *User Equipment* (UE). A validação experimental, conduzida sob diversos cenários de mobilidade, demonstra a capacidade do sistema em preservar a LoS e reduzir o tempo de obstrução em até **42%** em comparação com implementações estáticas do gNB. Os resultados obtidos confirmam a eficácia da solução proposta na integração de controlo multimodal e inteligente em redes móveis futuras.

Abstract

Traditional mobile networks have historically relied on monolithic, closed architectures that hinder flexibility, limit interoperability, and slow the pace innovation. As wireless communications demands increase, future networks are transitioning toward open, modular frameworks. One such paradigm is the Open Radio Access Network (O-RAN), which introduces standardized interfaces, such as the E2 interface, and Radio Access Network (RAN) Intelligent Controllers (RICs), enabling intelligent, real-time control of RAN behavior via modular applications known as xApps.

With these advances, future RANs are expected to operate in increasingly dynamic environments, where user mobility and Line-of-Sight (LoS) obstructions compromise the stability of high-performance links. This motivates research into mobile base stations (gNBs), capable of repositioning themselves in response to environmental conditions, alongside the integration of perception-aided systems to provide awareness of physical surroundings.

This dissertation, developed within the scope of the Horizon Europe CONVERGE project, addresses this challenge by proposing a multimodal O-RAN framework for intelligent gNB repositioning in the presence of LoS obstructions. The solution introduces novel Service Model messages to transmit spatial and visual information via the O-RAN E2 interface. These multimodal inputs are processed by a novel xApp, that incorporates a Deep Q-Network (DQN) model to infer and execute real-time gNB mobility decisions.

A further contribution of this dissertation, is the CONVERGE Chamber Simulator (CC-SIM), a custom-built 3D simulation environment designed to support perception-driven network experimentation. CC-SIM supports three operational modes: i) *training*, for reinforcement learning agent development; ii) *simulation*, for offline inference and validation, and iii) *live testing*, for real-time integration with the O-RAN framework. CC-SIM integrates with a standalone 5G network, taking advantage of the OpenAirInterface (OAI) Radio Frequency simulator (RFsimulator), ensuring operational realism.

In order to validate the proposed O-RAN framework, CC-SIM was used to successfully train a Reinforcement Learning (RL) agent capable of autonomously adjusting the gNB's position in response to dynamic obstacle and User Equipment movement. Experimental validation under diverse mobility scenarios, demonstrates the system's ability to preserve LoS and reduce obstruction time by up to **42%** compared to static deployments. The obtained results confirm the effectiveness of the proposed multimodal approach in enabling adaptive, intelligent control in future mobile networks.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice. This work is aligned with the following SDGs:

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 17 Strengthen the means of implementation and revitalize the global partnership for sustainable development

Target 9.5 Enhance scientific research, upgrade the technological capabilities of industrial sectors in all countries, in particular developing countries, including, by 2030, encouraging innovation and substantially increasing the number of research and development workers per 1 million people and public and private research and development spending

Target 17.6 Enhance North-South, South-South and triangular regional and international cooperation on and access to science, technology and innovation and enhance knowledge sharing on mutually agreed terms, including through improved coordination among existing mechanisms, in particular at the United Nations level, and through a global technology facilitation mechanism

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.5	Contributing to the broader research and development ecosystem within 6G and O-RAN initiatives.	Alignment with national and EU R&D priorities; contribution to funded research outputs;
17	17.6	Participation in the CONVERGE consortium and use of open-source platforms like OpenAirInterface and FlexRIC promote international collaboration, transparency, and innovation in future wireless networks.	Number of consortium partners engaged; number of interoperable components developed or contributed; volume of shared research outputs

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Professor Manuel Ricardo and Professor André Coelho, for their continuous support, guidance, and genuine concern throughout the development of this dissertation. Their expertise and availability contributed significantly for the direction and quality of this work. I also extend my thanks to INESC TEC, particularly the Centre for Telecommunications and Multimedia (CTM), for providing the research environment and resources necessary for this project.

I am especially grateful to my colleagues, whose companionship and insight made this journey both productive and memorable. In particular, I thank Alexandre Costa, Francisco Silva, Miguel Silva, Rodrigo Alves, and Tiago Claro for the countless brainstorming sessions and late-night dissertation discussions, which played a vital role in my progress and success.

Lastly, I want to thank my family for always being a source of strength and for creating the supportive environment that made this journey possible.

This work was supported by the CONVERGE project which has received funding under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101094831.

Pedro Duarte

"O querer e o trabalho transformam o sonho em realidade."

Colégio Didálvi

Contents

List of Figures

List of Tables

List of Acronyms

5GC	5G Core Network
AI	Artificial Intelligence
AR	Augmented Reality
BB	Bounding Box
BBU	Baseband Unit
CC-SIM	CONVERGE Chamber Simulator
C-gNB	CONVERGE gNB
CgNBCF	CONVERGE gNB Control Function
C-LIS	CONVERGE LIS
CLISCF	CONVERGE LIS Control Function
C-RAN	Centralized-Radio Access Network
CU	Central Unit
CV	Computer Vision
DL	Deep Learning
DNN	Deep Neural Network
DQN	Deep Q-Network
D-RAN	Distributed-Radio Access Network
DRL	Deep Reinforcement Learning
DU	Distributed Unit
E2AP	E2 Application Protocol
eMMB	Enhanced Mobile Broadband
gNB	Next-generation Node B
GUI	Graphical User Interface
IMSI	International Mobile Subscriber Identity
IoT	Internet of Things
ISAC	Integrated Sensing and Communications
ITU	International Telecommunication Union
KPM	Key Performance Metrics
LIS	Large Intelligent Surface
LoS	Line-of-Sight
MDP	Markov Decision Process
ML	Machine Learning
mMTC	Massive Machine-Type Communications
mmWave	Millimeter-wave
MS	Mobile Station
NFV	Network Function Virtualization
NG-RAN	Next-generation Radio Access Network
non-RT	non-Real Time
near-RT	near-Real Time

OAI	OpenAirInterface
O-CU-CP	O-RAN Central Unit Control Plane
O-CU-UP	O-RAN Central Unit User Plane
O-DU	O-RAN Distributed Unit
O-RAN	Open-Radio Access Network
O-RU	O-RAN Radio Unit
PLMN	Public Land Mobile Network
PUSCH	Physical Uplink Shared Channel
QoS	Quality of Service
RAN	Radio Access Network
RC	RAN Control
RIC	Radio Access Network Intelligent Controller
RIS	Reconfigurable Intelligent Surface
RF	Radio Frequency
RFsim	OAI RF Simulator
RGB-D	Red, Green, Blue and Depth
RL	Reinforcement Learning
RRU	Remote Radio Unit
RU	Radio Unit
SA	Standalone
SDN	Software Defined Networking
SM	Service Model
SNR	Signal-to-Noise Ratio
STCP	Stream Control Transmission Protocol
SVWC	Sensing and Computer Vision-aided Wireless Communications
THz	Terahertz
UE	User Equipment
URLLC	Ultra-Reliable Low-Latency Communications
V-RAN	Virtual-Radio Access Network

Chapter 1

Introduction

1.1 Context

The rapid evolution of mobile networks, driven by an increase in users, connected devices, and data-intensive applications, has driven the development of progressively advanced telecommunications solutions across successive generations. However, this progress has introduced increased complexity, typically characterized by proprietary, closed solutions that limit the openness and interoperability of network components. This reliance has motivated the adoption of open and interoperable systems.

The emergence of 6G networks seeks to address these challenges, offering a new paradigm focused on openness, flexibility, and improved capabilities. A central component of this evolution is the deployment of mobile base stations, which play a critical role in responding to rapid changes in network conditions by leveraging optimized Radio Access Network (RAN) configurations that ensure high levels of Quality of Service (QoS). These base stations are designed to be adaptive, scalable, and efficient, meeting the demands of 6G scenarios and use cases. The Open-RAN (O-RAN) architecture facilitates this adaptability by introducing open interfaces, enabling greater flexibility and interoperability between different vendors and network components. A fundamental aspect of this architecture is the E2 interface, which provides standardized connectivity between the RAN and external control functions such as the near-Real Time RAN Intelligent Controller, allowing for high-resolution telemetry collection and control actuation. Additionally, O-RAN supports programmable components, such as xApps, which optimize network operations based on real-time data, enabling dynamic adjustments to network conditions. An important enabler of xApps is sensing-aided communications, a key feature of 6G, which integrates video cameras and other sensors for improved environmental awareness. This integration allows the network to dynamically adjust to its surroundings, improving connectivity, reliability, and overall user experience.

Within this scope, computer vision (CV) plays a crucial role in interpreting the captured visual data, performing tasks such as object recognition and real-time monitoring, enabling proactive

obstacle detection to mitigate signal blockage. This integration brings numerous benefits, including improved situational awareness, optimized resource allocation, and enhanced user experience, while supporting state-of-the-art use cases, including autonomous vehicles, smart cities, and immersive virtual environments.

This dissertation is aligned with the CONVERGE research project [1], led by INESC TEC. CONVERGE embraces the motto of "view-to-communicate and communicate-to-view", aiming to create a novel toolset that tightly integrates radio, computer vision, and sensing-based technologies. As part of this effort, the project proposes the development of a physically controlled experimentation room, referred to as the CONVERGE chamber, designed to support multimodal data acquisition and real-time network control. An illustration of the chamber is presented in Figure 1.1.

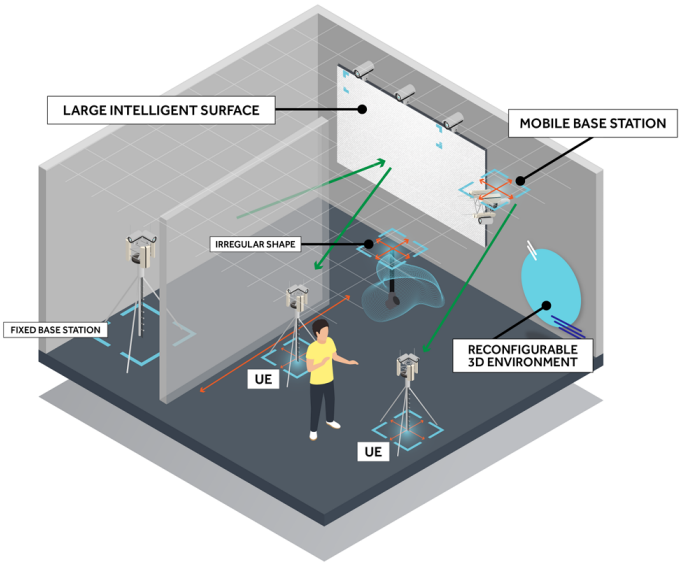


Figure 1.1: Illustration of the CONVERGE chamber [2].

1.2 Problem and Motivation

The motivation for this dissertation lies in addressing the growing challenges of mobile network configuration and management, while exploring the potential benefits of open and interoperable network components. 6G is positioned as a transformative solution to these challenges, promising unprecedented improvements in network performance, scalability, and adaptability.

A major challenge faced by next-generation mobile networks is the blockage of Line-of-Sight (LoS). Since 6G is envisioned to operate at sub-THz and THz frequencies, its communications links will be more vulnerable to obstacle disturbances in dynamic environments. Specifically, high-frequency signals in these bands struggle to pass through solid structures such as buildings and walls, resulting in significant signal attenuation. In urban environments, this challenge is exacerbated by moving vehicles, pedestrians, and other dynamic obstacles, further disrupting signal

propagation. The limitations of sub-THz and THz waves to penetrate common obstructions such as glass, concrete, and even human bodies limit their range and reliability, making it essential to develop new solutions to mitigate these effects, leveraging adaptive beam management and dynamic network reconfiguration. This challenge creates an opportunity to explore the integration of sensing, CV, and radio frequency (RF) technologies into wireless communications, given their shared dependency on LoS.

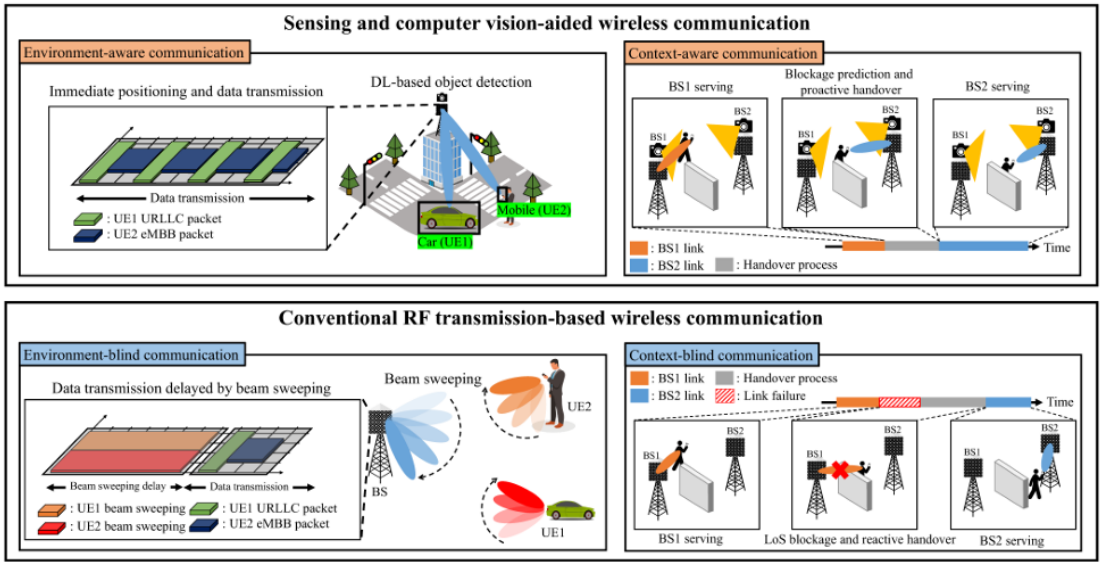


Figure 1.2: Comparison of a sensing and CV-aided wireless communications approach versus a conventional RF transmission-based communications system [3].

As depicted in Figure 1.2, combining CV techniques for real-time environmental understanding with advanced RF communications systems enables optimized signal propagation, improves network reliability, and mitigates interference caused by obstacles. Figure 1.2 compares the performance between a *sensing and CV-aided wireless communications* (SVWC) approach versus a conventional RF transmission-based communications system. The first case (left-hand side of Fig. 1.2) shows the traditional approach of beam sweeping, which introduces delay and overhead before starting the data transmission, opposed by the SVWC approach, which is capable of detecting the environment components and immediately starting data transmission. The second example (right-hand side of Fig. 1.2) shows the traditional handover approach, a reactive process only triggered when the BS senses a link failure, opposed by the SVWC approach, which is able to understand the environment's context, predict blockage and proactively hand over communications from BS1 to BS2. This allows for the introduction of the advantages of the SVWC approach in reducing latency, minimizing overhead, and enhancing the reliability of communications, emphasizing its potential to pave the way for efficient and adaptive 6G networks capable of intelligently adjusting to dynamic conditions.

1.3 Objectives

The main objective of this dissertation was to design and implement an intelligent, learning-based control mechanism capable of dynamically adjusting the positioning of a mobile gNB within a RAN. The aim is to ensure LoS connectivity and enhance communications QoS. The proposed controller leverages multimodal data sources — including RF signal metrics and visual inputs from distributed video cameras — to interpret the surrounding environment and make informed mobility decisions. The solution is developed and deployed within the O-RAN architecture, ensuring alignment with the CONVERGE project's vision and promoting openness and modularity.

The specific objectives of this dissertation are:

1. Develop a framework for synthetic data acquisition from the CONVERGE chamber.
2. Define and implement novel message types for the O-RAN E2 interface, enabling structured exchange of multimodal data between network nodes.
3. Develop an O-RAN xApp hosting a trained machine learning (ML) agent capable of autonomously controlling the movement of a gNB.
4. Integrate the developed components and conduct an evaluation of the proposed solution, assessing its ability to maintain LoS connectivity, adapt to environmental dynamics, and improve radio link performance.

1.4 Contributions

This dissertation advances the emerging research on integrating CV-based information into O-RAN management algorithms. The main contributions are as follows:

1. **A Multimodal O-RAN Framework for Intelligent Control of Mobile gNBs.** Design and implementation of a novel multimodal O-RAN framework incorporating CV-derived contextual information to enable proactive gNB mobility decisions, while supporting the maintenance of high-quality radio links and LoS connectivity.
2. **CC-SIM: A 3D simulator for perception-driven network experimentation.** Development of a custom simulation environment for the CONVERGE chamber, designed to emulate dynamic radio environments and generate synthetic multimodal data. CC-SIM supports three operational modes: 1) *training*, for reinforcement learning agents development; 2) *simulation*, for offline inference, validation, and visualization; 3) *live testing*, for real-time integration with the O-RAN framework.
3. **Multimodal-based xApp.** Development of a novel xApp leveraging multimodal data inputs–including spatial and vision-based data–and a trained Deep Neural Network (DNN) to improve decision-making processes and overall RAN performance.

4. **Validation and performance assessment of the proposed solution.** Evaluation of the proposed solution through comprehensive performance assessments, demonstrating its responsiveness to dynamic obstacles and its ability to maintain LoS connectivity.

Two publications presenting the proposed solutions, along with their experimental validation and evaluation, are being prepared for submission to international conferences in the field of wireless networking.

Collectively, these contributions provide novel methodologies and tools for researchers and practitioners, fostering innovation and advancing the state of the art in intelligent wireless communications systems built on the O-RAN framework.

1.5 Document Structure

This document is structured as follows. Chapter 2 reviews the state of the art and related work, presenting the fundamental concepts that motivate this dissertation and framing the main problem addressed. The chapter also surveys recent solutions proposed in the literature. Chapter 3 describes the proposed and developed solution, detailing the design choices and implementation of each module. Chapter 4 presents the system validation methodology, the evaluation metrics employed, and the performance results obtained through simulation testing. Finally, Chapter 5 concludes the document by summarizing the key findings and discussing potential directions for future research.

Chapter 2

State of the Art

This chapter explains the fundamental concepts and reviews existing solutions related to the problem addressed in this dissertation. Section 2.1 introduces 5G and 6G network generations, focusing on their main applications, carrier frequencies, and target performance requirements. Section 2.2 describes the overall architecture of 5G networks and focuses on the evolution of RAN deployment approaches over successive generations, including O-RAN. Section 2.3 explores the O-RAN architecture, based on the 5G Next-Generation RAN framework, and highlights the main components and interfaces leveraged in this dissertation. Section 2.4 presents the CONVERGE project's architecture and objectives, including the description of the open-source software components employed in the system. Section 2.5 introduces the foundational concepts of Reinforcement Learning, including Q-learning and Deep Q-Networks (DQN), to provide the theoretical background required to understand the learning-based mobility control approach used in this dissertation. Finally, in Section 2.6, different solutions related to perception-aided RAN control and the integration of intelligent xApps within the O-RAN framework are presented.

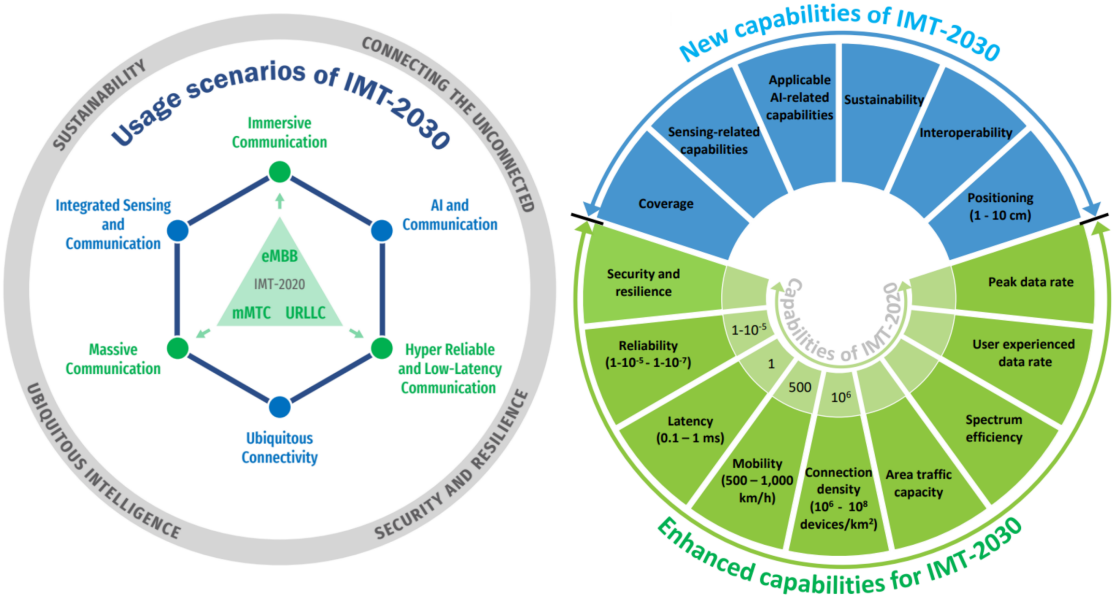
2.1 5G and 6G Characterization

The development of mobile communications systems has consistently transformed how humans interact with technology and one another. The progression from 4G to 5G marked a significant evolution in capabilities, enabling paradigms such as the Internet of Things (IoT) and augmented reality (AR). 5G introduced significant changes in wireless communications including ultra-high-speed data transfer rates, low latency, and massive connectivity. 5G networks primarily operate in the sub-6 GHz and millimeter-wave (mmWave) frequency bands, providing wide coverage and high capacity [4]. Despite significant advances, 5G faces challenges in meeting the growing traffic demand and supporting future applications requiring higher performance.

Building on 5G's foundation, 6G aims to redefine connectivity by surpassing 5G capabilities and introducing new ones, as shown in Figure 2.1b, while pushing frequencies to terahertz (THz) ranges and enabling ultra-low-latency communications for real-time applications. Standardized by the International Telecommunication Union (ITU) as IMT-2030 [5], 6G aims to expand the usage

scenarios established in IMT-2020 (5G), including enhanced Mobile Broadband (eMBB), Massive Machine-Type Communications (mMTC), and Ultra-Reliable Low-Latency Communications (URLLC); they are characterized as follows:

- **eMBB –** Provides high data rates and improved capacity for bandwidth-intensive applications, designed for scenarios such as urban, high-traffic demand areas.
- **mMTC –** Designed for massive IoT deployments, focusing on scalability and power efficiency over high data rates, supporting large numbers of low-power, low-data-rate devices, such as wireless sensors.
- **URLLC –** Designed for time-critical, highly reliable communications with ultra-low latency (below 1 ms), used in critical applications, including autonomous vehicles and industrial automation.



(a) Usage scenarios of IMT-2030.

(b) Capabilities envisioned by IMT-2030.

Figure 2.1: Characteristics defined by IMT-2030 [5].

In 6G, the traditional boundaries between eMBB, mMTC, and URLLC use cases are increasingly interrelated, giving rise to integrated and hybrid scenarios, as depicted in Figure 2.1a. For example, **Integrated Sensing and Communication (ISAC)** combines eMBB's high data capacity with mMTC's massive connectivity, enabling applications such as real-time environmental monitoring and advanced smart city infrastructure that require both extensive sensor deployment and high-speed data processing. **AI and Communication** integrates eMBB's bandwidth-intensive capabilities with ultra-reliable, low-latency communications, paving the way for intelligent systems such as AI-driven autonomous vehicles and robotic surgery, where real-time decision-making and data analysis are critical. Finally, **Ubiquitous Connectivity** combines mMTC and URLLC,

enabling seamless, reliable connectivity for billions of IoT devices in remote or mobile environments, while ensuring that devices maintain stable, low-latency communications, and enabling applications such as global logistics tracking and emergency response systems.

Another potential aspect of 6G is its integration with quantum communications technologies. Quantum communications offer unparalleled security through quantum key distribution (QKD) and enhanced computing capabilities [6]. These features emphasize 6G's transformative potential to address complex, diverse communications needs.

Overall, 6G goes beyond achieving higher data rates and enhanced wireless connectivity by promoting open-source solutions. This shift toward openness fosters increased collaboration within the community, accelerating technological advancements. By promoting transparency and flexibility, 6G cultivates an environment that accelerates innovation, facilitates rapid development of new technologies, and ensures a more inclusive and adaptable future for ubiquitous connectivity [7].

2.2 RAN Deployment Approaches

While the development of 6G architecture is in its early stages, the 5G architecture, defined by the 3GPP standards [8], provides a mature and comprehensive framework for this dissertation. The overall 5G system architecture, depicted in Figure 2.2, consists of three main components: User Equipment (UE), the Next-Generation Radio Access Network (NG-RAN), and the 5G Core Network (5GC).

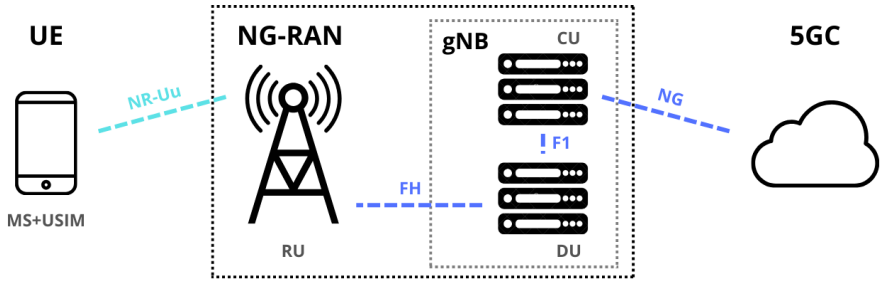


Figure 2.2: Overview of the 5G architecture.

- **UE**: Composed of a Mobile Station (MS) and a Universal Subscriber Identity Module (USIM), the UE includes devices such as smartphones, IoT nodes, and industrial equipment that connect to the network. The UE connects to the NG-RAN through the NR-Uu interface.
- **NG-RAN**: Responsible for managing wireless communications between the UE and the 5G Core Network, the main component of the NG-RAN is its 5G base station, referred to as next-generation Node B (gNB).
- **5GC**: A flexible, cloud-native core network that handles centralized functions such as authentication, data routing, and service delivery, enabling features such as network slicing

and multi-access edge computing. In 5G, the 5GC connects to the NG-RAN through the NG interface.

This modular architecture is designed for adaptability, with the **Radio Access Network** (RAN) acting as a crucial intermediary that ensures seamless communications between the UE and the Core network. It evolved significantly compared to earlier generations, driven by the need to support higher data rates, denser device connections, and more advanced services. This transformation includes advancements in the deployment architectures aimed at enhancing efficiency, flexibility, and scalability.

In traditional RAN architectures, particularly in 3G and 4G, the RAN is composed of two main components: the Baseband Unit (BBU) and the Remote Radio Unit (RRU), as depicted in Figure 2.3.

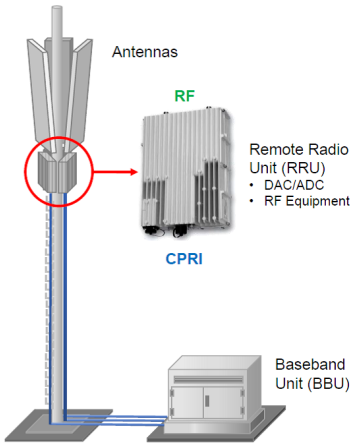


Figure 2.3: Traditional RAN deployment (D-RAN).

- **BBU -** Responsible for overall base station (BS) management, handling tasks such as baseband processing, radio resources management, and system maintenance for the efficient operation of the BS.
- **RRU -** Responsible for radio frequency (RF) signal conversion, filtering, and amplification. It interfaces with the BS antennas and the BBU through the Common Public Radio Interface (CPRI).

In earlier RAN deployments, RRUs were situated at cell towers or sites, closer to the antennas, while BBUs were located at the cell site and tightly coupled with the radio unit, resulting in high operational costs and limited flexibility. In this approach, known as **Distributed-RAN** (D-RAN), radio functions at each cell site are distributed, and backhaul links provide connectivity to the core network [9].

To improve scalability and flexibility, another deployment scheme, known as **Centralized or Cloud-RAN** (C-RAN), was introduced. This concept centralizes BBUs in a data center that

hosts multiple virtualized BBUs on a single hardware platform, enabling resource pooling and reducing costs, while the RRUs remain at the cell site near the antennas [9]. However, C-RAN presents some limitations, such as a higher risk of single-point failure and significant overhead in the links between the RRUs and the BBUs cloud [10]. These limitations led to advancements in virtualization and edge computing techniques.

Virtual-RAN (V-RAN) emerged as a derivative of C-RAN, leveraging Network Function Virtualization (NFV) and Software Defined Networking (SDN) principles to virtualize BBU functions, enabling them to run as software on general-purpose hardware. This virtualization decouples hardware from software, offering scalability and allowing operators to deploy network functions dynamically [9] [10]. However, it often relies on vendor-specific solutions, compromising interoperability between network components and limiting seamless integration across multi-vendor environments. These challenges motivated the evolution of the concept of **Open-RAN (O-RAN)**, designed around two main principles: *intelligence* and *openness*. The following section elaborates on the concept of O-RAN, as it is a central theme of this dissertation.

2.3 O-RAN

With the surge of 5G, the **Next-Generation RAN** (NG-RAN) was introduced as part of the 3GPP Release 15 specification [11], evolving from the previous RAN architecture to meet the demands for enhanced data rates, ultra-low latency, and massive device connectivity. As part of this evolution, the legacy Remote Radio Unit (RRU) evolved into the Radio Unit (**RU**), and the traditional Baseband Unit (BBU) was disaggregated into two distinct entities: the Central Unit (**CU**) and the Distributed Unit (**DU**). The NG-RAN adopts a modular design, consisting of the RU for RF functions, the DU for lower-layer processing, and the CU for higher-layer protocols. Figure 2.4 provides an overview of this architecture, showcasing the disaggregation of the gNB into the gNB-CU and gNB-DU, along with the interfaces between components.

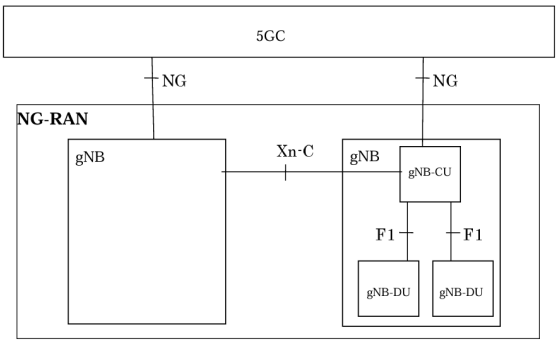


Figure 2.4: Overall architecture of NG-RAN [11].

The development of the **O-RAN** architecture builds on NG-RAN's modular foundation, advancing it with openness and interoperability as core design principles. While NG-RAN introduced structural disaggregation, O-RAN enhances this model through standardized interfaces and

functional split definitions, enabling multi-vendor deployments and dynamic network control. Led by the O-RAN Alliance, the architecture introduces intelligent, software-defined RAN components and real-time analytics that support AI/ML-driven optimization and closed-loop control.

2.3.1 O-RAN Functional Architecture

The O-RAN architecture extends the NG-RAN by introducing key components such as the RAN Intelligent Controllers (RICs), and defining open interfaces between disaggregated RAN elements. These additions allow for fine-grained and dynamic control over the radio network. Figure 2.5 provides a high-level view of the O-RAN architecture, emphasizing the integration of standardized control interfaces and the hierarchical structure of intelligent controllers.

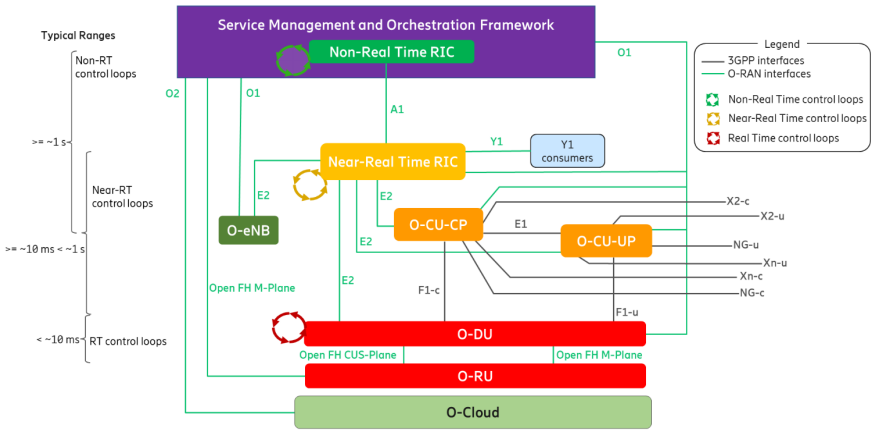


Figure 2.5: O-RAN architecture and control loops, with components and interfaces from O-RAN and 3GPP [12].

The Non-Real Time RIC (**non-RT RIC**), located within the Service Management and Orchestration (SMO) layer, performs long-timescale operations (>= 1s) such as AI/ML training, policy generation, and cross-domain analytics. These policies and insights are transmitted to the Near-Real Time RIC (**near-RT RIC**), which operates on a 10 ms to 1 s timescale and executes inference-driven control via modular applications known as xApps. The near-RT RIC communicates with RAN nodes via the standardized E2 interface, enabling the exchange of real-time telemetry and control commands. Below the RIC layers, the RAN is composed of disaggregated functional units: the O-RAN Central Unit Control Plane (**O-CU-CP**) and O-RAN Central Unit User Plane (**O-CU-UP**) handle signaling and user data respectively; the O-RAN Distributed Unit (**O-DU**) performs real-time processing, including scheduling and resource allocation, with latencies up to 10 milliseconds; and the O-RAN Radio Unit (**O-RU**) handles RF functions. These components are interconnected using standard 3GPP interfaces (e.g., F1, E1, and open fronthaul), allowing for modular and scalable deployments. Together, this architecture facilitates closed-loop control, improved interoperability, and the seamless integration of AI-native applications into the RAN.

2.3.2 Near-RT RIC and xApps

The **near-RT RIC** is one of the major innovations introduced by the O-RAN framework. Positioned between the SMO and the RAN, it serves as the execution layer for time-sensitive control and optimization functions. These are implemented via modular applications called **xApps**, which operate over real-time telemetry data (e.g., Key Performance Metrics (KPMs) such as throughput, latency, or packet loss) and can modify RAN behavior in response to environmental dynamics such as mobility, congestion, or interference.

The near-RT RIC communicates with the gNB (specifically the O-CU and O-DU) through the **E2 interface**, ingesting telemetry and issuing control commands in a looped manner. To support this interaction, each gNB hosts one or more **E2 agents**, which serve as intermediaries that translate internal node metrics and control information into E2-compliant messages. These agents implement one or more **Service Models**, each of which specifies how xApps running on the near-RT RIC interact with and control **E2 nodes** (gNBs).

2.3.3 E2 Interface and Service Models

The **E2 interface** is a key enabler of modular RAN control, providing a standardized channel for communication between the near-RT RIC and O-RAN-compliant network nodes such as the O-CU and O-DU. It supports the exchange of two primary categories of information:

- **Indication messages:** Telemetry data from the gNB to the RIC.
- **Control messages:** Commands from the RIC to reconfigure or influence RAN behavior.

The structure and procedures responsible for coordinating these exchanges are defined by the **E2 Application Protocol (E2AP)**, which manages the setup, maintenance, and teardown of the E2 connection. E2AP operates over Stream Control Transmission Protocol (SCTP) and serves as the control plane protocol for exchanging messages between the near-RT RIC and E2 nodes. Figure 2.6 illustrates the E2AP protocol stack and its relation to underlying transport layers.

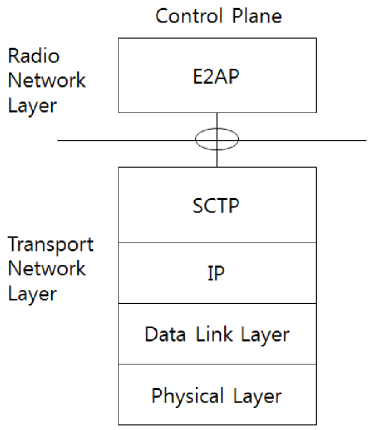


Figure 2.6: E2AP protocol stack [13].

Before any telemetry or control exchange can occur, the **E2 Setup Procedure** is initiated to establish communication between the RIC and E2 node. During this procedure, the E2 node announces its capabilities and supported Service Models by sending an *E2 Setup Request*, which the RIC acknowledges via an *E2 Setup Response*. This handshake formalizes the E2 connection and allows the RIC to subscribe to the telemetry types it expects to receive. The full procedure is depicted in Figure 2.7.

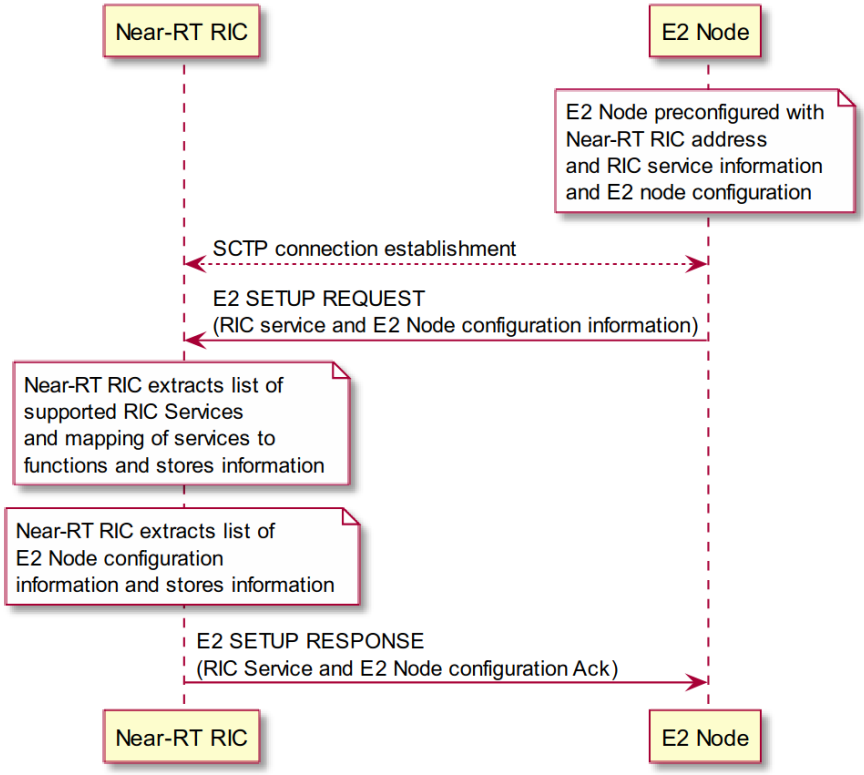


Figure 2.7: E2 Setup procedure [13].

Once the connection is established, real-time telemetry data can be streamed from the gNB to the near-RT RIC using E2AP **Indication** messages. These messages are received and processed by the appropriate **xApp** within the RIC, which monitors the data to infer the current network state. In parallel, the xApp can issue **Control** messages through the near-RT RIC, allowing it to reconfigure RAN behavior in real time. The message flows for these monitoring and controlling interactions are illustrated in Figure 2.8 and Figure 2.9, respectively.

The semantics and data models of the transmitted messages are determined by **Service Models (SMs)**, which specify domain-specific data types and procedures. While O-RAN defines standard SMs such as KPM and RAN Control (RC), it also supports the integration of custom SMs tailored to specific use cases.

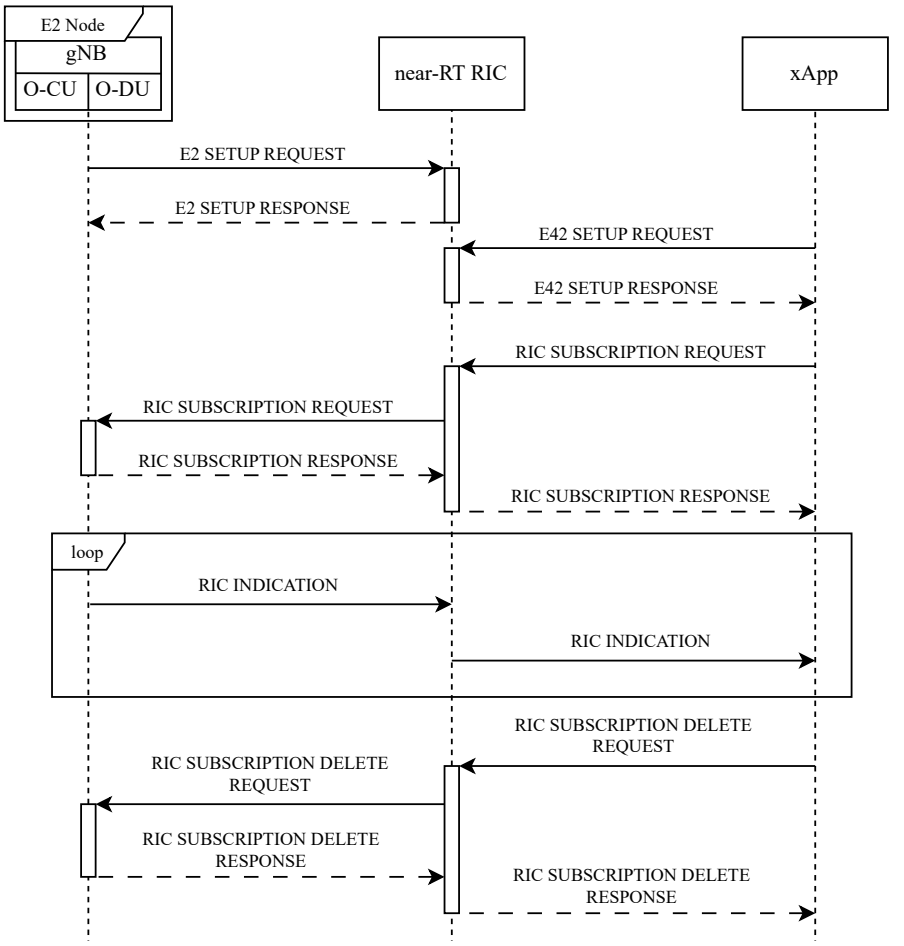


Figure 2.8: E2AP monitoring sequence diagram (adapted from [14]). The E42 interface functions similarly to the E2 interface and is designed for interfacing with xApps.

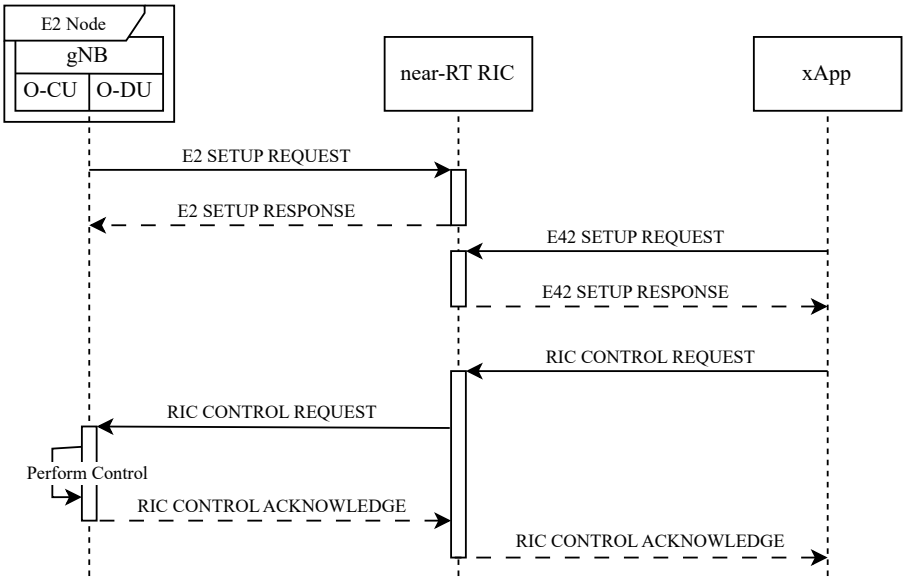


Figure 2.9: E2AP controlling sequence diagram (adapted from [14]).

In summary, O-RAN transforms traditional RAN architectures into programmable, AI-driven platforms. The introduction of components like the near-RT RIC, coupled with the flexibility of the E2 interface, enables the deployment of intelligent control strategies within real-time wireless systems.

2.4 CONVERGE Project

2.4.1 Architecture

In order to better understand the background of this dissertation, it is essential to analyze the architecture proposed by the Horizon Europe CONVERGE project. The architecture is organized into three main blocks, as depicted in Figure 2.10: 1) the CONVERGE chamber, a room for collecting of experimental data from radio communications, radio sensing, and CV-based sensing; 2) the CONVERGE Core, responsible for network functions, experiment session management, data orchestration, and ML algorithms; and 3) the CONVERGE Simulator, enabling data collection for the CONVERGE Digital Twin.

This dissertation particularly focuses on the CONVERGE chamber, illustrated in Figure 1.1. It is equipped with a gNB, a Large Intelligent Surface (LIS) [1], and a UE, enabling the collection of RF metrics. Each of these nodes is equipped with an RGB and Depth (RGB-D) camera, allowing the system to capture both visual data and depth information.

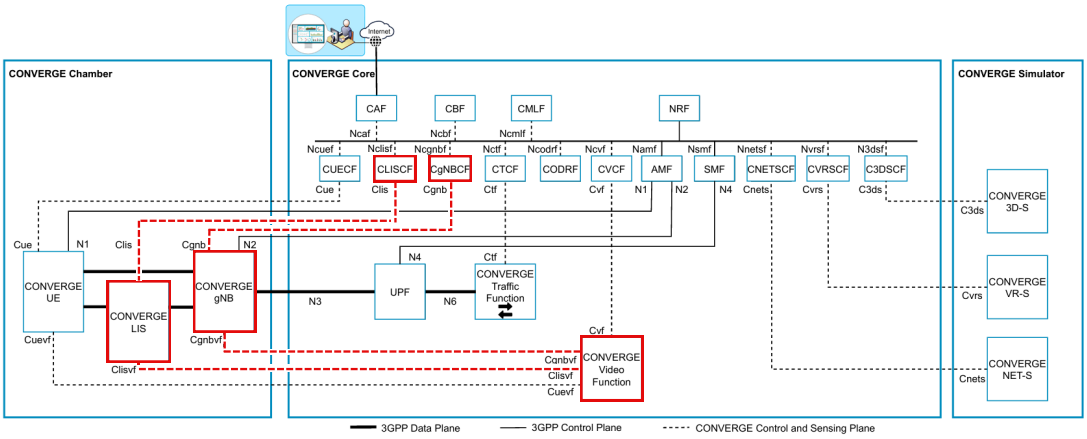


Figure 2.10: CONVERGE service-oriented architecture [2]. The main blocks and interfaces that were explored in this dissertation are highlighted in red.

The vision-aided CONVERGE gNB is the main focus of this dissertation, since the objective is to develop a system that autonomously control its placement. Based on the O-RAN architecture, as shown in Figure 2.11, it supports the integration of RAN intelligent controllers, enabling the collection of metrics required for informed decision-making and the dynamic configuration

[1] The Large Intelligent Surface (LIS) is a programmable Reconfigurable Intelligent Surface (RIS) equipped with video cameras to enable vision-aided wireless communications. It supports advanced functionalities, including beam steering, energy focusing, and programmable signal shaping.

of both O-CU and O-DU components. The CONVERGE gNB also includes an integrated RGB-D camera that communicates with the CONVERGE Video Function via the **Cgnbvf** interface. In parallel, it connects to the CONVERGE gNB Control Function (CgNBCF), hosted within the CONVERGE Core, through the **Cgnb** interface. This web-based interface operates over IP and exchanges JSON-formatted messages, allowing for bidirectional communication that supports both real-time monitoring of the gNB's position and velocity, as well as the delivery of control commands for mobility management. The integrated RGB-D camera serves as the local sensing module of the gNB, capturing depth-enhanced visual frames and forwarding them to the CONVERGE Video Function, which extracts semantic features used to inform RAN control decisions.

Another relevant node in this architecture is the CONVERGE LIS. It is equipped with an integrated RGB-D camera that interfaces with the CONVERGE Video Function via the **Clisvf** interface and connects to the CONVERGE LIS Control Function (CLISCF), also hosted within the CONVERGE Core, through the **Clis** interface, which operates similarly to Cgnb as a web-based, IP-enabled channel that exchanges JSON-formatted messages for position monitoring and control.

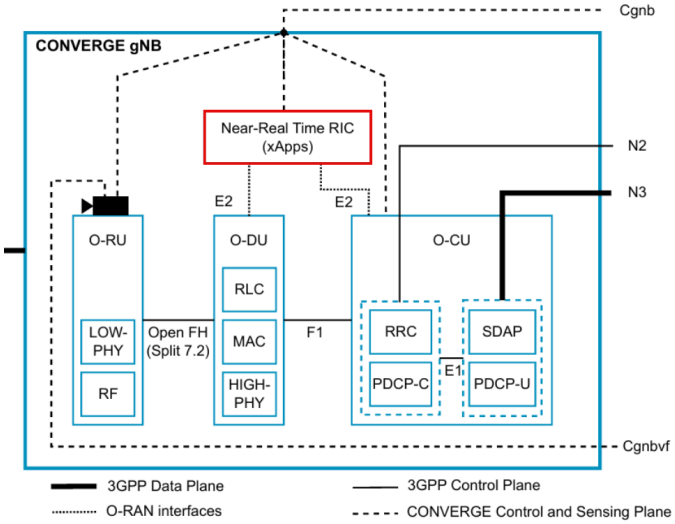


Figure 2.11: CONVERGE gNB architecture [2].

2.4.2 OpenAirInterface (OAI) and FlexRIC

Aligned with the vision of CONVERGE, which promotes open, modular, and perception-aware wireless systems, the project builds upon two main open-source platforms: **OpenAirInterface (OAI)** and **FlexRIC**. These software frameworks are widely recognized in both academia and industry for enabling advanced experimentation and rapid prototyping in 4G and 5G networks.

OpenAirInterface (OAI) is a complete, standards-compliant implementation of the 4G and 5G protocol stacks developed by the OAI Software Alliance [15]. It includes modular components for the 5G Core Network, gNB, and UE, and supports both simulated environments and over-the-air deployments. OAI's flexibility, transparency, and adherence to 3GPP specifications have made

it a foundational tool for wireless research, particularly where reproducibility and integration with external systems are essential.

FlexRIC [16], developed by Mosaic5G, is an extensible and lightweight near-RT RIC designed for the O-RAN architecture. It enables real-time RAN monitoring and control via the standardized E2 interface, allowing third-party applications (xApps) to influence RAN behavior based on telemetry and policy. FlexRIC is particularly valued for its modular design, support for custom Service Models (SMs), and ease of integration with open-source RAN platforms such as OAI.

Together, OAI and FlexRIC form the open-source software foundation of the CONVERGE project, providing a standards-aligned, interoperable, and highly programmable platform for implementing perception-aided and adaptive RAN control.

2.5 Reinforcement Learning

Machine Learning techniques, and in particular Reinforcement Learning (RL), have emerged as powerful tools for enabling intelligent control mechanisms in wireless networks, where rapidly changing conditions and dynamic environments require adaptive and data-driven decision-making strategies. RL is a machine learning paradigm in which an agent learns optimal behavior by interacting with an environment, aiming to maximize a cumulative reward over time. At each time step t, the agent observes the current state of the environment s_t from a state space S, selects an action a_t from an action space A according to a policy, and receives feedback in the form of a scalar reward and a new state. Through this trial-and-error process, the agent incrementally improves its decision-making strategy, typically modeled as a policy that maps states to actions. The learning objective is to discover a policy π that maximizes the expected sum of future rewards, often discounted over time [17].

Formally, an RL problem is modeled as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ), where:

- S is the set of environment states.
- A is the set of actions available to the agent.
- $P(s'|s,a)$ is the transition probability function, representing the probability of reaching state s' after taking action a in state s.
- $R(s,a)$ is the reward function, which returns a scalar feedback signal for action a taken in state s.
- $\gamma \in [0,1)$ is the discount factor, determining the importance of future rewards.

The agent's goal is to learn a policy $\pi : S \to A$ that maximizes the expected cumulative discounted reward:

$$\mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma r_t\right] \tag{2.1}$$

where r_t is the reward received at time step t under policy π.

RL has gained increasing relevance in wireless communications, where it offers a model-free and adaptive solution to problems characterized by uncertainty, time variance, and high-dimensional decision spaces. In scenarios involving dynamic user mobility, varying channel conditions, and resource constraints, RL-based approaches have demonstrated promising capabilities in optimizing tasks such as beamforming, scheduling, handover, and channel estimation [18].

2.5.1 Q-Learning

Q-Learning is a foundational algorithm in *model-free* reinforcement learning, introduced to estimate the optimal action-value function $Q^*(s,a)$ without requiring a model of the environment's dynamics. The agent learns from experience by iteratively updating the Q-values based on observed rewards and state transitions. The core idea is to approximate the maximum future return from a given state-action pair using the Bellman optimality equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{2.2}$$

where:

- α is the learning rate $(0 < \alpha \leq 1)$.
- r is the immediate reward received after taking action a in state s.
- $\max_{a'} Q(s',a')$ represents the best predicted reward for the next state s'.

Q-learning is considered an *off-policy* algorithm, as it learns the optimal policy independently of the agent's current behavior policy (often ε-greedy). It converges to the optimal action-value function Q^* under the conditions of sufficient exploration and a decaying learning rate.

While traditional Q-learning is effective for small, discrete state-action spaces, it becomes impractical when dealing with large or continuous spaces due to the need to store and update a full Q-table. This limitation motivates the use of function approximation methods such as Deep Q-Networks (DQN), which extend Q-learning to high-dimensional state spaces using neural networks.

2.5.2 Deep Q-Network (DQN)

Deep Q-Network (DQN) is a widely adopted Deep Reinforcement Learning (DRL) algorithm that integrates Q-learning with deep neural networks. It was introduced by Mnih et al. [19] to address the limitations of traditional tabular Q-learning in high-dimensional state spaces, such as those found in video games or vision-based control environments. In DQN, a neural network is used to approximate the action-value function $Q(s,a)$.

DQN introduces two key innovations to stabilize training:

- **Experience Replay:** Transitions (s_t, a_t, r_t, s_{t+1}) are stored in a replay buffer. During training, mini-batches are sampled randomly to break correlations between sequential samples and improve sample efficiency.
- **Target Network:** A separate target network $\hat{Q}$ is maintained, whose weights θ^- are updated periodically with the weights of the main Q-network. This reduces oscillations during learning by providing stable target values.

The training process follows an ε-greedy policy for action selection, balancing exploration and exploitation. With probability ε, the agent selects a random action to explore the environment, while with probability $1-\varepsilon$, it chooses the action with the highest estimated Q-value, promoting exploitation of learned behavior. The main Q-network is updated by minimizing the mean squared error between the predicted Q-values and the target values derived from the target network. The full algorithm is summarized in Algorithm 1.

DQN's robustness and generalization capabilities have led to its adoption in a wide range of domains, including wireless communications, where it can be employed in problems such as dynamic spectrum access, adaptive beamforming, and adaptive modulation schemes. In this dissertation, DQN serves as the foundation for training the RL agent that controls the gNB's movement to maintain LoS with the user equipment under dynamic obstruction conditions.

Algorithm 1 Deep Q-Network (DQN) [17]

Input: the pixels and the game score
Output: Q action-value function (from which we obtain policy and select action)
Initialize replay memory D
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- \leftarrow \theta$
for episode = 1 to M **do**
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    for $t = 1$ to T **do**
        Following ε-greedy policy, select $a_t = \begin{cases} \text{a random action} & \text{with probability } \varepsilon \\ \arg\max_a Q(\phi(s_t), a; \theta) & \text{otherwise} \end{cases}$
        Execute action a_t in emulator and observe reward r_t and image x_{t+1}
        Set $s_{t+1} = \{s_t, a_t, x_{t+1}\}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
        `// Experience Replay`
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
        Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ w.r.t. θ
        `// Periodic update of target network`
        Every C steps reset $\hat{Q} = Q$, i.e., set $\theta^- = \theta$
    end for
end for

2.6 Related Work

Sensing and CV-aided wireless communications have received increasing attention recently, leading to significant research efforts. Multiple solutions presented in the literature leverage ML algorithms, especially DL networks, which rely on the availability of large, high-quality datasets to train the models effectively.

In [20], DeepSense6G, a large-scale real-world dataset, is presented. It includes data from different modalities, including RGB images, GPS, mmWave receive power, 2D/3D LiDAR, and radar measurements. This dataset is designed to support the development and training of ML models for wireless communications and sensing tasks. In [21], Vision-Wireless (ViWi) is proposed as a data-generation framework that creates synthetic datasets by simulating realistic communications and sensing environments. The framework generates a dataset containing four different scenarios and the associated data, including images, depth maps, signal departure/arrival angles, path gains, and user location. In [22], the Vision Objects for Millimeter and Terahertz Communications (VOMTC) dataset is introduced, containing 20,232 pairs of RGB and depth images associated with three object classes (person, mobile, laptop). The efficacy of this dataset is demonstrated through the development of a CV-based algorithm for beam management. For that purpose, the authors use a VOMTC-trained object detector to extract the position of the targets and generate a beam towards that position.

Among these, the VOMTC dataset is explicitly intended to support the training of computer vision models for beam selection and object localization. On the other hand, DeepSense6G and ViWi provide richer multimodal data that are more closely aligned with the type of dataset used in this dissertation, particularly in integrating visual and wireless information. However, both lack dynamic mobility in the gNB placement, a critical aspect for learning perception-driven mobility control policies. As such, they are unsuitable for this application, even if used to train a supervised learning model.

Regarding RAN configuration control employing vision-aided communication systems, beam management has received increasing attention in recent years. In [23], future beam indices are predicted based on previously observed beam indices and street-views images. The authors use ResNet, 3D ResNeXt, and a long short-term memory network for predictions. In [24], a framework implementing two DL models based on ResNet18 is proposed for beam selection and blockage prediction, leveraging images from RGB cameras at base stations and sub-6 GHz channels data. In [25], the authors propose a vision-aided beam focusing technique, using object detection to extract the position and depth images for distance estimation. The authors of [22] and [25] expand work in [3] by addressing more use cases related to SVWC, including beam management, proactive handover (referred to as cell association), and crowd estimation for random access. The experimental setup involved base stations equipped with two RGB cameras, enabling the extraction of semantic and spatial information from the environment. This information is processed using CV object detectors to identify and localize relevant targets. In [26], proactive handovers are also explored using an ML framework that relies on visual data captured by RGB cameras deployed at

the base stations. To train the DL algorithm, the authors leverage data generated using ViWi.

Despite significant research on vision-aided beam management and proactive handover solutions, the integration of these solutions with the O-RAN architecture remains underexplored, with only brief mentions in [3]. Regarding O-RAN deployments, [27] provides design insights on the integration of ML solutions for O-RAN closed-loop control, explaining how to embed these solutions into xApps using OpenRAN gym [28], an open toolbox for data-driven O-RAN experimentation. The authors demonstrate the potential of OpenRAN gym with two xApps managing scheduling policies and allocating resources to different BS network slices. In [29], the authors present an xApp-based solution that addresses dynamic resource allocation, leveraging an ML-based data-driven approach. Additionally, [30] presents a tutorial for xApp development, providing a guide that covers the framework from theoretical foundations to practical implementations within the O-RAN ecosystem. It explores the design and configuration of xApps while explaining the steps involved in deploying and managing xApps on a Near-RT RIC. The works in [27], [29], and [30] are not directly related to SVWC; however, they provide important references and guidelines for understanding the development of intelligent solutions within the O-RAN ecosystem, including xApp design and ML integration.

From the literature, it is possible to conclude that sensing and vision-aided wireless communications are emerging as key topics in network management, with solutions addressing use cases such as beam management [3, 22, 23, 24, 25, 31] and proactive handovers [3, 26]. However, these works primarily focus on static base station configurations and do not address the dynamic control of gNB mobility.

With respect to the concept of mobile gNBs, [32] presents a mobility management xApp integrated into an O-RAN-compliant deployment. While this work represents a step toward autonomous base station repositioning, the gNB was limited to switching between only two predefined positions and relied exclusively on RF-based data for mobility control, without incorporating any sensing or vision-based information.

To the best of our knowledge, no solutions explore the positioning of vision-aided gNBs to prevent LoS blockage while integrating with the O-RAN architecture, opening opportunities for novel contributions to vision-aided O-RAN networks.

Chapter 3

System Overview, Design, and Implementation

The main goal of this dissertation was to design and implement a perception-driven O-RAN framework capable of performing intelligent control of a mobile gNB to maintain reliable wireless connectivity with a UE. The envisioned system follows a standalone (SA) 5G architecture and integrates CV and RL to enable the gNB to autonomously reposition itself. This mobility control is guided by real-time multimodal perception of the environment, allowing the network to proactively adapt to obstacles and dynamic scenarios that may compromise the LoS and degrade the communication link to the UE.

This chapter presents the specifications, design decisions, and overall implementation of the proposed system. Section 3.1 focuses on the system overview and proposed architecture, outlining the core components and their interactions. Sections 3.2, 3.3, 3.4, and 3.5 detail the internal design of each module, highlighting their functionalities and integration within the framework.

3.1 System Overview

3.1.1 Physical Environment and Modeling Assumptions

Figure 3.1 illustrates a top-down view of the CONVERGE Chamber, which serves as the environment where the system operates. The chamber is a bounded 3D space with width W, length L, and height H, and its coordinate system is defined such that the origin is placed at the bottom-left-front corner. All object positions and motion are expressed in Cartesian coordinates (x, y, z), although for simplicity, all entities are constrained to move only within the x-y plane, with constant z-coordinate values.

In this scenario, the gNB and LIS are equipped with RGB-D cameras, represented in red, which observe the chamber from different positions. While the gNB is mobile, it is constrained to move only along the x-axis, maintaining fixed y and z coordinates during operation. The LIS remains static throughout the experiment and is assumed to have full visibility over all objects in the room. It is assumed that each camera provides semantic and positional data for visible

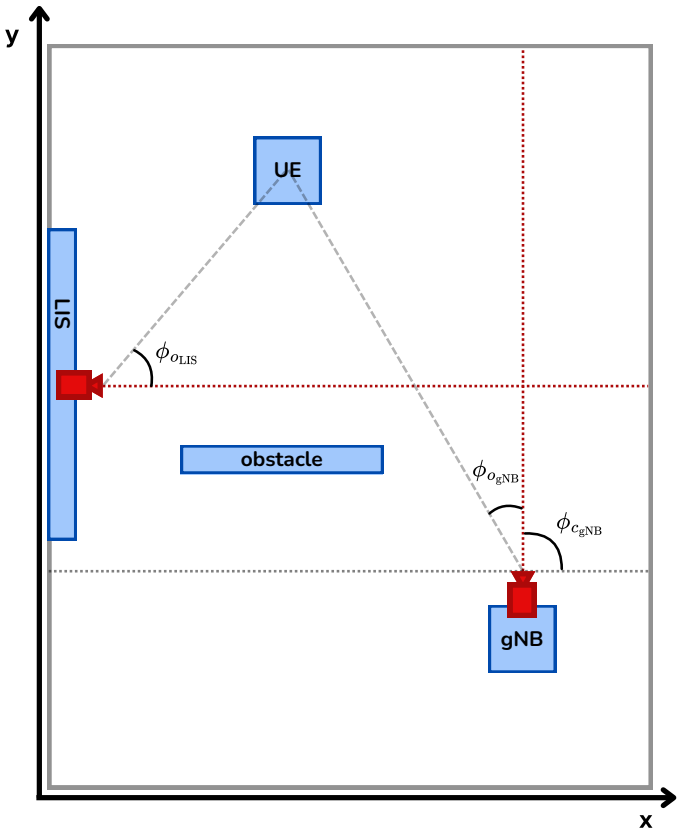


Figure 3.1: Top-view layout of the CONVERGE Chamber with relevant vectors and angles.

objects, including the class, bounding box, and distance of detected objects, from which angles θ_o (elevation) and ϕ_o (azimuth) can be computed using CV modules.

The positions of detected objects are inferred from polar to Cartesian coordinates using camera-centric observations:

$$x_o = x_c + r_o \cdot \cos(\phi_c + \phi_o) \cdot \cos(\theta_o), \qquad y_o = y_c + r_o \cdot \sin(\phi_c + \phi_o) \cdot \cos(\theta_o) \tag{3.1}$$

where (x_c, y_c) and ϕ_c are the position and orientation of the camera, θ_o is the elevation angle, ϕ_o is the azimuth angle, and r_o is the range to the object.

To simplify the simulation and control logic, the following assumptions are adopted:

- Only one UE and one obstacle are present in the environment at any time.
- The gNB moves only along the x-axis.
- The LIS is static and assumed to have complete, unobstructed visibility of the chamber at all times.
- All positional updates are synchronous and based on current time-stamped detections.

These simplifications facilitate reproducibility and fast iteration during agent training, while keeping the system modular for future enhancements involving dynamic 3D motion, multi-camera fusion, or occlusion-aware perception.

3.1.2 Proposed Architecture

The proposed architecture[1] of the system is illustrated in Figure 3.2. It represents a complete system built on top of the CONVERGE architecture, extending its modular and service-oriented design with custom integrations for vision-based sensing, real-time inference, and RAN reconfiguration through the O-RAN control loop. The system combines standardized interfaces and modules defined by O-RAN with custom components tailored for multimodal perception and control, forming a cohesive pipeline for real-time, vision-guided RAN configuration.

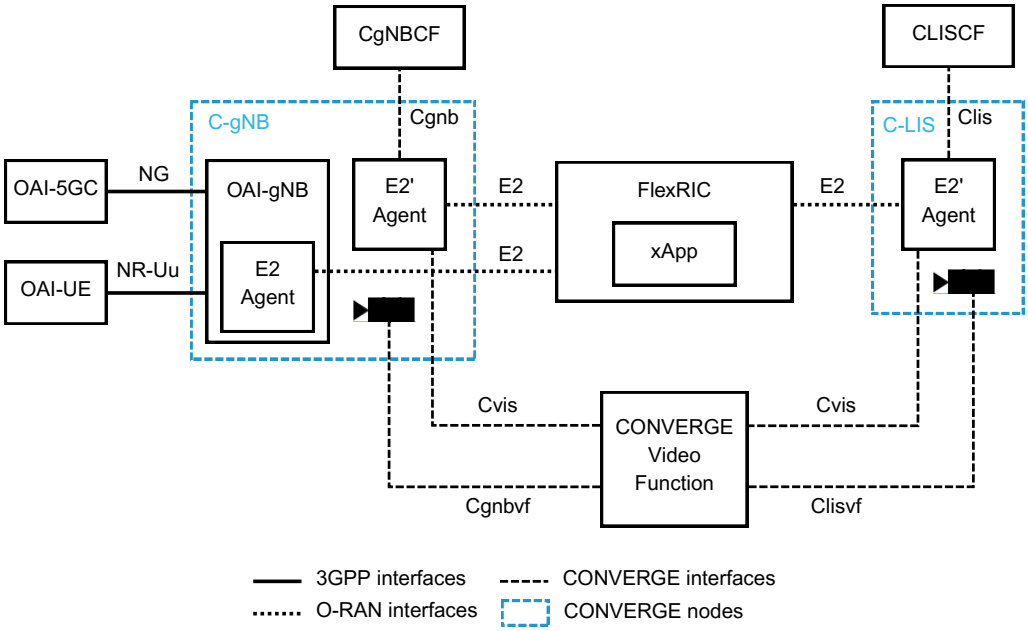


Figure 3.2: Proposed architecture.

At the core of the proposed architecture is the **nearRT-RIC** (FlexRIC), interfacing with network nodes via the standardized **E2 interface**. In this dissertation, we decided to extend the E2 interface's role beyond traditional RAN functions, leveraging it as the sole communications channel for all data required by the xApp. This architectural decision aimed to simplify the development of RAN management solutions, allowing developers to focus exclusively on developing a single xApp. All necessary information, ranging from RF metrics to vision-based semantic data, is made accessible through E2 message subscriptions, promoting modularity, scalability, and seamless integration within the O-RAN ecosystem.

To enable this extended use of the E2 interface, the architecture required the definition of new RAN functions (messages) capable of transporting both spatial and CV semantic information.

[1] While this is not the final implementation architecture, it remains highly relevant, as the core nodes, interfaces, and interactions described are preserved in the final system design.

This included the design of corresponding Service Models, tailored to encapsulate data such as object positions, movement vectors, and scene understanding features extracted via CV. These custom SMs are handled by specialized E2 agents–hereafter referred to as **E2′ agents**–serving as the bridge between the CONVERGE modules and the nearRT-RIC. The E2′ agents are intended to manage the periodic generation and transmission of custom messages, ensuring that all relevant multimodal data are delivered to the RIC via standardized E2 procedures.

The **CONVERGE gNB** (C-gNB), as the main radio node, hosts the **OAI-gNB**, responsible for handling traditional radio access functions. Following the 3GPP specifications, the OAI-gNB interfaces with the **OAI-5GC** through the NG interface for control- and user-plane signaling and with the **OAI-UE** over the NR-Uu interface. The C-gNB hosts both the OAI standard E2 agent and a custom **E2′ agent**. The former manages traditional RAN metrics and control messages in accordance with O-RAN procedures, while the latter handles the transmission of spatial and vision-based semantic data derived from the CONVERGE modules. Specifically, the E2′ agent, hosted on the C-gNB, obtains position and velocity information from **CgNBCF** through the **Cgnb** interface, while the video-semantic information is acquired via the Cvis interface, which connects directly to the CONVERGE Video Function. Both data streams are forwarded to the nearRT-RIC via the standardized E2 interface. Apart from reporting information to the nearRT-RIC, the E2′ agent is also responsible for forwarding position control coordinates issued from the xApp to the **CgNBCF** via the **Cgnb** interface.

The **CONVERGE Video Function**, connected at both the C-gNB and C-LIS nodes, processes raw visual data captured by integrated RGB-D cameras via the **Cgnbvf** and **Clisvf** interfaces. It performs object detection and extracts semantic features, including object class, bounding box coordinates and dimensions, and polar coordinates such as relative distance and orientation angle with respect to the local vision node. These features are encoded into structured formats and transmitted to the corresponding E2′ agents via the **Cvis** interface, an IP-based communication link that exchanges JSON-formatted messages between the CONVERGE Video Function and its host agent.

The **CONVERGE LIS** (C-LIS), while modeled as a reconfigurable intelligent surface, was primarily included to serve as a reference vision node. It is assumed to have full visibility of the CONVERGE Chamber, thereby providing ground truth visual data for all objects in the environment. Like the gNB, it hosts its own E2′ agent. This agent interfaces with the CONVERGE Video Function via the Cvis interface to receive semantic data extracted from the integrated RGB-D camera, and with the **CLISCF** via the **Clis** interface to obtain the current position of the LIS. Both vision-derived and spatial information are reported to the nearRT-RIC through the standardized E2 interface. This setup allows the LIS to continuously supply comprehensive semantic information, supporting validation and enhanced situational awareness.

Finally, the **xApp**, deployed within the nearRT-RIC, acts as the central decision-making entity. It subscribes to E2 messages from both the gNB and LIS, aggregating multimodal data to perform intelligent RAN control. Based on this information, the xApp issues control messages back to the gNB's E2′ agent via the E2 interface, completing a full-loop control mechanism that enables

real-time reconfiguration of the RAN. The architecture's reliance on the O-RAN interfaces ensures interoperability with existing O-RAN deployments, while the CONVERGE interfaces (Cgnb, Clis, Cgnbvf, Cliscf, Cvis) provide the necessary extensions for perception-aided control. The decision logic is driven by a trained DRL model embedded in the xApp, which interprets the incoming metrics and semantic data to determine optimal mobility strategies in real time.

In summary, the proposed testbed was designed to leverage FlexRIC and a custom xApp to orchestrate real-time, perception-driven RAN management. The integration of CONVERGE modules, E2′ agents, and standardized O-RAN and 3GPP interfaces aimed to deliver a scalable and modular system architecture, converging vision intelligence with wireless control for future network deployments.

3.2 Multimodal O-RAN Framework

While the system overview establishes the architecture and high-level integration of components, this section delves into the design and implementation of the multimodal O-RAN framework developed module. It includes the novel Service Models and E2′ agents for custom RAN functions.

The following sections describe the design rationale, implementation details, and integration strategies for each module in the system.

3.2.1 Custom Service Models: POS and VIS

To support perception-driven RAN control within the O-RAN framework, two custom Service Models were developed and integrated into the FlexRIC platform: the **POS** (position) SM and the **VIS** (vision) SM. These models extend the FlexRIC control and monitoring capabilities by introducing new message types that capture spatial dynamics and visual semantic understanding of the environment, respectively.

Each SM was designed following the FlexRIC architecture, which requires implementing a set of functions to handle message encoding/decoding, subscription management, and inter-process communication via the E2 interface. These functions were registered within the FlexRIC agent and RIC libraries, enabling full support for E2-based control loops. This implementation process followed the official service model development guide provided by the FlexRIC project [33].

POS Service Model – ID 101

The POS SM provides both **monitoring** and **control** capabilities for the spatial state of the mobile network nodes. It enables the RIC to receive position and motion information from distributed network entities (e.g., gNB, LIS), and to issue movement commands back to the mobile gNB, thereby closing the control loop for mobility control.

This service model is composed of two main message types:

- **Indication Message** (`pos_ind_msg_t`): sent from the E2′ agents to the RIC, containing the current positions, velocities, and orientation of the sending node. The structure of

each reported entity is described in Table 3.1, and the overall message format is shown in Table 3.2.

- **Control Message** (`pos_ctrl_msg_t`): sent from the RIC xApp to the gNB's E2′ agent, defining a target position to move the gNB. The control message structure is provided in Table 3.3.

Table 3.1: POS Entry Fields (`pos_stats_t`).

DType	Name	Description
`int16_t`	`type`	Entity type: 0 = gNB, 1 = UE, 2 = LIS, 3 = obstacle
`int32_t`	`x, y, z`	Cartesian position (cm)
`int32_t`	`v_x, v_y, v_z`	Velocity vector components (cm/s)
`int32_t`	`theta, phi`	Elevation and azimuth angles (rad $\times$ 100)

Table 3.2: POS Indication Message Structure (`pos_ind_msg_t`).

DType	Name	Description
`pos_stats_t*`	`pos_stats`	Pointer to array of POS data entries
`uint32_t`	`len`	Number of entries in the array
`int64_t`	`tstamp`	Timestamp of the report (µs)

Table 3.3: POS Control Message Structure (`pos_ctrl_msg_t`).

DType	Name	Description
`int32_t`	`x, y, z`	Target Cartesian coordinates (cm)
`int64_t`	`tstamp`	Timestamp of the control command (µs)

The POS SM was registered in FlexRIC with ID 101 and integrated into both the nearRT-RIC and custom E2′ agents.

VIS Service Model – ID 102

The VIS SM is designed to provide semantic scene understanding from CV modules deployed at the network edge. It provides the nearRT-RIC with real-time information about detected objects in the environment, including their class, location, orientation, and distance relative to the camera.

This service model consists of a single message type:

- **Indication Message** (`vis_ind_msg_t`): sent from the E2′ agents to the RIC, reporting all objects detected in the current frame by the vision module. The structure of each detected object is detailed in Table 3.4, and the full message format is shown in Table 3.5.

Table 3.4: VIS Entry Fields (`vis_stats_t`).

DType	Name	Description
`int16_t`	`id`	Unique ID of the tracked object
`int32_t`	`cls`	Class of the object
`int32_t`	`bb_x`, `bb_y`	Centroid coordinates of the bounding box (px)
`int32_t`	`bb_w`, `bb_h`	Width and height of the bounding box (px)
`int32_t`	`theta`, `phi`	Elevation and azimuth angles (rad $\times$ 100)
`int32_t`	`r`	Distance of the object from the camera (cm)

Table 3.5: VIS Indication Message Structure (`vis_ind_msg_t`).

DType	Name	Description
`vis_stats_t*`	`vis_stats`	Pointer to array of detected object data entries
`uint32_t`	`len`	Number of objects detected in the frame
`int64_t`	`tstamp`	Timestamp of the report (µs)

The VIS SM was registered in FlexRIC with ID 102 and implemented in both the nearRT-RIC and E2′ agents.

3.2.2 E2′ Agents

The E2′ agents serve as specialized communication bridges between the nearRT-RIC and external perception and positioning nodes. Implemented as modified versions of FlexRIC agents, they support the **POS** and **VIS** service models, enabling periodic reporting of spatial and vision-based information, and accepting RAN control commands. Each agent instance is configured for either the gNB or the LIS node and initialized with a set of configuration parameters including Public Land Mobile Network (PLMN) identifiers and node type.

Each E2′ agent was designed to interface with their respective CONVERGE Control Functions via the Cgnb and Clis interfaces, exchanging JSON-formatted POS data, while simultaneously communicating with the CONVERGE Video Function via the Cvis interface to obtain VIS information derived from semantic scene analysis. The indication files are periodically parsed and loaded into internal buffers by dedicated threads. The agent's internal I/O interface, which manages its interaction with FlexRIC, binds callback functions for reading indication messages and handling control commands associated with the registered Service Models:

- `read_pos_sm()` parses and returns a `pos_ind_msg_t` message from the latest POS data, enabling real-time awareness of the agent's position.
- `read_vis_sm()` parses and returns a `vis_ind_msg_t` message from the latest VIS data, enabling object detection reports to be forwarded to the RIC.
- `write_ctrl_pos_sm()` handles incoming `pos_ctrl_msg_t` control commands from the RIC xApp, logging the instructions and exporting the new target coordinates to a control JSON file for external actuation.

The read functions populate internal message structures using thread-safe mechanisms, passing them to the RIC through FlexRIC's standard SM interface. Figure 3.3 illustrates the internal data flow of the gNB E2′ agent and its interaction with FlexRIC and the local perception/control files. The LIS E2′ agent, illustrated in Figure 3.4, functions similarly, but without implementing a control mechanism, as it operates solely in a perception-reporting role.

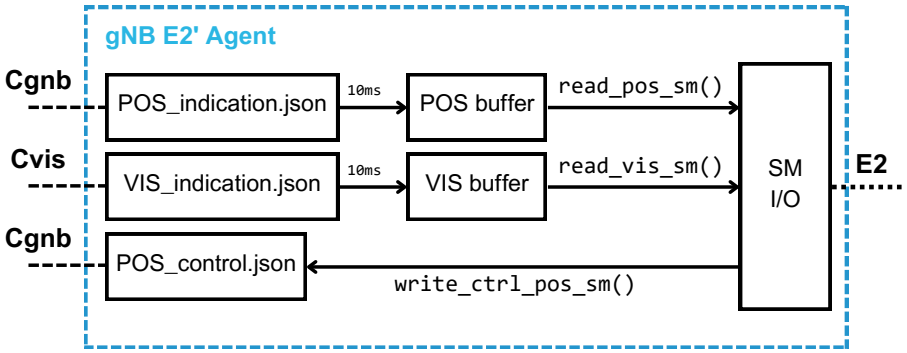


Figure 3.3: gNB E2′ agent. Data ingestion and SM I/O.

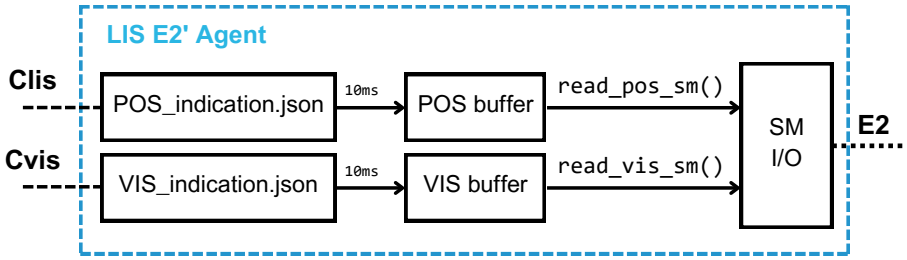


Figure 3.4: LIS E2′ agent. Data ingestion and SM I/O.

At startup, the agent configures its PLMN identifiers, initializes its I/O interface, and starts the FlexRIC framework via `init_agent_api()`. The application then enters a polling loop, maintaining E2 communications while periodically updating the internal buffers with new data.

3.3 CC-SIM: Converge Chamber Simulator

While the proposed multimodal O-RAN framework was designed with real-world deployment in mind, the full experimental setup was not functional during this dissertation, limiting access to real-time data required to train and validate the RL agent. As a result, it became necessary to create a simulation environment that can emulate the behavior of the gNB, UE, and dynamic obstacles in a controlled 3D space. This led to the development of the **CONVERGE Chamber Simulator (CC-SIM)**, a custom environment built to simulate the data inputs/outputs expected by the CONVERGE interfaces, specifically position data and vision-based semantic information. It replaces the physical gNB and LIS nodes as the source of information exchanged with the E2′ agents, while replicating expected interface behaviors and supporting the iterative training of the

mobility control agent using RL. Beyond its role as a workaround for unavailable hardware, CC-SIM proved to be an essential tool for rapid prototyping, debugging, and performance evaluation of the entire closed-loop control framework, including the training and validation of the RL agent under diverse and reproducible mobility scenarios.

The system architecture used in this dissertation, depicted in Figure 3.5, shows the transition from a physical testbed to a simulation-driven framework. It emphasizes the simulator as the core execution environment while preserving the modular structure compatible with future integration into a real experimental testbed.

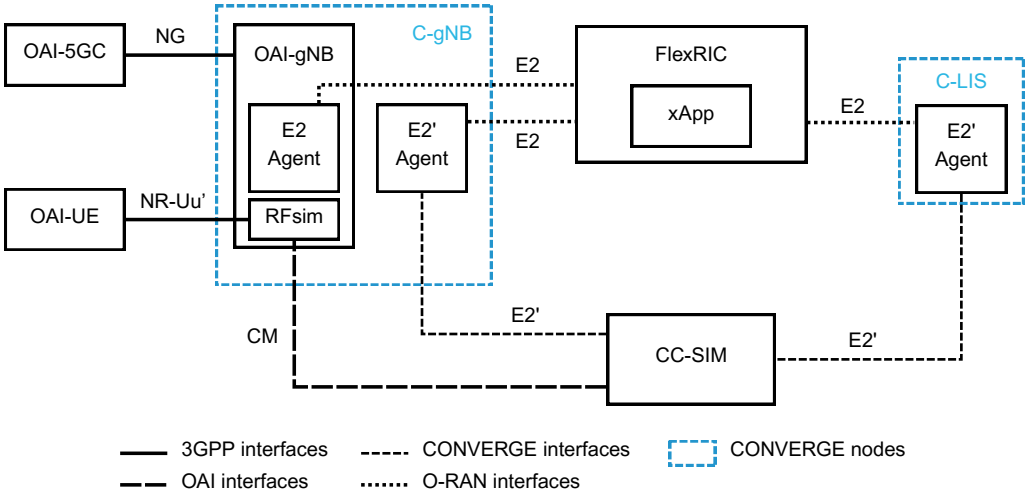


Figure 3.5: Simulation-driven system architecture.

In this system, E2′ agents communicate with CC-SIM through the **E2′** interface. This interface replaces the previously defined **Cgnb**, **Clis**, and **Cvis** interfaces, which were all designed for JSON-based message exchange over IP. The E2′ interface remains fully compatible with these formats while unifying the communication pipeline. It provides both position-related and vision-semantic data to the respective E2′ agents based on simulated data, while accepting control messages from the gNB E2′ agent.

In this architecture, communication between the gNB and UE via the NR-Uu interface is emulated using the **OAI RF Simulator (RFsim)** [34], which provides a controlled environment for modeling over-the-air transmission without physical RF hardware. The **channel simulation** [35] feature within the RFsim allows for real-time modification of channel conditions, such as path loss, noise power, and fading, based on realistic propagation models. This is important as it enables us to dynamically configure the parameters of the simulated channel model based on the CC-SIM environment state. To achieve this, the CC-SIM connects to the RFsim via the **CM** interface[2], sending periodic updates of path loss conditions based on the current positions of simulated nodes. This mechanism allows the system to replicate the effects of environmental changes on RF

[2]While not an official designation, the term *CM interface* is used here to refer to the Telnet-based control mechanism implemented provided by the OAI RF Simulator, which enables real-time channel model configuration from external controllers.

propagation, such as LoS obstruction, thereby enabling realistic testing and validation in a fully closed-loop setting.

The remaining components and interactions within the system architecture remain consistent with the description provided in Section 3.1.2, including the role of the nearRT-RIC, xApp decision logic, standardized O-RAN interfaces, and the integration of the OAI 5G standalone network. These elements continue to support a modular, perception-driven control loop for intelligent RAN reconfiguration.

As noted, the CC-SIM is a custom-built 3D simulation environment designed to replicate the spatial, visual, and RF behavior of the CONVERGE chamber. Implemented in Python, it models the chamber's geometry, object mobility, LoS status, and channel condition dynamics. To provide a clearer understanding of its internal structure, the main components and functional layers of CC-SIM are presented in the following subsections.

3.3.1 Object3D

At the core of the simulation lies the `Object3D` class, which serves as the elemental unit representing physical entities such as the gNB, LIS, UE, and obstacles. Each object instance tracks position, velocity, and dimension attributes and handles object dynamics such as velocity updates, boundary enforcement, random movement, or constant motion propagation. `Object3D` instances can export positional (POS) and visual semantic information (VIS) through JSON files to be consumed by the E2′ agents, and their position can be externally updated by importing JSON-based control inputs. An overview of the primary attributes associated with `Object3D` is provided in Table 3.6.

Table 3.6: Attributes of `Object3D` Class.

DType	Attribute	Description
`list`	`init_center`	Initial object position – x, y, z – used to reset the Environment
`list`	`prev_center`	Previous object position – x, y, z – used for velocity calculation
`float`	`x, y, z`	Current object position (m)
`float`	`v_x, v_y, v_z`	Velocity components along each axis in (m/s)
`float`	`w, l, h`	Physical dimensions: width, length, and height (m)
`float`	`theta, phi`	Elevation and azimuth angle of object (rad)
`float`	`attenuation`	Object-specific signal attenuation, used for obstacles (dB)
`string`	`label`	Entity label (e.g., gNB, UE, LIS, Obs)

3.3.2 GUI

CC-SIM features a **graphical user interface (GUI)** that offers real-time 3D visualization of the chamber and all moving entities. This interface enables researchers to observe the behavior of the gNB as it responds to dynamic scenarios under the guidance of external controls. The GUI provides a clear depiction of LoS paths, obstacles, and relative movement, helping validate both the spatial awareness and reactivity of the trained model.

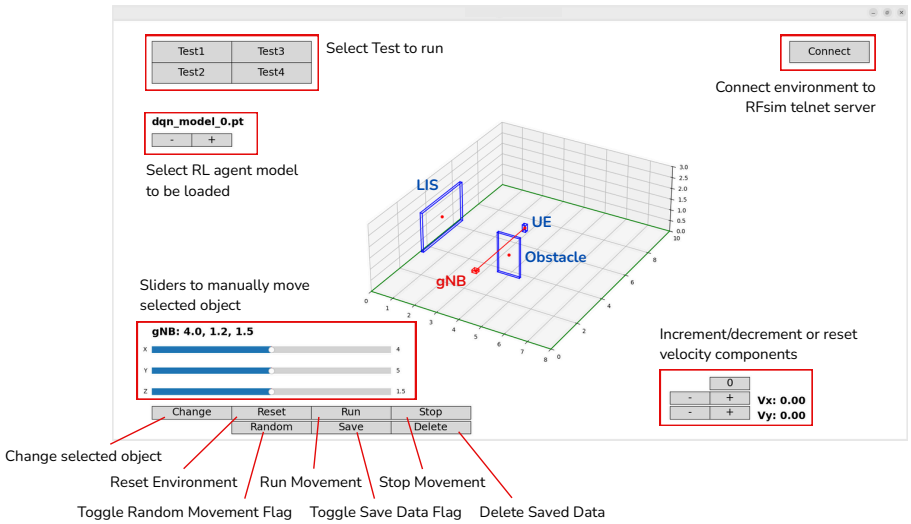


Figure 3.6: CC-SIM GUI. This GUI includes features for model selection, manual object movement sliders, and velocity adjustment options. It also provides action buttons (Run/Stop/Random), data management functions (Save/Delete), and environment controls (Reset/Random Movement). Additionally, it supports telnet server connectivity for the RF simulator configuration.

The GUI's 3D rendering is implemented using Matplotlib [36], a choice driven primarily by its simplicity and rapid prototyping advantages during early development. Matplotlib's 3D axes enable dynamic camera adjustments and path tracing, which are helpful for debugging movement and focusing on different perspectives of the scene. Additionally, Matplotlib provides built-in widget support, enabling interactive control over environment parameters such as object positions and scenario management, as illustrated in Figure 3.6. While its straightforward API accelerated initial implementation, the library's limitations for high frame rate rendering became apparent as the simulator evolved. For future iterations, migrating to a dedicated 3D engine is under consideration to address these real-time constraints.

3.3.3 Environment

Designed with Reinforcement Learning in mind, the core system logic is encapsulated within the `Environment` class. This class orchestrates the simulation flow and can be operated in three different modes: **training**, **simulation-based testing**, and **live testing** with FlexRIC integration. These modes facilitate a smooth workflow from offline training and validation to real-time inference and control using live data streams.

Training Mode

In **training mode**, the CC-SIM is used as a standalone RL environment. The `Environment` class was designed to be compatible with the OpenAI Gym API specification [37], implementing essential methods such as `reset()` and `step()` to support standardized agent-environment

interaction. This design facilitates seamless integration with RL training pipelines and promotes reproducibility across different learning scenarios.

The agent interacts with the system by calling the `step()` function with a given action a, which updates the gNB's velocity and position based on the provided action and progresses the simulation forward by one timestep. At every step, the function applies scenario-specific behavior to the UE and obstacles (e.g., constant velocity motion or synchronized patterns) as determined by the current scenario identifier, further explained in Section 3.4. At each step, the environment updates the LoS status and computes a scalar reward r based on the communication link quality. It then returns a tuple consisting of the next observation, the reward, a boolean flag indicating if the episode is complete, and a string summarizing the returned information, as shown in Table 3.7. These attributes are used by the RL agent to evaluate the outcomes of its actions; their composition and significance are further detailed in Section 3.4. The GUI can be simultaneously used in this mode to evaluate the training method qualitatively. It helps verify whether scenarios are being correctly applied and allows researchers to monitor the behavior of the RL agent throughout training. By visually observing the gNB's motion and interactions with the environment, developers can diagnose convergence issues, confirm policy responsiveness, and make necessary adjustments to the reward structure.

Table 3.7: Return attributes of `step()` method.

DType	Attribute	Description
`list`	`state`	The next observation vector s_{t+1}
`float`	`reward`	The reward r_t of taking the action a_t in state s_t — $R(s_t, a_t)$
`bool`	`end`	Flag indicating if the episode is complete
`string`	`info`	A string summarizing the returned values (debugging purposes)

Simulation vs Live Testing Mode

In simulation and live testing modes, the GUI plays a crucial role by allowing researchers to directly interact with the environment and assess the responsiveness of the gNB controlled by the trained DRL model.

During **simulation-based** testing, the DRL model runs in the same process as the environment, and the gNB's position is updated directly, allowing for tightly coupled agent-environment interaction. The input state vector used by the model is extracted internally via the environment's own `get_state()` method, providing access to relevant state information without intermediate data serialization. This makes the simulation mode a valuable tool as it provides a rapid and flexible empirical validation environment, enabling researchers to assess the performance of trained models across diverse, controlled scenarios without the need to deploy the full system or rely on hardware components.

In contrast, during **live testing**, the trained DRL model is deployed as part of an xApp running within FlexRIC. Here, control decisions are issued in the form of JSON-based target positions generated by the E2′ agent, as described in Section 3.2.2. The CC-SIM environment periodically

consumes these inputs and applies them to update the gNB's position. Simultaneously, it updates the UE and obstacle states and recalculates the channel conditions. In this mode, CC-SIM acts as a virtual replacement for the physical CONVERGE nodes, exporting **POS** and **VIS** information for both the gNB and LIS via JSON files. POS data is extracted directly from the corresponding `Object3D` instance attributes, including absolute position and velocity vectors, while VIS data is computed by projecting the relative positions of visible entities into polar coordinates using the inverse of the transformation equations 3.1, defined in Section 3.1.1. The `id` and `cls` fields are assigned according to the entity type conventions listed in Table 3.1, and placeholder values are used to populate bounding box dimensions as they are not used for this specific application.

At every simulation loop iteration, the path loss between the gNB and UE is recalculated based on their relative positions and current LoS status. The value is computed using the formula:

$$\mathrm{PL}_{\mathrm{dB}} = 20\log_{10}(d) + A_{\mathrm{obs}} \cdot L_{\mathrm{status}}$$

where d is the Euclidean distance between the gNB and UE, A_{obs} is the attenuation of the obstacle, and L_{status} is a binary flag set to 1 if the LoS is obstructed, and 0 otherwise. This path loss value is then transmitted to RFsim via the CM interface, emulating dynamic RF conditions in real time.

The main loop responsible for executing this behavior is implemented in a background thread, as described in Algorithm 2.

Algorithm 2 Main CC-SIM Control Loop Thread

```
while True do
    if simulation is running then
        Import target position JSON and update gNB position
        Move UE and obstacle based on their current velocities and cycle period
    end if
    Recalculate LoS status and path loss
    Export POS and VIS JSONs for gNB and LIS
    if CM interface is active then
        Send path loss update to RFsim via telnet
    end if
    Sleep until next wake-up time
end while
```

3.4 RL for gNB Mobility Control

To control the gNB positioning, we employed a DNN to learn optimal movement strategies that allow preserving LoS. For this purpose, we chose Reinforcement Learning as the learning paradigm, given its ability to handle sequential decision-making problems where the environment is dynamic and partially observable. RL allows the agent to learn control strategies through trial-and-error interaction, making it well-suited for adapting gNB placement in response to moving UEs and obstacles under uncertain and time-varying RF conditions.

The RL-based control agent was designed to learn how to dynamically reposition the gNB in order to maintain LoS with the UE while adapting to potential obstacle movements. This section focuses on the rationale behind the state vector, action space, and reward function design, as well as the neural network architecture and training methodology.

3.4.1 State Vector, Action Space, and Reward Function Design

For this intelligent mobility application, we drew inspiration from human use of vision, primarily to perceive spatial and depth relationships within the environment. For example, if a person notices an obstacle moving toward a line of sight between them and a target, they intuitively reposition to maintain visibility, relying on positional awareness, motion cues, and occlusion reasoning. Translating this into a machine learning setting, we assumed that geometric input allows the RL agent to infer spatial relationships and make informed movement decisions. Therefore, the state vector was designed to capture the relative positioning and dynamics of key entities in the CONVERGE chamber, as presented in Table 3.8.

Table 3.8: State Vector Features.

Feature	Description
x_{gnb}	Absolute x-position of the gNB (m)
$x_{\text{gnb-ue}}$, $y_{\text{gnb-ue}}$	Relative x and y displacement from gNB to UE (m)
$x_{\text{gnb-obs}}$, $y_{\text{gnb-obs}}$	Relative x and y displacement from gNB to obstacle (m)
$v_{x_{\text{gnb}}}$	Velocity of the gNB in the x-axis (m/s)
$v_{x_{\text{ue}}}$, $v_{y_{\text{ue}}}$	Velocity of the UE in x and y directions (m/s)
$v_{x_{\text{obs}}}$, $v_{y_{\text{obs}}}$	Velocity of the obstacle in x and y directions (m/s)
L_{status}	Boolean indicating LoS status (0 if clear, 1 if obstructed)

The state s is represented by an 11-dimensional vector that includes the current x-position and velocity of the gNB, and two relative position vectors: one pointing from the gNB to the UE and another from the gNB to the obstacle (both in the x-y plane). It also includes the x and y velocity components of the UE and the obstacle. Lastly, a binary flag indicates whether the UE is currently in LoS with the gNB or if the signal is obstructed by the obstacle. This boolean helps the agent learn to prefer states where LoS obstruction is avoided.

To ensure smooth and realistic control behavior, the action space, presented in Table 3.9, was defined in terms of velocity increments rather than absolute position changes. This avoids abrupt gNB displacements and allows the policy to fine-tune its motion using gradual acceleration or deceleration along the x-axis. Without this design, the controller is prone to jerky movements or overshooting, impairing its ability to respond effectively in dynamic scenarios.

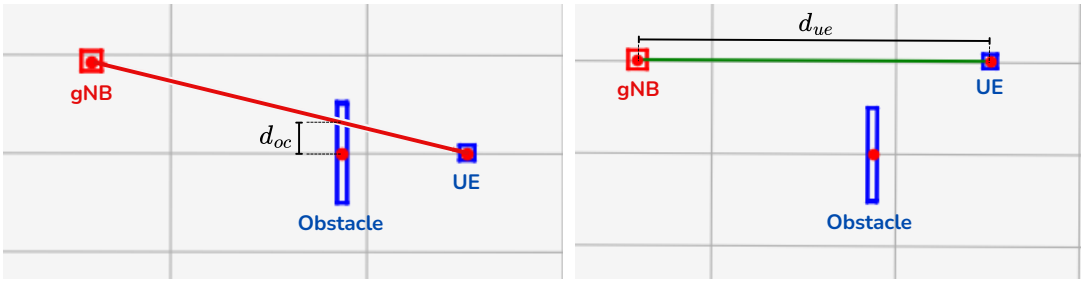
Table 3.9: Action Space.

Action	Description
0	Maintain current velocity
1	Increment velocity by δ
2	Decrement velocity by δ

The reward function was designed to guide the agent toward behaviors that promote LoS preservation while considering mobility constraints. When LoS is blocked, the agent receives a penalty in the range $[-1,0[$ scaled according to the distance between the intersection point of the LoS and the obstacle's surface and the obstacle's center, d_{oc}, thereby encouraging the agent to initiate avoidance maneuvers earlier when obstruction is imminent or to move away from the obstacle if obstruction exists. When the path is clear, the goal is to promote high radio link quality by minimizing path loss, which translates into reducing the distance between the gNB and the UE. In this case, the agent is rewarded based on the inverse of the distance to the UE, promoting proximity; a mapping function is used to scale this inverse proportion into a normalized value within the range $[0,1]$. To emphasize the influence of distance in the reward signal, the normalized value is then squared, sharpening the gradient for closer positions, and further scaled by a constant factor k to modulate its impact. The reward associated with taking action a_t in state s_t is defined by the following function:

$$R(s_t,a_t) = \begin{cases} -1+\overline{d_{\text{oc}}}, & \text{if LoS obstructed} \\ k\cdot[\text{map}(d_{\text{ue}})]^2, & \text{otherwise} \end{cases} \tag{3.2}$$

Where $\overline{d_{\text{oc}}}$ is the normalized value of the distance between the obstacle's center and the LoS intersection point, k is the scaling factor and $\text{map}(d_{\text{ue}})$ represents a scaled inverse mapping of the distance between the gNB and UE. Both $\overline{d_{\text{oc}}}$ and d_{ue} are computed based on the resulting state s_{t+1}. For a clearer understanding, Figure 3.7 visually illustrates the variables involved in the reward function.



(a) (b)

Figure 3.7: Variables used in reward function, with LoS obstructed (a) and LoS clear (b).

3.4.2 DNN Architecture and Training Methodology

The Q-network architecture used to approximate the action-value function $Q(s,a)$ is designed as a fully connected feedforward neural network tailored for tabular discrete action spaces. It consists of an input layer that receives an 11-dimensional input vector, as depicted in Figure 3.8. This is followed by three fully connected hidden layers: the first with 64 neurons, the second with 128 neurons, and the third with 64 neurons, all employing ReLU activation functions. The final output layer comprises 3 neurons, each representing the estimated Q-value for one of the available discrete actions.

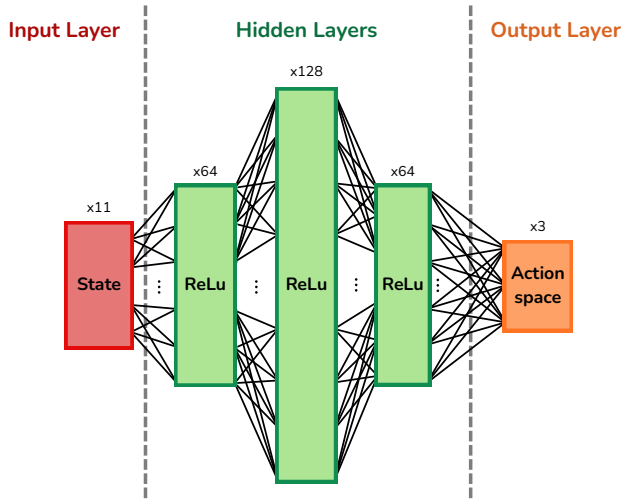


Figure 3.8: RL Agent's DNN structure.

The agent is trained using the DQN algorithm, previously described in Algorithm 1, and the implementation approach is based on the PyTorch tutorial provided by Zhou [38]. Each training episode is composed of a sequential composition of four predefined mobility scenarios, each lasting for a fixed number of steps. This episodic formulation allows the agent to experience increasing complexity and diversity within a single episode, while preserving structured transitions between behaviors.

The obstacle and UE are constrained to move only along the x-axis to reduce the training complexity. The four mobility scenarios are defined as follows:

- **Scenario A:** A static obstacle is placed in front of the UE. This serves as a baseline configuration with persistent LoS obstruction.
- **Scenario B:** The obstacle begins moving at a constant velocity to the right. Upon reaching a lateral boundary, it reverses direction and continues bouncing between the chamber walls, while UE remains stationary.
- **Scenario C:** The obstacle stops moving, and the UE starts to move in the same manner as the obstacle in Scenario B.
- **Scenario D:** Both the UE and the obstacle move independently, bouncing between the walls at different rates due to their varying dimensions.

In all scenarios, the gNB is controlled by the RL agent. As the obstacle and UE have distinct sizes, their asynchronous bouncing generates diverse spatial configurations throughout the episode. This results in varying LoS conditions, including situations where both objects move in the same or opposite directions. Such variability enriches the agent's learning experience and fosters more robust policy development. Each scenario transition occurs after a predefined step count, ensuring repeatable and interpretable progression within each episode.

Table 3.10 outlines the main training parameters and episode configuration, while Table 3.11 describes the utilized training hyperparameters. The system is designed to perform control updates with a timing precision of 200 ms, which aligns with the near-RT constraints of the control loop.

The **maximum gNB velocity** is set to 1 m/s, reflecting typical specifications found in mobile robotic platform datasheets [39]. Through empirical testing, it was observed that allowing the agent to increase its velocity over at least three consecutive steps was necessary to avoid erratic behavior caused by frequent direction changes during the exploration phase. Without this capability, the agent failed to stabilize and converge toward meaningful strategies for LoS preservation. To support this requirement while maintaining fine-grained control, a **velocity step** (δ) of 0.35 m/s was defined, enabling the agent to reach maximum speed within 3 steps in less than a second.

The **UE and obstacle velocities** are set to 0.6 m/s, offering a realistic approximation of human or platform mobility in indoor environments. These values generate sufficiently dynamic scenarios to challenge the RL agent, while remaining within the operational bounds of typical robotic systems.

The training consists of **three episodes**, each limited to **3000 steps**. As mentioned before, within each episode, the environment transitions through four predefined mobility scenarios to diversify the agent's experience. The durations of these scenarios were empirically chosen to balance exposure to each situation while emphasizing complex interactions in Scenario D, which occupies the majority of the episode. This structure is intended to improve generalization and robustness in the learned policy.

Table 3.10: Training parameters and configuration of episodes.

Parameter	**Value**
Time step (Δt)	0.2
Velocity step (δ)	0.35
Max gNB velocity (v_{gNB}^{max})	1
UE and obstacle velocity (v_{obj})	0.6
Reward scale factor (k)	2
Number of Training Episodes (N)	3
Episode Step Limit ($S_{\max}$)	3000
Scenario A Duration	100
Scenario B Duration	450
Scenario C Duration	450
Scenario D Duration	2000

Table 3.11: DQN Training Hyperparameters

Parameter	**Value**
Batch Size	64
Learning Rate (α)	0.001
Exploration Epsilon (initial) (ε)	0.9
Reward Discount Factor (γ)	0.9
Target Network Update Frequency	100 steps
Replay Buffer Capacity	1000 transitions

Action selection followed an ε-greedy policy, where the exploration rate ε was initially set to 0.9 to encourage broad exploration of the environment. It then decayed linearly at each time step to balance exploration and exploitation, following the update rule:

$$\varepsilon_{t+1} = \max\left(0.1,\ \varepsilon_t - \frac{0.9}{S_{\max} \cdot N}\right)$$

where $S_{\max}$ is the episode step limit and N is the number of training episodes. This schedule ensured that exploration remained significant in the early stages, gradually giving way to more refined, exploitation-driven behavior as training progressed. The remaining training hyperparameters, followed commonly adopted values in typical RL applications.

While other network configurations, training methodologies, and hyperparameter values were explored during development, the configuration described here was adopted for the final agent due to its stable training behavior and satisfactory empirical performance. It is important to note that this dissertation did not focus on exhaustive hyperparameter tuning or model optimization, as the primary goal was to validate the feasibility of perception-aided mobility control using reinforcement learning within the CONVERGE system framework. Additional training results–obtained after the main implementation phase–are presented in Appendix B, including an analysis of agent convergence time under different training episode counts.

3.5 xApp for Real-Time gNB Repositioning

The developed xApp is responsible for orchestrating the real-time repositioning of the gNB by leveraging spatial and vision-based data collected from the network. It is deployed within the nearRT-RIC and communicates with connected E2 nodes through the E2 interface, subscribing to relevant SMs and issuing control commands based on learned policies.

At runtime, the xApp establishes a connection with the RIC API, discovers connected E2 nodes, and subscribes to the POS (ID 101), VIS (ID 102), and MAC (ID 142) SMs. The subscription period is configured to 10 ms. Each SM has a dedicated callback function:

- `sm_cb_mac()`: collects SNR statistics for performance evaluation, logging the average Physical Uplink Shared Channel (PUSCH) SNR every 200 ms.

- `sm_cb_pos()`: stores incoming POS indication messages into rolling buffers (`gnb_pos` and `lis_pos`), maintaining the last N_{buffer} positions, velocities, and orientations of both the gNB and LIS. This function implements the logic defined in Algorithm 3.
- `sm_cb_vis()`: uses the previously stored positional data along with incoming VIS indication messages to compute the absolute positions of detected objects from both the gNB and LIS perspectives, using the transformation equations defined in (3.1). The resulting object positions are stored in the `gnb_objects` and `lis_objects` buffers, which maintain the last N_{buffer} detections to enable velocity estimation through temporal differencing. The overall processing logic for this function is defined in Algorithm 4.

Algorithm 3 POS SM callback function.

Require: POS indication message `pos_ind`
1: Lock mutex for thread-safe access
2: **if** `pos_ind` received from gNB **then**
3: Update `gnb_pos` buffer with `pos_ind`
4: **else if** `pos_ind` received from LIS **then**
5: Update `lis_pos` buffer with `pos_ind`
6: **end if**
7: Unlock mutex

Algorithm 4 VIS SM callback function.

Require: VIS indication message `vis_ind`
1: Lock mutex for thread-safe access
2: **if** `vis_ind` received from gNB **then**
3: Update `gnb_objects` buffer using current `gnb_pos` and `vis_ind`
4: Compute `gnb_objects` velocities based on previous `gnb_objects` positions
5: **else if** `vis_ind` received from LIS **then**
6: Update `lis_objects` buffer using current `lis_pos` and `vis_ind`
7: Compute `lis_objects` velocities based on previous `lis_objects` positions
8: **end if**
9: Unlock mutex

All data processing and decision-making logic is executed within a dedicated control loop thread `ctrl_thread()` that runs every 200 ms. During each iteration, the xApp invokes the `pos_fusion()` function, which fuses UE and obstacle detections from gNB and LIS perspectives into a unified positional estimate. If the UE is not visible to the gNB, LIS observations are treated as ground truth and `los_status` is set to 1 (LoS obstructed). This fusion process is detailed in Algorithm 5.

The estimated positions and LoS status are then used to construct a feature vector, which is passed to the trained reinforcement learning model. The model outputs one of the three discrete actions described in Table 3.9. Based on the current gNB position and the chosen action, a control message is generated, containing the target coordinates of the gNB, and is transmitted to the gNB

via the E2 interface using the `control_sm_xapp_api()` API. The control logic for this process is presented in Algorithm 6, and for a better visual reference Figure 3.9 provides a visual representation of the data dependencies, illustrating the flow from the reception of indication messages to the transmission of a control command; data types are highlighted in green for clarity.

Algorithm 5 Multi-perspective position fusion.

Require: Object positions estimated from gNB and LIS: `gnb_objects` and `lis_objects`

`los_status` $\leftarrow 0$
if UE not detected by gNB **then**
    `los_status` $\leftarrow 1$
    Set `ue` position to the most recent LIS estimate
    Set `ue` velocity to the trimmed mean of the last N_{buffer} velocities estimated by LIS
else
    Set `ue` position as the average of the latest gNB and LIS estimates
    Set `ue` velocity as the average of the trimmed means of the last N_{buffer} velocities estimated by gNB and LIS
end if
Set `obs` position as the average of the latest gNB and LIS estimates
Set `obs` velocity as the average of the trimmed means of the last N_{buffer} velocities estimated by gNB and LIS
return `ue`, `obs` and `los_status`

Algorithm 6 xApp mobility control loop.

while `ctrl_thread_running` **do**
    Lock mutex for thread-safe access
    Estimate `ue` and `obs` positions and get `los_status` using `pos_fusion()`
    Update `ue_pos` and `obs_pos` buffers with `ue` and `obs`
    Compute input `state_vector` from current `gnb_pos`, `ue_pos`, `obs_pos`, and `los_status`
    Run inference RL model with `state_vector` to get `action`
    Generate control message from current `gnb_pos` and inferred `action`
    Send control command to gNB via `control_sm_xapp_api()`
    Unlock mutex
    Compute next wake-up time and sleep until then
end while

This loop allows the xApp to continuously react to dynamic conditions, making real-time adjustments to the gNB's position and logging relevant performance indicators for offline evaluation.

This xApp demonstrates a practical and modular approach to AI-driven mobility control, highlighting how vision-based perception and control policies can be integrated within the O-RAN framework using FlexRIC. Its design promotes future extensibility with minimal changes to the interface logic, supporting further research in sensing and perception-aided RAN optimization.

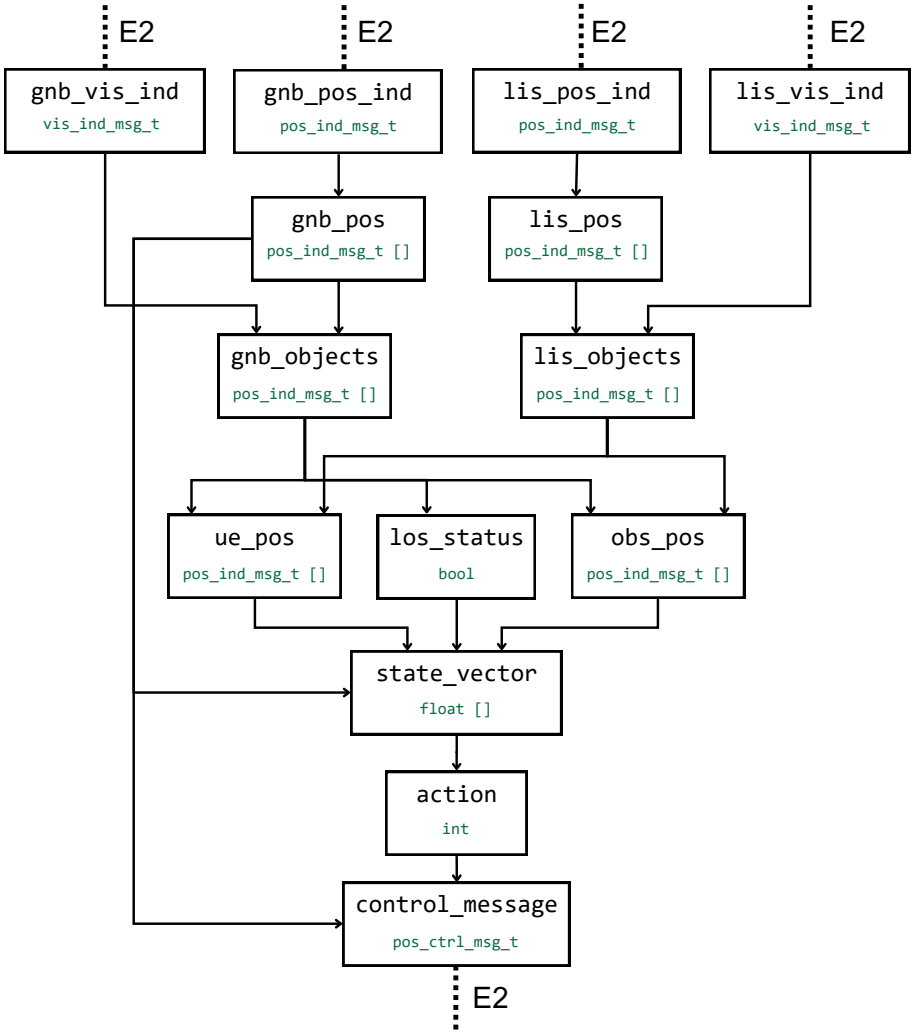


Figure 3.9: xApp Data Dependency Diagram. "$[d_1] \rightarrow [d_2]$" means "d_2 depends of d_1".

3.6 Summary

This chapter presented the complete system architecture, design, and implementation of a novel multimodal O-RAN framework for intelligent gNB mobility control. The proposed system is composed of several key components, each fulfilling a specific role in enabling real-time, perception-aided RAN configuration:

- **Multimodal O-RAN Framework**: Built upon the FlexRIC platform, this framework introduces two custom Service Models—**POS** (position) and **VIS** (vision)—to extend traditional RAN functionalities with spatial and vision-semantic perception capabilities. These SMs are handled by customized E2 agents, referred to as **E2′ agents**, which act as communication bridges between the nearRT-RIC and external modules such as the gNB and the LIS. Each E2′ agent collects data from local control and vision functions, formats it according to the SM structures, and reports it to the RIC. In the case of the gNB, the agent also processes control commands received from the xApp, forwarding them to the local controller.

- **CONVERGE Chamber Simulator (CC-SIM)**: A Python-based 3D simulation environment that emulates the physical and visual behavior of the gNB, UE, and obstacles. It generates synthetic position and vision-semantic data in JSON format to feed the E2′ agents, while being able to receive and process control commands. CC-SIM also interfaces with the OAI RF simulator to simulate RF path loss dynamics based on LoS conditions of the simulated environment. CC-SIM supports three operational modes: 1) *training*, for reinforcement learning agents development; 2) *simulation*, for offline inference, validation, and visualization; and 3) *live testing*, for real-time integration with the multimodal O-RAN framework.

- **Reinforcement Learning Agent**: A Deep Q-Network trained to learn gNB mobility strategies that preserve LoS with the UE. The agent operates over a compact geometric state space, produces velocity-based actions, and is trained using a structured episodic formulation within CC-SIM.

- **xApp for Mobility Control**: Deployed within the nearRT-RIC, this xApp collects POS and VIS messages from both the gNB and the LIS E2′ agents, estimates object positions and motion, and runs inference using the trained RL model. It then issues mobility commands to the gNB via the E2 interface, closing the perception-to-action control loop.

Together, these components form a modular and interoperable system that combines computer vision, reinforcement learning, and O-RAN control. The integration of standardized interfaces (E2, NG, NR-Uu) with custom extensions (POS, VIS) and simulated hardware (via CC-SIM and RFsim) enables realistic testing of advanced control strategies. This design lays a strong foundation for future extensions involving multi-agent control, occlusion-aware sensing, or real-world experimental deployments.

Chapter 4

System Deployment and Validation

This chapter presents the deployment and validation of the proposed perception-aided O-RAN framework. Section 4.1 outlines the overall testing methodology adopted for modular and integrated evaluation. Section 4.2 validates the deployment of the standalone 5G network using OpenAirInterface components. Section 4.3 focuses on the implementation and integration of the custom Service Models and E2′ agents. Section 4.4 demonstrates the interaction with the CC-SIM environment and validates the real-time update of radio conditions via the channel model interface. Section 4.5 presents the execution of dynamic use case scenarios using the trained RL model, comparing the behavior of controlled and static gNB configurations. Finally, Section 4.6 evaluates the alignment between simulated and live control behaviors, validating the reliability of the simulator for policy assessment prior to deployment.

4.1 Methodology

The system validation strategy was structured to incrementally test each subsystem before evaluating the integrated framework. The methodology involved sequential deployment, functional validation, and performance assessment of all major components, namely the OAI-based 5G network, FlexRIC with E2′ agents, the CC-SIM environment, and the intelligent xApp.

First, the standalone 5G network was deployed and validated through basic connectivity and performance tests using standard tools such as *ping* and *iPerf*. This ensured a working baseline communication between the Core Network and UE via the RFsimulator.

Second, custom Service Models and E2′ agents were validated by verifying successful message exchanges over the E2 interface. FlexRIC was configured to register the SMs and interface with the agents, while POS and VIS indications were logged to confirm end-to-end integration between the E2′ agents and testing xApps.

Third, the CC-SIM environment was connected to the system via both the E2' and CM interfaces. The E2' connection enabled the periodic export of spatial (POS) and vision (VIS) context, while the CM interface allowed CC-SIM to dynamically modify the emulated RF environment by updating path loss values in real-time.

Fourth, control logic was introduced via the xApp running a trained DNN model. The xApp subscribed to RAN state via FlexRIC and issued gNB control commands based on inference results from the learned policy.

Finally, performance was evaluated under defined use case scenarios using two mobility patterns each. Metrics such as SNR, path loss, throughput, and LoS obstruction time were monitored to quantify system responsiveness. A baseline comparison with static gNB configurations was also conducted to highlight the advantages of the proposed control approach.

After validating the proposed use cases, an evaluation was conducted in both simulation and live modes. In live mode, all modules–including FlexRIC, xApp, CC-SIM, and the RFsim-enabled gNB–interacted in real time. In simulation mode, the same inference logic was evaluated without executing the physical channel or RAN stack, enabling pre-deployment testing.

4.2 Standalone 5G Network

The initial validation focused on verifying the correct operation of the OpenAirInterface modules responsible for deploying a standalone 5G network. The overall deployment procedure followed the guidelines provided in the tutorial [40]. The 5G Core Network (5GC) was launched using Docker containers, as shown in Figure 4.1 (a). The `-d` flag ensures the containers run in detached mode, suppressing terminal output while keeping the services running in the background.

Figure 4.1 (b) shows the command used to launch the OAI-gNB. The `-O` flag specifies the configuration file to be used; `-gNBs.[0].min_rxtxtime` sets the minimum reception-transmission time offset to ensure stable signal handling; and `-rfsim` enables the use of the OAI RFsimulator, which emulates the physical radio link between the gNB and the UE in software. Upon initialization, the gNB outputs a recommended configuration set, including subcarrier spacing, frame parameters, and numerology to ensure UE synchronization, as shown in Figure 4.1 (b).

The UE is then deployed using the command shown in Figure 4.1 (c). The `-C`, `-r`, and `-numerology` flags are configured using the values provided by the gNB, ensuring compatibility in subcarrier spacing and bandwidth. The `-band` flag sets the frequency band used in simulation, while `-uicc0.imsi` specifies the International Mobile Subscriber Identity (IMSI) of the simulated UE. The IMSI value `001010000000001` is reserved for test networks. The `-rfsim` flag once again enables simulated RF communication, and `-rfsimulator.serveraddr` defines the IP address of the server hosting the RFsimulator. After initialization, the UE terminal confirms the successful configuration of its network interface with the IP address `10.0.0.2`, as shown in Figure 4.1 (c).

To verify the correct operation of the deployed standalone 5G network, the *ping* utility was used to assess end-to-end connectivity between the 5GC and the UE. As illustrated in Figure 4.2 (a), the UE interface `oaitun_ue1` successfully pinged the external data network interface of the 5GC (`oai-ext-dn`) at IP address `192.168.70.135`. The successful exchange of ICMP packets, as confirmed by the Wireshark capture in Figure 4.2 (b), demonstrates that the UE is able to successfully reach and communicate with the Core Network.

```
ubuntu@ubuntu:~/oai-cn5g$ sudo docker compose up -d
```

(a) OAI-5GC.

```
ubuntu@ubuntu:~/oai/path_to_ran_build/build$ sudo ./nr-softmodem -O ../../../targets/PROJECTS/
GENERIC-NR-5GC/CONF/gnb.sa.band78.fr1.106PRB.usrpb210.conf --gNBs.[0].min_rxtxtime 6 --rfsim
(...)
[PHY]   Command line parameters for OAI UE: -C 3619200000 -r 106 --numerology 1
```

(b) OAI-gNB.

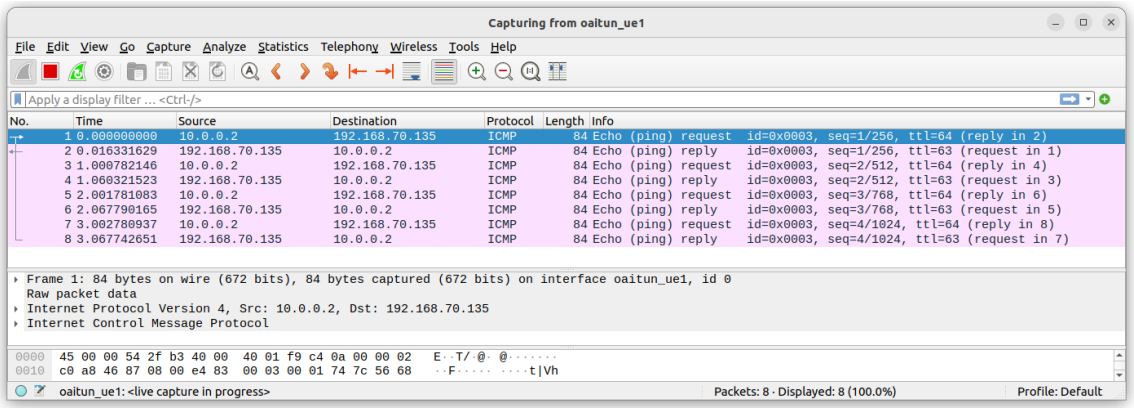
```
ubuntu@ubuntu:~/oai/path_to_ran_build/build$ ./nr-uesoftmodem -C 3619200000 -r 106 --numerology 1
--band 78 --uicc0.imsi 001010000000001 --rfsim --rfsimulator.serveraddr 127.0.0.1
(...)
[OIP]   Interface oaitun_ue1 successfully configured, IPv4 10.0.0.2, IPv6 (null)
```

(c) OAI-UE.

Figure 4.1: OAI standalone 5G network deployment.

```
ubuntu@ubuntu:~$ ping 192.168.70.135 -I oaitun_ue1
PING 192.168.70.135 (192.168.70.135) from 10.0.0.2 oaitun_ue1: 56(84) bytes of data.
64 bytes from 192.168.70.135: icmp_seq=1 ttl=63 time=64.4 ms
(...)
```

(a) Connectivity validation test.



(b) Wireshark capture.

Figure 4.2: Pinging OAI-5GC (oai-ext-dn) from OAI-UE.

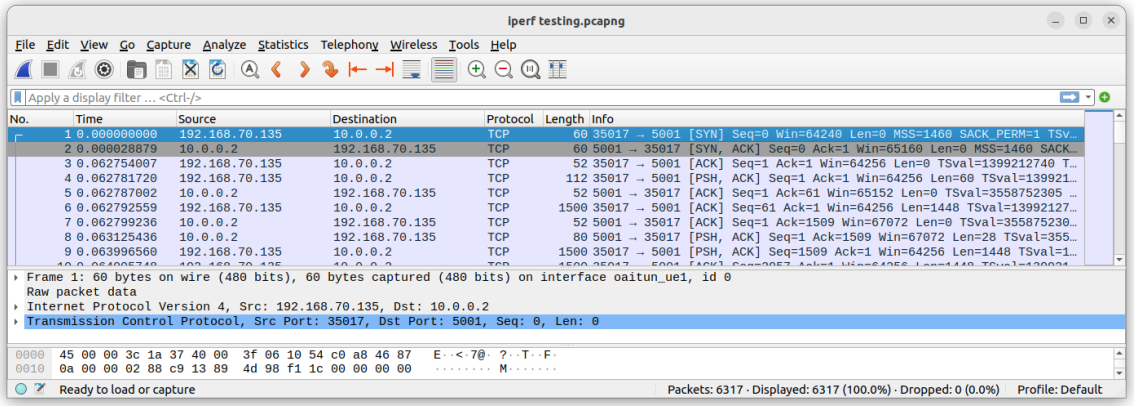
Following the successful connectivity test, the transmission of TCP packets was evaluated using the *iPerf* tool. An *iPerf* server was first launched on the UE side, bound to the corresponding interface, as shown in Figure 4.3 (a). Subsequently, an *iPerf* client was executed within the `oai-ext-dn` container to initiate TCP traffic toward the UE, as illustrated in Figure 4.3 (b). The packet exchange captured in Wireshark, presented in Figure 4.3 (c), confirms that the TCP traffic flow was successfully established. This validation is particularly important, as *iPerf* will serve as the primary tool for evaluating the network performance in subsequent stages of the system.

```
ubuntu@ubuntu:~$ iperf -s -i 1 -B 10.0.0.2
```

(a) *iPerf* server binded to OAI-UE.

```
ubuntu@ubuntu:~$ sudo docker exec -it oai-ext-dn iperf -i 1 -B 192.168.70.135 -c 10.0.0.2
```

(b) *iPerf* client inside the oai-ext-dn container.



(c) Wireshark capture.

Figure 4.3: *iPerf* testing and validation.

4.3 Service Models and E2′ Agents

After verifying the correct operation of the standalone 5G network, the next step involved validating the integration of the developed custom Service Models. When implementing new SMs within the FlexRIC framework, a set of predefined testing files is automatically generated to verify the basic functionality of the SM interface. Following the development of the custom POS SM, its corresponding test was executed using the command shown in Figure 4.4, resulting in a successful output. A similar procedure was applied to the VIS SM, confirming its correct implementation and compatibility with the FlexRIC infrastructure.

```
ubuntu@ubuntu:~/path_to_flexric/build$ ./test/sm/pos_sm/test_pos_sm
Success
```

Figure 4.4: POS SM testing.

We then proceeded to test the correct operation of the developed E2′ agents. The first step involved verifying their successful connection to the near-RT RIC via the E2 interface. FlexRIC was initially launched, as shown in Figure 4.5 (a), where we confirmed the proper registration of the custom Service Models (SMs) with IDs 101 and 102.

Subsequently, the E2′ agents for both the gNB and LIS were launched using the commands depicted in Figures 4.5 (b) and 4.5 (c), respectively. Upon initialization, the terminal outputs of

both agents indicated the successful loading of the corresponding SM plugins, followed by the execution of the E2 Setup procedure, as presented in Figure 2.7.

The FlexRIC terminal also confirmed the establishment of the E2 connection, displaying the acceptance of RAN function capabilities advertised by the E2′ agents, thereby validating the end-to-end integration of the agents with the near-RT RIC.

```
ubuntu@ubuntu:~/path_to_flexric/build$ ./examples/ric/nearRT-RIC
(...)
[NEAR-RIC]: Loading SM ID = 101 with def = POS_STATS_V0  // LOADING CUSTOM SM
[NEAR-RIC]: Loading SM ID = 102 with def = VIS_STATS_V0
(...)
[E2AP]: E2 SETUP-REQUEST rx from PLMN 1. 1 Node ID 1 RAN type ngran_gNB  //SETUP PROCEDURE WITH E2' AGENTS
[NEAR-RIC]: Accepting RAN function ID 101 with def = POS_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 102 with def = VIS_STATS_V0
[E2AP]: E2 SETUP-REQUEST rx from PLMN 1. 1 Node ID 3 RAN type ngran_LIS
[NEAR-RIC]: Accepting RAN function ID 101 with def = POS_STATS_V0
[NEAR-RIC]: Accepting RAN function ID 102 with def = VIS_STATS_V0
```

(a) FlexRIC.

```
ubuntu@ubuntu:~/path_to_flexric/build$ ./examples/agents/gnb_custom_agent
(...)
[E2 AGENT]: nb_id 1, mcc 1, mnc 1, mnc_digit_len 2, ran_type ngran_gNB
[E2 AGENT]: Initializing ...
[E2 AGENT]: Opening plugin from path = /usr/local/lib/flexric/libpos_sm.so  // OPENING CUSTOM SM PLUGINS
[E2 AGENT]: Opening plugin from path = /usr/local/lib/flexric/libvis_sm.so
[E2 AGENT]: E2 SETUP-REQUEST tx  // E2 SETUP PROCEDURE
[E2 AGENT]: E2 SETUP-RESPONSE rx
[E2 AGENT]: Transaction ID E2 SETUP-REQUEST 0 E2 SETUP-RESPONSE 0
```

(b) gNB E2′ agent.

```
ubuntu@ubuntu:~/path_to_flexric/build$ ./examples/agents/lis_custom_agent
(...)
[E2 AGENT]: nb_id 3, mcc 1, mnc 1, mnc_digit_len 2, ran_type ngran_LIS
[E2 AGENT]: Initializing ...
[E2 AGENT]: Opening plugin from path = /usr/local/lib/flexric/libpos_sm.so  // OPENING CUSTOM SM PLUGINS
[E2 AGENT]: Opening plugin from path = /usr/local/lib/flexric/libvis_sm.so
[E2 AGENT]: E2 SETUP-REQUEST tx  // E2 SETUP PROCEDURE
[E2 AGENT]: E2 SETUP-RESPONSE rx
[E2 AGENT]: Transaction ID E2 SETUP-REQUEST 0 E2 SETUP-RESPONSE 0
```

(c) LIS E2′ agent.

Figure 4.5: E2′ agents deployment and validation.

Finally, to validate the end-to-end transmission of information from the E2′ agents to the xApp, a monitoring xApp was developed. This xApp was launched using the command shown in Figure 4.6 (a), and its correct operation was confirmed through the successful execution of the RIC Subscription Request and Subscription Delete procedures–depicted in Figure 2.8–as well as successful printing of the indication messages.

Figures 4.6 (b) and 4.6 (c) display the terminal outputs of FlexRIC and the gNB E2′ agent, respectively, upon receiving the aforementioned subscription messages. These results confirm that the xApp was able to interface with FlexRIC, subscribe to POS indication messages, and initiate

communication through the E2 interface, completing the functional validation of the monitoring path. The transmission of control messages from the xApp to the E2′ agent was similarly tested.

```
ubuntu@ubuntu:~/path_to_flexric/build/$ ./examples/xApp/c/custom/xapp_monitor
(...)
// RIC SUBSCRIPTION REQUEST
[xApp]: E42 RIC SUBSCRIPTION REQUEST tx RAN_FUNC_ID 101 RIC_REQ_ID 1  // POS
[xApp]: SUBSCRIPTION RESPONSE rx
[xApp]: Successfully subscribed to RAN_FUNC_ID 101
[xApp]: E42 RIC SUBSCRIPTION REQUEST tx RAN_FUNC_ID 102 RIC_REQ_ID 2  // VIS
[xApp]: SUBSCRIPTION RESPONSE rx
[xApp]: Successfully subscribed to RAN_FUNC_ID 102
(...)
// LAST INDICATION MESSAGES
    100 POS ind_msg tstamp = 7065883090 [µs]
Node Type: 0
Position: x = 300, y = 125, z = 150
Velocity: x = 0, y = 0, z = 0
Angles: (theta = 0, phi = 0)

    100 VIS ind_msg tstamp = 7065883090 [µs]
Node Type: 0
ID: 1, Class: 1, Bounding Box: (x = 300, y = 500, w = 15, h = 30)
Angles: (theta = 0, phi = 0), Distance: 375
ID: 3, Class: 3, Bounding Box: (x = 400, y = 375, w = 100, h = 180)
Angles: (theta = -21, phi = 38), Distance: 275
// RIC SUBSCRIPTION DELETE
[xApp]: E42 RIC_SUBSCRIPTION_DELETE_REQUEST tx RAN_FUNC_ID 101 RIC_REQ_ID 1
[xApp]: E42 SUBSCRIPTION DELETE RESPONSE rx
[xApp]: E42 RIC_SUBSCRIPTION_DELETE_REQUEST tx RAN_FUNC_ID 102 RIC_REQ_ID 2
[xApp]: E42 SUBSCRIPTION DELETE RESPONSE rx
[xApp]: Sucessfully stopped
```

(a) Monitoring xApp.

```
(...)
// SUBSCRIPTION REQUEST
[iApp]: SUBSCRIPTION-REQUEST RAN_FUNC_ID 101 RIC_REQ_ID 1 tx  // POS
[iApp]: SUBSCRIPTION-REQUEST RAN_FUNC_ID 102 RIC_REQ_ID 2 tx  // VIS
(...)
// SUBSCRIPTION DELETE
[NEAR-RIC]: SUBSCRIPTION_DELETE_REQUEST tx RAN_FUNC_ID 101 RIC_REQ_ID 1021
[iApp]: RIC_SUBSCRIPTION_DELETE_REQUEST tx RIC_REQ_ID 1021
[iApp]: RIC_SUBSCRIPTION_DELETE_REQUEST tx RAN_FUNC_ID 101 RIC_REQ_ID 1

[NEAR-RIC]: SUBSCRIPTION_DELETE_REQUEST tx RAN_FUNC_ID 102 RIC_REQ_ID 1022
[iApp]: RIC_SUBSCRIPTION_DELETE_REQUEST tx RIC_REQ_ID 1022
[iApp]: RIC_SUBSCRIPTION_DELETE_REQUEST tx RAN_FUNC_ID 102 RIC_REQ_ID 2
```

(b) FlexRIC.

```
(...)
// SUBSCRIPTION REQUEST
[E2 AGENT]: RIC_SUBSCRIPTION_REQUEST rx RAN_FUNC_ID 101 RIC_REQ_ID 1021  // POS
[E2-AGENT]: RIC_SUBSCRIPTION_REQUEST rx
[E2 AGENT]: RIC_SUBSCRIPTION_REQUEST rx RAN_FUNC_ID 102 RIC_REQ_ID 1022  // VIS
[E2-AGENT]: RIC_SUBSCRIPTION_REQUEST rx
// SUBSCRIPTION DELETE
[E2-AGENT]: RIC_SUBSCRIPTION_DELETE_REQUEST rx RAN_FUNC_ID 101  RIC_REQ_ID 1021
[E2-AGENT]: RIC_SUBSCRIPTION_DELETE_REQUEST rx RAN_FUNC_ID 102  RIC_REQ_ID 1022
```

(c) gNB E2′ agent.

Figure 4.6: Monitoring xApp deployment for end-to-end communication validation.

4.4 CC-SIM

After confirming the correct operation of the E2′ agents, the communication with the CC-SIM environment was also validated, focusing on testing the CM interface. Throughout all experimental validations, the entire system was deployed on a single machine. To streamline development and avoid unnecessary network overhead, the E2′ interface was emulated locally by writing and reading JSON files in shared directories accessed by the corresponding E2′ agents.

To enable the CM interface, additional arguments had to be specified when launching the OAI-gNB, as outlined in [35]. In particular, the flag `-rfsimulator.options chanmod` was used to activate the RFsimulator's channel model emulation capabilities, while the `-telnetsrv` flag enabled the Telnet-based interface used to receive real-time channel parameter updates from external controllers such as CC-SIM.

To verify that CC-SIM was correctly transmitting dynamic path loss values to the gNB, a LoS obstruction test was conducted. In this test, the gNB and UE were kept static, while the obstacle was moved perpendicularly across their direct line of sight, stopping at the midpoint between them. This scenario is visually depicted in Figure 4.7, and the initial spatial and physical parameters of the involved objects are listed in Table 4.1.

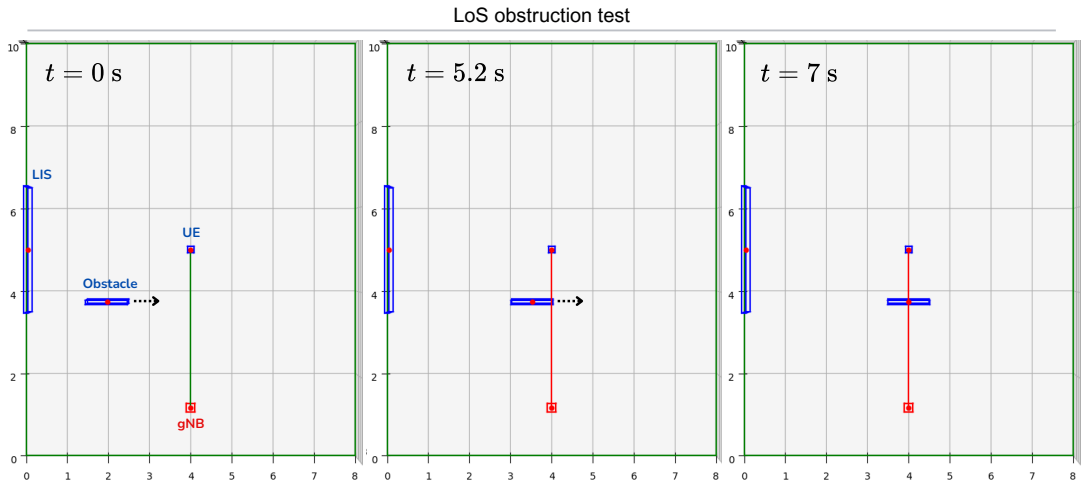


Figure 4.7: LoS obstruction test. Object positions on relevant timestamps.

Table 4.1: Initialized Object Attributes in CC-SIM.

Object	x	y	z	(W × L × H)	Attenuation [dB]
gNB	4	1.25	1.5	0.2 × 0.2 × 0.1	—
UE	4	5.00	1.5	0.15 × 0.15 × 0.3	—
Obstacle	2	3.75	0.9	1.0 × 0.1 × 1.8	25

Figure 4.8 illustrates both the updated gNB launch command and indicative SNR values observed before and after LoS obstruction. A notable drop in the reported SNR occurs when the

obstacle enters the LoS path, validating the expected interaction between CC-SIM and the RF simulator component.

```
ubuntu@ubuntu:~/path_to_flexric/build$ sudo ./nr-softmodem -O ../../../targets/PROJECTS/GENERIC-NR-5GC/CONF/
gnb.sa.band78.fr1.106PRB.usrpb210.conf --gNBs.[0].min_rxtxtime 6 --rfsim --rfsimulator.options chanmod --telnetsrv

UE 1fa7: ulsch_rounds 25/0/0/0, ulsch_errors 0, ulsch_DTX 0, BLER 0.04305 MCS (0) 9 (Qm 2 deltaMCS 0 dB) NPRB 5
SNR 50.5 dB
(...)
[TELNETSRV] Telnet client connected...
// RECEIVING PATH LOSS FROM CC-SIM
UE 1fa7: ulsch_rounds 40/0/0/0, ulsch_errors 0, ulsch_DTX 0, BLER 0.01094 MCS (0) 9 (Qm 2 deltaMCS 0 dB) NPRB 5
SNR 39.0 dB  // LoS CLEAR
(...)
UE 1fa7: ulsch_rounds 81/2/2/2, ulsch_errors 2, ulsch_DTX 0, BLER 0.04305 MCS (0) 9 (Qm 2 deltaMCS 0 dB) NPRB 23
SNR 13.5 dB  // LoS OBSTRUCTION
```

Figure 4.8: OAI-gNB SNR values before and after CC-SIM connection, without and with LoS obstruction.

At this stage, we validated the system's responsiveness to real-time updates in the channel model. To assess this, continuous radio traffic was generated between the 5GC and UE using the *iPerf* tool to monitor throughput, while a dedicated monitoring xApp recorded SNR values every 200 ms. The results, depicted in Figure 4.9, show a clear correlation between increased path loss and degraded radio performance. A sharp drop in average SNR is observed immediately after the path loss rises — reflecting a LoS obstruction event — which ultimately leads to a complete loss of throughput, reaching 0 Mbit/s under the given attenuation conditions.

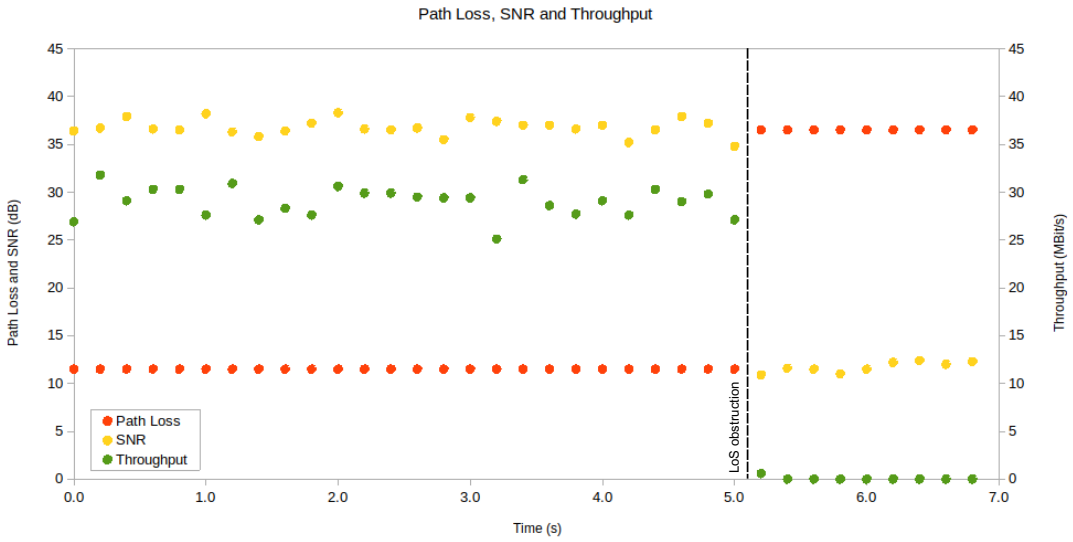


Figure 4.9: LoS obstruction test. Path loss, SNR, and throughput variation over time. The results demonstrate the system's sensitivity to dynamic RF conditions: a rise in path loss due to LoS obstruction leads to a proportional drop in SNR, which directly causes throughput degradation.

4.5 Use Case Validation

To validate the complete system framework and the trained DNN model, we carried out two distinct use cases with two different movement patterns each. Each test is labeled using a letter to denote the use case and a number to indicate the corresponding movement pattern. The use cases and movement scenarios are defined as follows:

- **Use Case O:** Mobile **O**bstacle and static UE.
- **Use Case U:** Mobile **U**E and static obstacle.
- **Movement 1:** Mobile node starts from $x = 2$ m and moves constantly with $v = 0.6$ m/s until it reaches $x = 6$ m.
- **Movement 2:** Mobile node starts from $x = 6$ m and moves constantly with $v = -0.6$ m/s until it reaches $x = 2$ m.

To execute the validation tests, each module of the system was deployed sequentially using the previously described commands, following the order below:

1. **Deploy the standalone 5G network**, initializing the OAI-5GC, OAI-gNB, and OAI-UE components to establish end-to-end connectivity.
2. **Start the FlexRIC controller**, enabling the reception of E2 connections.
3. **Launch the custom E2′ agents** for the gNB and LIS, which establish E2 connections with FlexRIC.
4. **Run CC-SIM in live mode**, configured to periodically export POS and VIS messages to the E2′ agents.
5. **Establish the CM interface connection** between CC-SIM and the gNB, enabling real-time updates to the RF channel model and emulating physical conditions.
6. **Initiate traffic generation using *iPerf***, transmitting data between the OAI-5GC and OAI-UE to evaluate radio performance under dynamic conditions.
7. **Launch the xApp**, which subscribes to SM indications and starts issuing gNB movement commands based on the trained DNN model inference.
8. **Execute the test in CC-SIM**, moving the UE or obstacle accordingly to the selected use case, while allowing the gNB to be dynamically controlled via near-RT RIC decisions.

During the execution of the tests, **the gNB exhibited adaptive behaviors aligned with the intended control strategy**. Table 4.2 summarizes the observed actions taken by the gNB in response to each use case scenario, demonstrating its ability to proactively adjust its position to maintain LoS connectivity. Visual snapshots corresponding to each scenario are provided in Appendix A.1.

Table 4.2: Description of gNB control behavior for the evaluated Use Cases.

Use Case O	The gNB initially moves in the same direction as the mobile obstacle to maintain LoS with the static UE. Upon detecting an imminent LoS obstruction, it reverses direction to counteract the obstacle's movement and re-establish unobstructed connectivity.
Use Case U	The gNB starts by moving opposite to the direction of the mobile UE. As the LoS becomes threatened, the gNB dynamically adjusts by following the UE's trajectory to preserve connectivity and minimize blockage.

To assess the advantages introduced by the RL-based mobility controller, each use case was also executed with a static gNB, serving as a baseline for comparison[1]. Figures 4.10 and 4.11 present the radio performance metrics for Use Case O and Use Case U, respectively, where both static (top figure) and controlled (figures from the middle and the bottom) configurations are shown. Each subplot illustrates the temporal evolution of path loss, SNR, and throughput. **Two key benefits** of the proposed control strategy are clearly observable: first, the controller is able to **delay the LoS obstruction** by dynamically adjusting the gNB position; second, it significantly **reduces the total duration of LoS obstruction**, allowing faster recovery of link quality and throughput. Quantitatively, in Use Case O, the average LoS obstruction time was reduced by approximately **12.5% to 25%** compared to the static baseline. In Use Case U, this reduction ranged between **25% and 41.6%**, demonstrating the controller's responsiveness and effectiveness in adapting to different movement patterns and preserving radio link performance.

[1] Only one movement pattern was tested for the static gNB baseline, as the system exhibits spatial symmetry, leading to in equivalent outcomes under mirrored conditions.

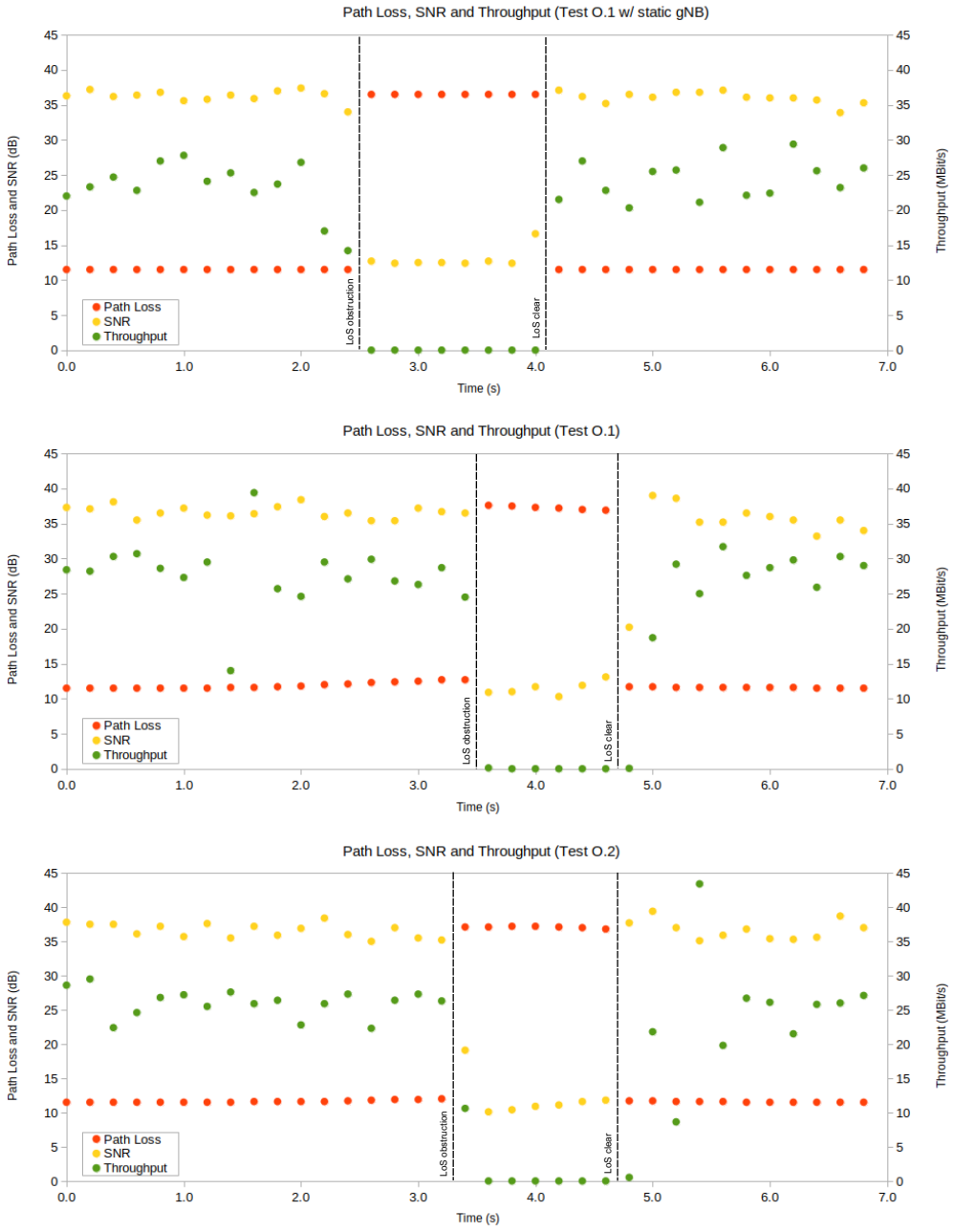


Figure 4.10: Use case O. Path loss, SNR, and throughput variation over time. The results show that the RL-based mobility controller is able to dynamically reposition the gNB to delay LoS obstruction and reduce the overall duration of degraded link quality. Compared to the static baseline (top figure), LoS obstruction time was reduced by up to 25%.

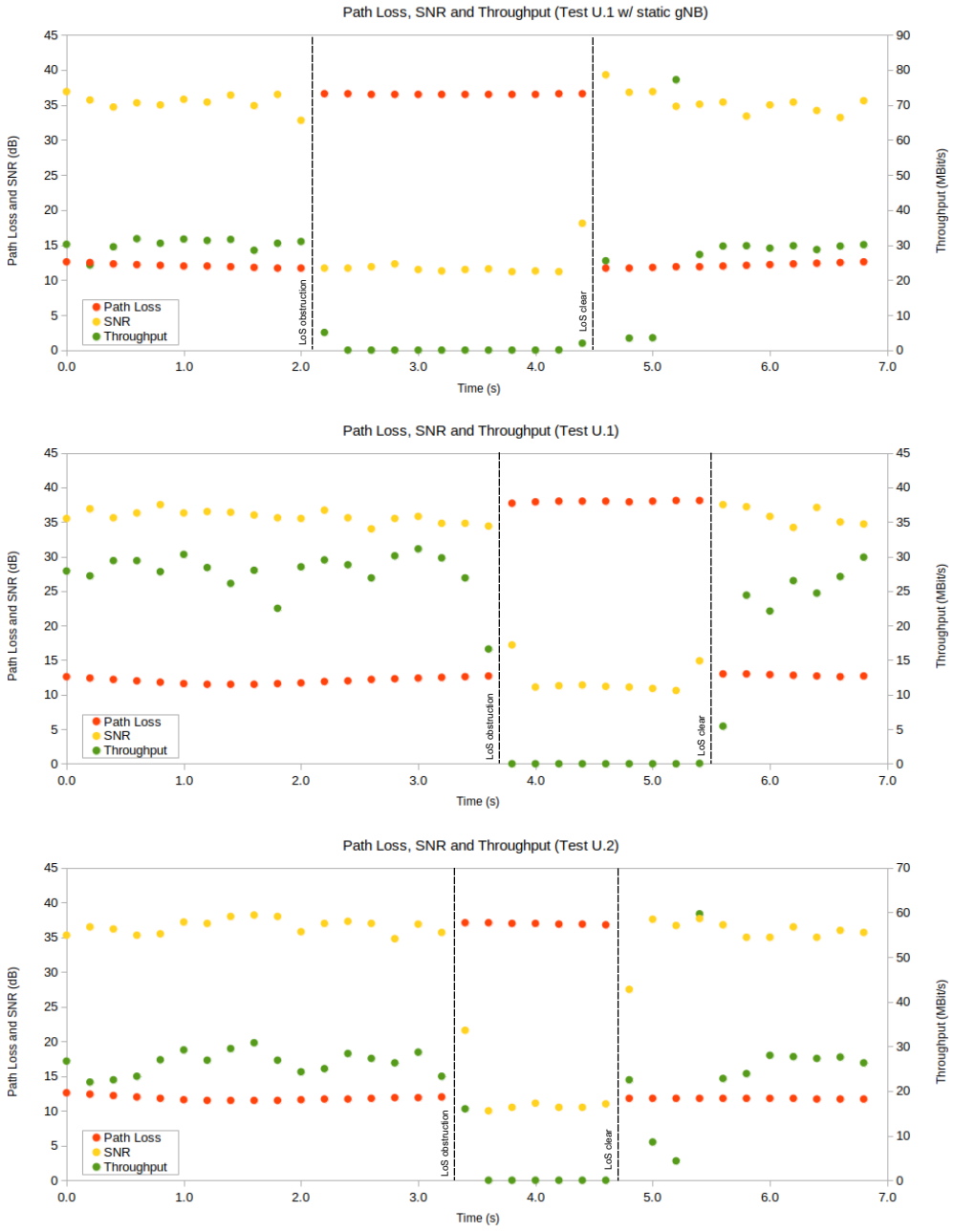


Figure 4.11: Use case U. Path loss, SNR, and throughput variation over time. The results show that the RL-based mobility controller is able to dynamically reposition the gNB to delay LoS obstruction and reduce the overall duration of degraded link quality. Compared to the static baseline (top figure), LoS obstruction time was reduced by up to 41.6%.

4.6 Simulation vs Live Control

Finally, we tested and validated the precision of the control loop execution when transitioning from simulation-based to live deployment. In simulation mode, the RL agent is integrated directly into the simulator, and control decisions are applied internally with immediate effect; in live mode, the RL model is embedded in the xApp, which receives periodic E2 indication messages, performs inference, and sends control commands to the gNB through the O-RAN control loop. To do this, we compared the gNB's x-axis position over time during simulation-controlled and live-controlled executions across the same scenarios described in the previous section. Figures 4.12 and 4.13 presents this comparison, illustrating the gNB's movement as determined by the RL agent in both modes.

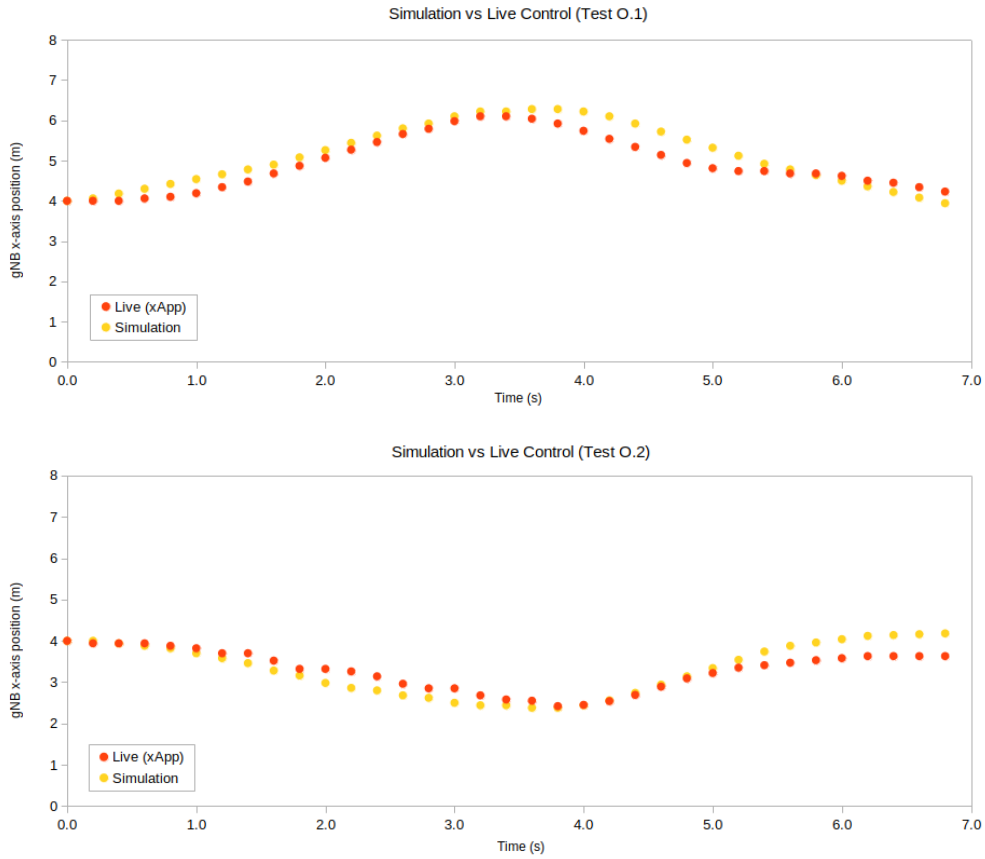


Figure 4.12: Use Case O. Comparison of gNB X-axis position over time in simulation and live deployment.

While the overall trends align closely, deviations between the simulation and live trajectories can be observed. These differences are primarily attributed to small inaccuracies in velocity estimation, especially due to the short time intervals between indication messages, which slightly affect the resulting state vector updates during live execution. Additionally, unit conversions between the internal CC-SIM representation and the standardized formats used in SM messages may introduce rounding effects.

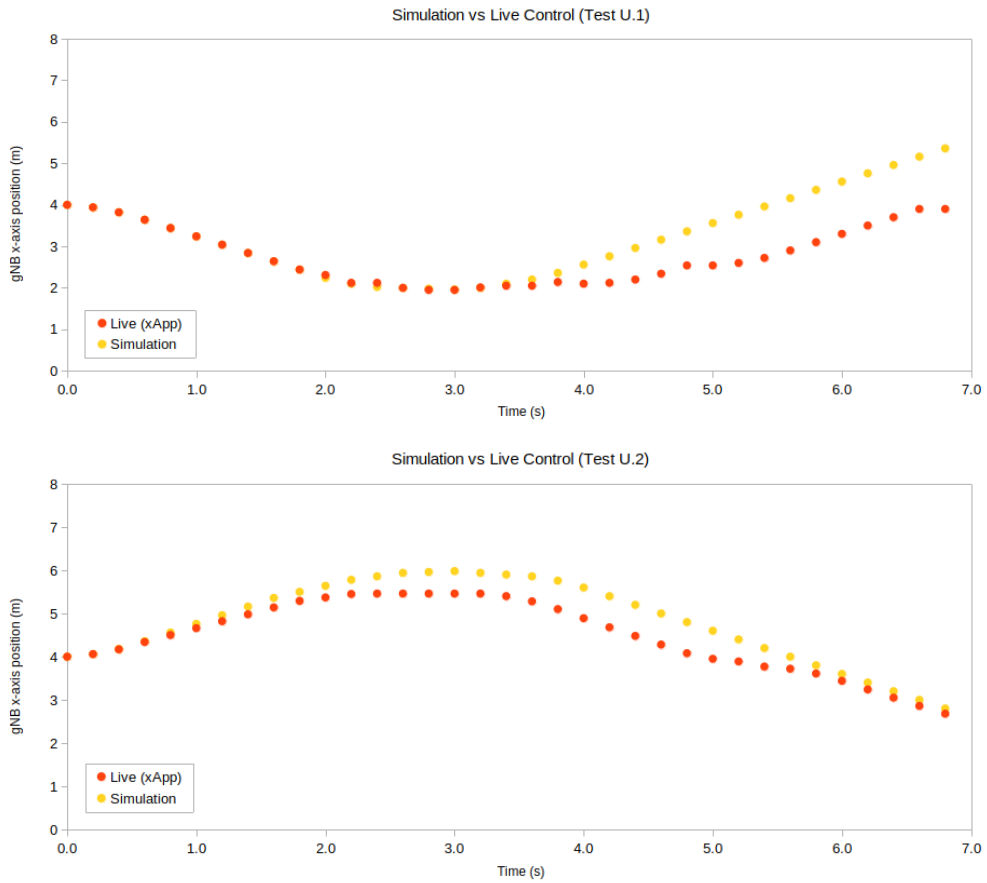


Figure 4.13: Use Case U. Comparison of gNB X-axis position over time in simulation and live deployment.

Despite these discrepancies, the strong alignment between the position curves confirms that the inference behavior of the trained neural network remains consistent when transitioning from simulation to live deployment via the xApp. This validates the correct development of simulation mode, establishing it as a reliable framework for assessing control performance, allowing developers to anticipate the gNB's real-time behavior using only the simulator, without requiring full system deployment. Such capability is especially valuable in early development stages, enabling performance evaluation and debugging without the need for a complete runtime environment. Moreover, the results obtained from simulation-controlled runs can serve as a reference to verify the correct implementation of the xApp logic, ensuring that inference decisions and control actions are preserved during live execution.

4.7 Discussion

This dissertation experimentally demonstrated the feasibility and effectiveness of a novel gNB mobility control framework for 5G networks. Deployed on top of a fully functional standalone

architecture, the system achieved its intended purpose–real-time intelligent positioning of a mobile gNB to preserve and optimize link quality in dynamic radio environments.

The integration of the O-RAN architecture was instrumental to the solution. The standardized E2 interface enabled seamless communication between the RAN and external network entities, allowing telemetry to be exported and interpreted by a custom xApp for mobility control. This modularity enabled non-invasive, flexible control over the gNB without requiring changes to the internal logic of the gNB or UE software stack.

Initial tests confirmed the correct operation of the OAI-based 5G network, including the RF-simulator's ability to emulate realistic radio links. Each subsystem was incrementally validated, starting from basic end-to-end connectivity, through Service Model registration and E2′ agent integration, and finally real-time channel emulation with CC-SIM. These steps ensured that the foundation for closed-loop control was robust and reliable.

Final validation tests confirmed the effectiveness of the RL-based controller in mitigating LoS obstructions and maintaining connection quality. The gNB exhibited adaptive behavior, proactively repositioning itself based on vision and spatial data to maintain line-of-sight with the UE. Quantitative results showed significant improvements in throughput and reduced obstruction times when compared to static deployments. Additionally, the comparison between simulation and live control confirmed that inference performance was preserved, validating the use of CC-SIM as a lightweight and accurate development environment.

Although the tests were conducted in a simulated setting, the modular and standards-compliant architecture ensures that the proposed framework can be seamlessly adapted for real-world deployments. The same xApp logic, agent interactions, and service model structure are fully reusable with physical hardware components, requiring only the deployment of the gNB and UE on dedicated platforms, and the integration with the CONVERGE interfaces–replacing the local E2′ interface–enabling direct communication with the physical infrastructure. Furthermore, once the full experimental setup is operational, the CC-SIM environment can be readily adapted into a digital twin, enabling real-time mirroring, visualization, and supervision of the physical system for monitoring, debugging, and closed-loop experimentation.

While the results are promising, some simplifications were made to constrain the scope of this work. In particular, the system was evaluated in a scenario with only one UE and one obstacle. This decision was taken considering the available time and complexity constraints of a master's dissertation. Nonetheless, the modular architecture and the design of the Service Models–particularly the VIS SM–were conceived with scalability in mind, including support for multi-UE and multi-obstacle scenarios, as the the SM structure is already capable of representing more complex spatial configurations. Extending the system to handle such scenarios would primarily require additional effort at the level of the xApp logic, particularly in processing and aggregating inputs from multiple vision-based messages. Furthermore, a more complex state representation and the adoption of an RL algorithm—or alternative learning method—capable of generalizing across higher-dimensional input spaces and partially observable environments would be necessary.

Similarly, the gNB in this work was restricted to movement along a single axis. This constraint

simplified both the simulation and control logic, while still allowing for meaningful evaluation of mobility-aware decision-making. However, the core principles of the controller—namely, sensing-driven policy inference and adaptive repositioning—are not inherently limited to one-dimensional motion. In principle, the same architecture and learning pipeline could be extended to two or three dimensions. This would require retraining the RL agent in a higher-dimensional action space and possibly redesigning the reward function to account for additional degrees of freedom and spatial trade-offs. These directions present a natural progression for future work.

Overall, the results demonstrate that the proposed system offers a practical and effective approach to mobility-aware RAN optimization in 5G deployments. The use of a modular O-RAN architecture allowed for the rapid development and deployment of a custom xApp, capable of interfacing with the RAN and applying intelligent control policies in real time. These findings reinforce the value of open, programmable RAN architectures in facilitating the integration of novel intelligent control strategies within real-world wireless systems.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This dissertation proposed and validated a modular framework for intelligent gNB mobility control in 5G networks, leveraging the flexibility of the O-RAN architecture and emulation capabilities. The system enables a mobile gNB to dynamically reposition itself in real time, with the goal of maintaining high-quality radio links in environments characterized by mobility, obstruction, and spatial uncertainty.

The architecture was designed around open interfaces and components. OpenAirInterface was used to deploy a 5G standalone network, including the 5G Core, gNB, and UE components. FlexRIC served as the near-RT RIC, hosting custom-designed Service Models and enabling E2-based communication with the RAN. A novel E2' agent structure was implemented to transmit position and visual data (POS and VIS messages) from the developed CONVERGE Chamber Simulator (CC-SIM) into the near-RT control loop. Moreover, an xApp was developed to infer control decisions using a trained Deep Q-Network (DQN) model, issuing movement commands to a mobile gNB.

The system was incrementally validated through a structured methodology: first, by ensuring the correct operation of the 5G network and inter-node communication; second, by validating the functioning of the new E2′ agents, including end-to-end subscription and control messaging; third, by conducting dynamic tests involving LoS obstruction scenarios, traffic-based performance measurements, and control evaluation.

Results showed that the trained reinforcement learning agent is able to successfully adapt the gNB's position in response to real-time occlusion risks, reducing LoS obstruction durations by up to 41.6% and preserving link throughput where static configurations failed. Comparison of simulation-mode and live-mode execution demonstrated that the inference logic remains consistent across deployment contexts, validating the use of CC-SIM as a development and testing platform.

Although the validation was conducted in a simulated environment, the framework is inherently designed for portability. The same Service Models, xApp logic, and agent infrastructure can

be used in real-world deployments by integrating with the CONVERGE hardware interfaces. Furthermore, once physical components are in place, the CC-SIM simulator can be repurposed into a digital twin, supporting real-time supervision, visualization, and safe experimentation.

In conclusion, this dissertation successfully achieved its objectives through three main contributions:

1. A novel multimodal O-RAN framework, incorporating CV-derived contextual information to enable proactive gNB mobility decisions, while supporting the maintenance of high-quality radio links and LoS connectivity.

2. CC-SIM, a custom simulation environment of the CONVERGE chamber, designed to emulate dynamic radio environments and generate synthetic multimodal data. CC-SIM supports three operational modes: 1) *training*, for reinforcement learning agents development; 2) *simulation*, for offline inference, validation, and visualization; and 3) *live testing*, for real-time integration with the O-RAN framework.

3. A novel xApp, leveraging multimodal data inputs and a trained Deep Reinforcement Learning model to improve decision-making processes and overall RAN performance.

The proposed solution lays the groundwork for future work on intelligent RAN mobility, particularly in scenarios involving multi-user environments, complex obstacle configurations, and large-scale field validation. It also provides a reproducible platform for testing and benchmarking other ML-based RAN control strategies.

5.2 Future Work

While the present dissertation successfully achieved all the proposed objectives, demonstrating the feasibility and effectiveness of intelligent gNB mobility control within an O-RAN environment, several opportunities exist to expand and refine the system for broader applicability and real-world integration.

Integration with a Real-World Testbed

A natural continuation of this dissertation is the deployment of the full control framework in a physical testbed environment. The modular design of the architecture, particularly the xApp logic, E2' agents, and service model infrastructure, allows for seamless reuse of the control plane with physical hardware. This transition requires only the replacement of the simulated gNB and UE components with real-world radio units and the corresponding integration of the CONVERGE interfaces, which are already designed for compatibility with FlexRIC. This step will enable real-time validation of the controller in practical scenarios, accounting for wireless impairments, sensor noise, and real-world dynamics.

CC-SIM Insights Toward CONVERGE Simulator Development

The CC-SIM environment, implemented as a lightweight Python prototype, served as a crucial tool for validating the proposed system and training the reinforcement learning controller.

Although the CONVERGE Simulator was not yet developed at the time of this dissertation, CC-SIM offers valuable insights that can directly inform its future design, specifically the CONVERGE 3D-S node. Lessons learned from CC-SIM's architecture, data handling, and control integration may serve as a foundation for building a more scalable and robust simulation framework. In particular, future development should prioritize refactoring CC-SIM into a high-performance C-based implementation, which will allow native compatibility with the FlexRIC runtime and enable direct embedding of E2' agents inside the simulator. This change would eliminate the need for intermediate data serialization and reduce interface latency, allowing tighter feedback loops and real-time testing under more complex scenarios.

Moreover, an appropriate high-performance 3D rendering engine should be selected to replace the current visualization layer. This would not only improve graphical fidelity and simulation clarity but also support the development of a real-time digital twin – an interactive, synchronized virtual representation of the physical testbed. This digital twin will enhance debugging, system monitoring, and user interaction, while providing a visual platform for demonstrations, training, and future scenario planning within the CONVERGE project.

Generalization of the RL agent

The RL agent employed in this dissertation was trained for a specific set of scenarios with a limited number of objects. Although effective within this scope, its generalization capability remains constrained. Future work should aim to develop more robust RL agents capable of handling a broader variety of movement patterns and object counts. This will involve training with more diverse simulated scenarios, containing more obstacles and/or more UEs. In such scenarios, RF data captured from the xApp will be an essential control input, aiming to maximize long-term link quality across multiple users while dynamically avoiding LoS blockages and minimizing interference.

Visual Perception and Real-Time Object Tracking

One of the key future challenges in real-world deployment lies in the accuracy and reliability of visual perception. In this dissertation, object positions and visibility were obtained deterministically within the simulator. In practice, however, object detection and tracking depend on the performance of onboard or external video cameras, and are subject to errors due to occlusions, lighting conditions, and sensor limitations. Ensuring that the system can accurately label and track objects in real time will be crucial for maintaining control policy stability. As such, improving perception accuracy will become a key focus area in future investigations, particularly as the system scales to more complex and dynamic environments.

References

[1] CONVERGE Project Consortium. *CONVERGE Project - Vision-Enabled Mobile Communications for 6G*. Accessed: 2025-06-24. 2023. URL: https://converge-project.eu/.

[2] Filipe B. Teixeira et al. "CONVERGE: A Vision-Radio Research Infrastructure Towards 6G and Beyond". In: *2024 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. 2024, pp. 1015–1020. DOI: 10.1109/EuCNC/6GSummit60053.2024.10597071.

[3] Seungnyun Kim et al. "Role of Sensing and Computer Vision in 6G Wireless Communications". In: *arXiv preprint arXiv:2405.03945* (2024).

[4] International Telecommunication Union (ITU). *Recommendation ITU-R M.2083: IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond*. Tech. rep. International Telecommunication Union, 2015. URL: https://www.itu.int/rec/R-REC-M.2083-0-201509-I/en.

[5] International Telecommunication Union (ITU). *Recommendation ITU-R M.2160: Framework and overall objectives of the future development of IMT for 2030 and beyond*. Tech. rep. International Telecommunication Union, 2023. URL: https://www.itu.int/rec/R-REC-M.2160-0-202311-I/en.

[6] Jordan M. Thomas et al. "Quantum teleportation coexisting with classical communications in optical fiber". In: *Optica* 11.12 (Dec. 2024), pp. 1700–1707. DOI: 10.1364/OPTICA.540362. URL: https://opg.optica.org/optica/abstract.cfm?URI=optica-11-12-1700.

[7] Cheng-Xiang Wang et al. "On the Road to 6G: Visions, Requirements, Key Technologies, and Testbeds". In: *IEEE Communications Surveys & Tutorials* 25.2 (2023), pp. 905–974. DOI: 10.1109/COMST.2023.3249835.

[8] 3rd Generation Partnership Project (3GPP). *5G System Overview*. Accessed: December 2024. 2022. URL: https://www.3gpp.org/technologies/5g-system-overview.

[9] Bubbleran Documentation Team. *RAN Evolution*. Accessed: December 2024. 2024. URL: https://bubbleran.com/docs/tutorials/studio/lessons/ran-evolution/.

[10] Liljana Gavrilovska, Valentin Rakovic, and Daniel Denkovski. "From cloud RAN to open RAN". In: *Wireless Personal Communications* 113 (2020), pp. 1523–1539.

[11] European Telecommunications Standards Institute (ETSI). *Technical Specification 138 401 V15.2.0: General Packet Radio Service (GPRS); Service Description; Stage 2*. Technical Specification TS 138 401 V15.2.0. Accessed: 2024-12-22. ETSI, 2024. URL: https://www.etsi.org/deliver/etsi_ts/138400_138499/138401/15.02.00_60/ts_138401v150200p.pdf.

[12] O-RAN Alliance. *O-RAN Work Group 1 (Use Cases and Overall Architecture). O-RAN Architecture Description v.12.0*. 2024. URL: https://orandownloadsweb.azurewebsites.net/specifications.

[13] O-RAN Alliance. *O-RAN Near-Real-time RAN Intelligent Controller Architecture & E2 General Aspects and Principles 2.02*. Tech. rep. O-RAN.WG3.E2GAP-v02.02. https://www.o-ran.org/specifications. O-RAN Alliance, July 2022.

[14] MOSAIC5G Project. *FlexRIC with E42 Interface Update*. https://gitlab.eurecom.fr/mosaic5g/flexric/-/tree/e42-update. Accessed: 2025-06-24. 2023.

[15] Navid Nikaein et al. "OpenAirInterface: A flexible platform for 5G research". In: *ACM SIGCOMM Computer Communication Review* 44.5 (2014), pp. 33–38.

[16] Robert Schmidt, Mikel Irazabal, and Navid Nikaein. "FlexRIC: An SDK for next-generation SD-RANs". In: *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*. 2021, pp. 411–425.

[17] Yuxi Li. "Deep reinforcement learning: An overview". In: *arXiv preprint arXiv:1701.07274* (2017).

[18] Dinh Thai Hoang et al. *Deep Reinforcement Learning for Wireless Communications and Networking: Theory, Applications and Implementation*. John Wiley & Sons, 2023.

[19] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[20] Ahmed Alkhateeb et al. "DeepSense 6G: A large-scale real-world multi-modal sensing and communication dataset". In: *IEEE Communications Magazine* 61.9 (2023), pp. 122–128.

[21] Muhammad Alrabeiah et al. "ViWi: A deep learning dataset framework for vision-aided wireless communications". In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE. 2020, pp. 1–5.

[22] Sunwoo Kim, Yongjun Ahn, and Byonghyo Shim. "Computer vision-aided beamforming for 6G wireless communications: Dataset and training perspective". In: *ICC 2024-IEEE International Conference on Communications*. IEEE. 2024, pp. 672–677.

[23] Yu Tian, Gaofeng Pan, and Mohamed-Slim Alouini. "Applying deep-learning-based computer vision to wireless communications: Methodologies, opportunities, and challenges". In: *IEEE Open Journal of the Communications Society* 2 (2020), pp. 132–143.

[24] Muhammad Alrabeiah, Andrew Hredzak, and Ahmed Alkhateeb. "Millimeter wave base stations with cameras: Vision-aided beam and blockage prediction". In: *2020 IEEE 91st vehicular technology conference (VTC2020-Spring)*. IEEE. 2020, pp. 1–5.

[25] Seungnyun Kim et al. "Vision-aided positioning and beam focusing for 6G terahertz communications". In: *IEEE Journal on Selected Areas in Communications* (2024).

[26] Gouranga Charan, Muhammad Alrabeiah, and Ahmed Alkhateeb. "Vision-aided 6G wireless communications: Blockage prediction and proactive handoff". In: *IEEE Transactions on Vehicular Technology* 70.10 (2021), pp. 10193–10208.

[27] Leonardo Bonati et al. "Intelligent closed-loop RAN control with xApps in OpenRAN gym". In: *European Wireless 2022; 27th European Wireless Conference*. VDE. 2022, pp. 1–6.

[28] Leonardo Bonati et al. "OpenRAN gym: An open toolbox for data collection and experimentation with AI in O-RAN". In: *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2022, pp. 518–523.

[29] Mohammed MH Qazzaz et al. “Machine learning-based xApp for dynamic resource allocation in O-RAN networks”. In: *arXiv preprint arXiv:2401.07643* (2024).

[30] Joao F Santos et al. “Managing O-RAN Networks: xApp Development from Zero to Hero”. In: *arXiv preprint arXiv:2407.09619* (2024).

[31] Weihua Xu et al. “Computer vision aided mmWave beam alignment in V2X communications”. In: *IEEE Transactions on Wireless Communications* 22.4 (2022), pp. 2699–2714.

[32] Gonçalo Queirós et al. “Autonomous Control and Positioning of a Mobile Radio Access Node Employing the O-RAN Architecture”. In: *2024 19th Wireless On-Demand Network Systems and Services Conference (WONS)*. IEEE. 2024, pp. 25–28.

[33] MOSAIC5G Project. *FlexRIC Wiki: Create a Service Model*. https://gitlab.eurecom.fr/mosaic5g/flexric/-/wikis/Create-a-service-model. Accessed: June, 2025. 2025.

[34] OpenAirInterface Project. *OpenAirInterface 5G: RF Simulator*. https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/2025.w25/radio/rfsimulator/README.md. Accessed: June, 2025. 2025.

[35] OpenAirInterface Project. *OpenAirInterface 5G: Channel Simulation Documentation*. https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/2025.w25/openair1/SIMULATION/TOOLS/DOC/channel_simulation.md. Accessed: June, 2025. 2025.

[36] The Matplotlib Development Team. *Matplotlib: Visualization with Python*. May 2025. DOI: 10.5281/zenodo.15375714. URL: https://doi.org/10.5281/zenodo.15375714.

[37] Greg Brockman et al. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[38] Morvan Zhou. *PyTorch DQN Reinforcement Learning Tutorial*. https://github.com/MorvanZhou/PyTorch-Tutorial/blob/master/tutorial-contents/405_DQN_Reinforcement_learning.py. Accessed: 2025-06-12. 2018.

[39] Universal Robots A/S. *UR10e Technical Specification*. Tech. rep. Universal Robots A/S, Sept. 2024. URL: https://www.universal-robots.com/products/ur10-robot/.

[40] OpenAirInterface and OpenAI Cellular. *FlexRIC Documentation - OpenAI Cellular Workshop (OAIC 2024)*. https://openaicellular.github.io/oaic/OAIC-2024-Workshop-oai-flexric-documentation.html. Accessed: 2025-06-25. 2024.

Appendix A

Additional Figures

A.1 Use Case Snapshots

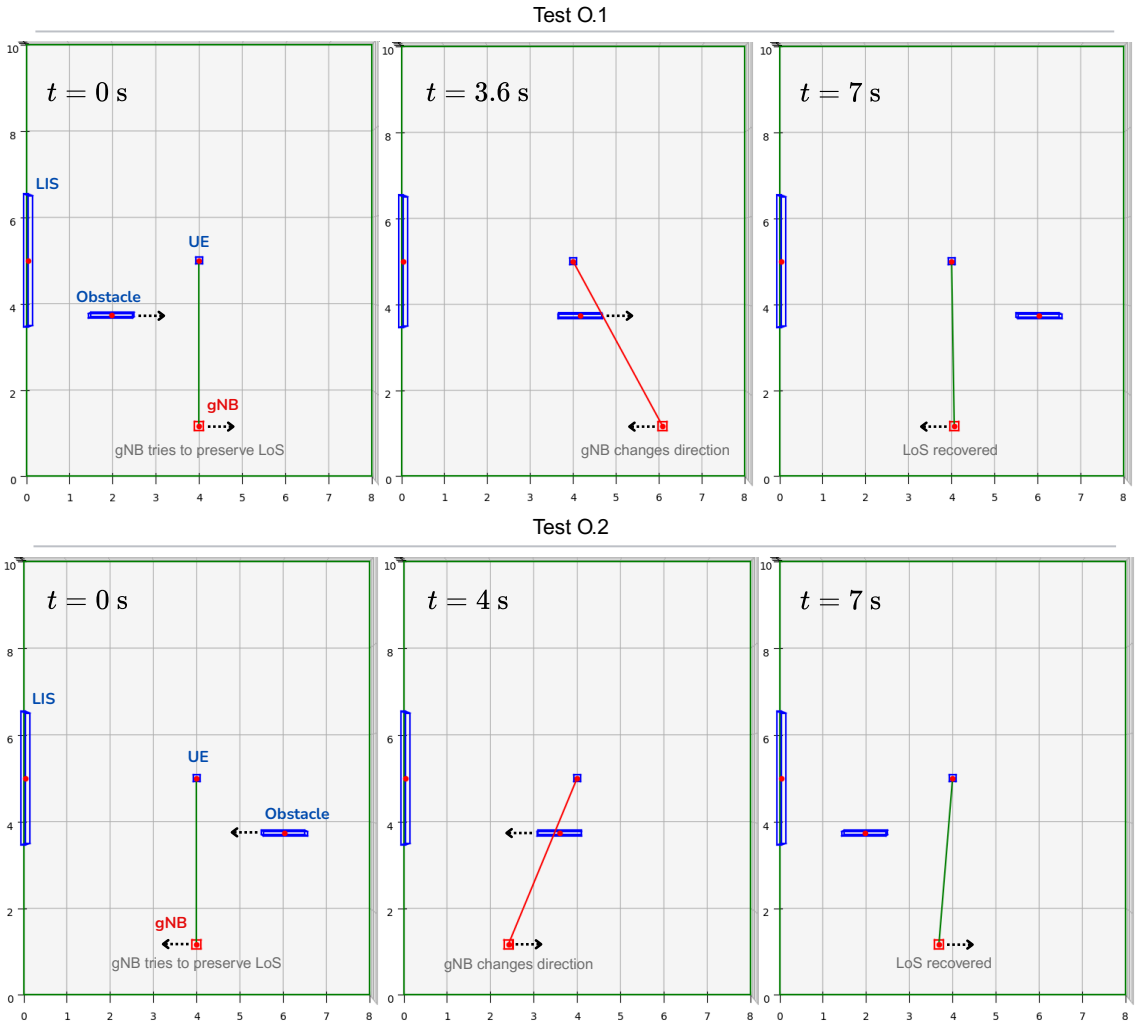


Figure A.1: Temporal snapshots of object positions during tests O.1 (top) and O.2 (bottom). Note that velocity vectors are not scaled.

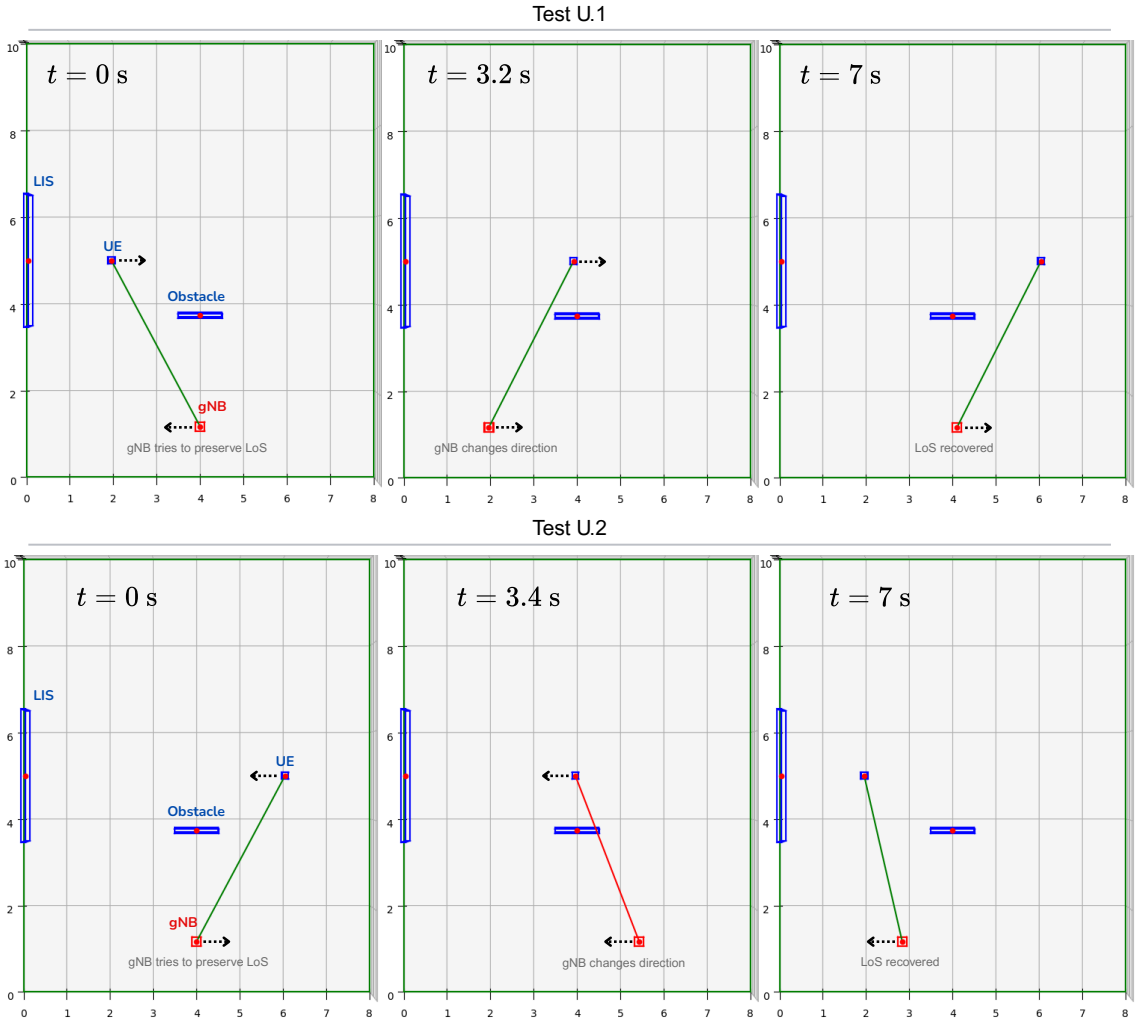


Figure A.2: Temporal snapshots of object positions during tests U.1 (top) and U.2 (bottom). Note that velocity vectors are not scaled.

Appendix B

Reinforcement Learning Agent Training Evaluation

B.1 Impact of Training Episode Count on Convergence

The results presented in this appendix were obtained after the completion of the main dissertation work. They evaluate how the number of training episodes influences the performance of the RL agent using the final environment and reward formulation described in Section 3.4. For each episode count considered, five independent agents were trained and subsequently evaluated using the same episode structure as during training.

In the controlled environment used for training, the UE was fixed along the Y-axis and the gNB was allowed to move only along the X-axis. For these specific positions, the theoretical maximum instantaneous reward–corresponding to the gNB being in LoS and positioned at the minimum allowable distance–was $r_{\max} = 1.39$. Given that each episode spans 3000 steps, the maximum achievable cumulative reward per episode, under ideal conditions, would be $R_{\max} = 4170$. However, such ideal conditions are unattainable in practice due to the episodic formulation, where LoS obstructions are inevitable at certain steps, preventing the agent from maintaining the optimal position throughout the entire episode. For comparison, a static gNB placed at the center of the environment achieved a baseline cumulative reward of $R_{\text{static}} = 1911$, representing a non-adaptive policy that remains vulnerable to occlusions. This provides a meaningful lower bound for evaluating the performance gains achieved by the trained RL agent.

Figure B.1 shows the average cumulative reward R achieved during evaluation for each tested number of training episodes. Remarkably, the results demonstrate that the agent is able to converge to effective policies with as few as **two training episodes**. This is attributable to the relatively low complexity of the environment, characterized by an 11-dimensional state vector, a 3-action discrete space, and movement constrained to a single spatial dimension (x-axis).

Based on these results, the final model used in this work—trained with three episodes—was deemed sufficient, as it consistently demonstrated an understanding of the environment and successfully learned policies that yield strong performance.

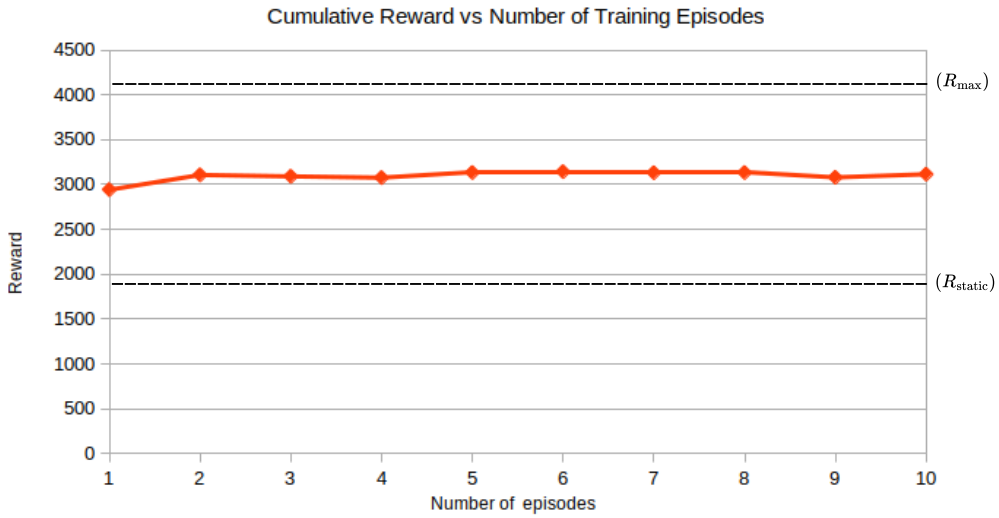


Figure B.1: Average cumulative reward per evaluation episode for agents trained with different numbers of training episodes. Five agents were trained per configuration. Convergence is achieved rapidly due to the low complexity of the environment.