D4.6 REPORT ON 6G-NTN RADIO CONTROLLER

Final Version

v.1.1

Work package	WP 4
Task	Task 4.2
Due date	31/12/2025
Submission date	12/01/2026
Deliverable lead	CTTC
Version	1.1
Authors	Husnain Shahid, Miguel A. Vazquez (CTTC) Ted Hsu , Bernardo Camajori Tedeschini, Ji Lianghai, (QCOM), Carla Amatetti (UNIBO) Amel Tibhirt (ORA)
Reviewers	Sorya Tong (Thales SIX) Estefania Recayte (DLR)
Abstract	This deliverable provides the final report on 6G-NTN radio intelligent controller (RIC) leveraging the AI/ML aspects to dynamically manage the resource allocation considering the challenges such as dynamic user traffic, unpredictable channel conditions and limited number of available resources etc. In this aspect, the contribution comprises on proposing the integration of AI/ML functions into disaggregated and distributed TN-NTN architecture to flexibly fulfill each network function requirements. Moreover, this report further extends towards the optimization of potential resource network functions identified in D4.2, by integrating self-adaptive AI models capable of continual learning in response to dynamic network conditions. To ensure alignment, the network function utilizes the architecture definition and system payload from WP3. The outcomes reveal the overall significant gain in KPIs.
Keywords	Radio Resource Management (RRM), RAN Intelligent Controller (RIC), Artificial Intelligence (AI), Open RAN (O-RAN), AI-RAN.

Grant Agreement No.: 101096479
Call: HORIZON-JU-SNS-2022
Topic: HORIZON-JU-SNS-2022-STREAM-B-01-03
Type of action: HORIZON-JU-RIA

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	01/10/2024	ToC modifications and locked	Husnain Shahid , Miguel A. Vazquez (CTTC) Carla Amatetti (UNIBO) Ji Lianghai, (QCOM)
V0.2	07/01/2025	Open RAN interfaces and architecture	Husnain Shahid (CTTC)
V0.3	01/05/2025	Split architecture for each network function and the corresponding requirements	Husnain Shahid, Miguel A. Vazquez (CTTC) Carla Amatetti (UNIBO) Ji Lianghai, (QCOM) Amel Tibhirt (ORA)
V0.4	01/11/2024-20/10/2025	Network function simulations	Husnain Shahid (CTTC) Carla Amatetti (UNIBO) Ted Hsu , Bernardo Camajori Tedeschini, Ji Lianghai, (QCOM) Amel Tibhirt (ORA)
V0.5	24/10/2025	Polish and align the contribution	Husnain Shahid (CTTC)
V0.6	31/10/2025	Release to review	Husnain Shahid, Miguel A. Vazquez (CTTC)
V0.7	15/11/2025	Reviews Completed	Sorya Tong (TH-SIX), Estefania Recayte (DLR)
V0.8	02/12/2025	Reviews Addressed	Husnain Shahid (CTTC) Carla Amatetti (UNIBO) Ted Hsu , Bernardo Camajori Tedeschini, Ji Lianghai, (QCOM) Amel Tibhirt (ORA)
V1.0	10/12/2025	Final Version Released	Husnain Shahid (CTTC)
V1.1	25/02/2026	PDF Conversion Issues Fixed	Husnain Shahid (CTTC)

DISCLAIMER

Co-funded by the European Union

6GSNS

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

6G-NTN (*6G Non Terrestrial Network*) project has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101096479. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. This work has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI).

COPYRIGHT NOTICE

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	**to specify R, DEM, DEC, DATA, DMP, ETHICS, SECURITY, OTHER***	
Dissemination Level		
PU	*Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)*	✓
SEN	*Sensitive, limited under the conditions of the Grant Agreement*	
Classified R-UE/ EU-R	*EU RESTRICTED under the Commission Decision No2015/ 444*	
Classified C-UE/ EU-C	*EU CONFIDENTIAL under the Commission Decision No2015/ 444*	
Classified S-UE/ EU-S	*EU SECRET under the Commission Decision No2015/ 444*	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.

EXECUTIVE SUMMARY

D4.6 is one of the deliverables of the task T4.2 in work package (WP) 4 of 6G-NTN project which is set to provide a final comprehensive report on the design of artificial intelligence (AI)-enabled radio access network (RAN) intelligent controller (RIC) to effectively manage the resource allocation in terrestrial networks (TN) and non-terrestrial networks (NTN). It is envisaged that efficient radio resource management (RRM) has paramount significance in integrated TN-NTN networks to ensure quality of service (QoS) requirements and overall system optimization by dynamically allocating the available resources leveraging the challenges such as non-uniform user traffic demand and dynamic user densities as a function of time, and heterogeneous and unpredictable channel conditions etc., just to mention a few. In this context, the urge of adaptive resource allocation given such challenging conditions is significant both in non-real time (non-RT) and near-real time (n-RT) scenarios in order to ensure global coverage, seamless connectivity, and ubiquitous communication requirements. To this end, this report provides comprehensive analysis regarding adaptive resource management and allocation where the solution proposed herein covers the broaden domains, spanning from the detailed simulation of the prospective RIC resource network functions that were originally identified to be optimized, to the end of studying artificial intelligence/machine learning (AI/ML) architectural aspects. These aspects explore the AI/ML framework deployment scenarios corresponding to these identified network functions and subsequently expand the contribution to potential simulation results of deployed AI/ML optimization algorithms.

Recalling from D4.2, the radio resource control (RRC) of 6G-NTN hosts the following AI/ML based potential network functions:

- Traffic Off-Loading (TOL)
- Fractional Frequency Reuse (FFR)
- Beam Hopping (BH)
- Link Quality Prediction (LQP)

Taking this into account, the overall document encompasses following critical elements:

- To be specific, at first, it provides the global qualitative analysis of deploying AI/ML agents in 6G-NTN Distributed Architecture defined in WP3. It mainly involves assessing the feasibility and suitability of hosting the AI/ML functions in split architecture, responsible for training, management, and inference located either on the feeder or service satellites.
- Subsequently, it indulges into in-depth exploration of the low-level architecture details relevant to the distributed split processing, open interfaces requirements and the deployment of AI/ML agents within the RIC component considering the open RAN (O-RAN) and AI-RAN as baseline, corresponding to each defined network function.
- Following this, the report offers a thorough characterization of each network function with its system level description, based on WP3 components to ensure system-wide alignment. It further formulates the resource allocation optimization problem, articulating the objective functions and delineating the system constraints that govern the solution space.

- The final section provides a comprehensive set of solutions to the defined resource optimization problems, encompassing both single agent and multi-agent AI/ML approaches. This provides an examination of how distributed intelligence can be leveraged to achieve efficient and scalable resource management across the network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

3D	3 Dimensions	**FS**	Feeder Satellite
3GPP	3rd Generation Partnership Project	**FSPL**	Free Space Path Loss
AI	Artificial Intelligence	**GA**	Genetic Algorithm
APIs	Application Programming Interfaces	**GEO**	Geostationary Earth Orbit
AWGN	Additive While Gaussian Noise	**GPU**	Graphic Processing Unit
BH	Beam Hopping	**GRU**	Gated Recurrent Unit
BS	Base Stations	**HAPs**	High-Altitude Platforms
CDF	Cumulative Distribution Function	**HLS**	High Layer Split
CN	Core Network	**I-DQL**	Independent Deep Q-Learning
CP	Cyclic Prefix	**IFFT**	Inverse Fast Fourier Transform
CSI	Channel State Information	**ISD**	Inter-Site Distance
CTDE	Centralized Learning and Decentralized Execution	**ISL**	Inter-Satellite Link
CU	Centralized Unit	**KPIs**	Key Performance Indicators
CV	Cross Validation	**KPMs**	Key Performance Measurements
DCI	Downlink Channel Information	**LCM**	Life Cycle Management
DL	Downlink	**LEO**	Low Earth Orbit
DNN	Deep Neural Network	**LLS**	Lower-Layer Split
DQN	Deep Q Network	**LOS**	Line of Sight
DQL	Deep Q Learning	**LQP**	Link Quality Prediction
DRA	Direct Radiating Arrays	**MAC**	Medium Access Control
DRL	Deep Reinforcement Learning	**MARL**	Multi-Agent Reinforcement Learning
DRA	Direct Radiating Array	**MEC**	Multi Access Edge Computing
DU	Distributed Unit	**MEO**	Medium Earth Orbit
E2E	End-to-End	**MSE**	Mean Square Loss
ENU	East-North-Up	**n-RT**	Near Real Time
FB	Fixed Beamwidth	**NCD**	Normalized Capacity Deviation
FFR	Fractional Frequency Reuse	**NN**	Neural Network
FR	Frequency Reuse	**NLOS**	Non-Line of Sight

non-RT	non-Real Time
NS	No Signal
NTN	Non-Terrestrial Networks
NW	Network
O-RAN	Open RAN
OTT	Over-The-Top
PHY	Physical Layer
PRB	Physical Resource Block
QoS	Quality of Service
RAN	Radio Access Network
RATs	Radio Access Technologies
ReLu	Rectified Linear Unit
RF	Radio Frequency
RIC	RAN Intelligent Controller
RL	Reinforcement Learning
RLC	Radio Link Control
RMa	Rural Macro
RRC	Radio Resource Control
RRM	Radio Resource Management
RRUR	Radio Resource Usage Ratio
RSRP	Reference Signal Received Power
RU	Radio Unit
SBR	Shooting and Bouncing Rays
S-FFR	Strict Fractional Frequency Reuse
SINR	Signal to Interference and Noise Ratio
SoC	System on Chip
SS	Service Satellite
TD	Temporal-Difference
TN	Terrestrial Networks
TOL	Traffic Off-Loading
UE	User Equipment
UL	Uplink
UP	User-Plane
UTs	User Terminals
VNFs	Virtual Network Functions
WP	Work Package
ZSM	Zero-Touch Network and Service Management

1. INTRODUCTION

Before delving into the details of radio access network (RAN) intelligent controller (RIC) functions, this deliverable proposes, at first, a concise discussion about the overall 6G-NTN (6G-Non-Terrestrial Networks) project objectives that are intended to be accomplished throughout the project duration. This helps to better understand the context and the purpose of this deliverable, which provides the final contribution by designing Artificial Intelligence (AI)-enabled RIC in the realm of integrated Terrestrial Network (TN) and NTN. Since the inputs are acquired from other work packages to fulfil the design requirements of RIC, this section further extends the discussion by identifying the interconnection of this deliverable with the tasks corresponding to other work packages in order to better understand the correlation among them. In essence, the interconnection is with the use cases already defined in WP2 [1], the antenna definition [2], architecture definition from WP3 [3] and the open datasets provided in WP4 [4] that are utilized to train the AI network in RIC. Subsequently, the overall organization of the document is illustrated.

1.1 GOALS OF 6G-NTN PROJECT

The broad ambition of the 6G-NTN project is the full integration of NTN component into future 6G infrastructure which allows to meet consumer expectations in terms of performance and integrity enhancement, seamless connectivity in mobility, resilience in service corresponding to traffic variations etc. This part of the deliverable briefly discusses the potential goals and ambitions of 6G-NTN in a more technical way that are expected to be executed during the project duration. The potential objectives of the project are the following:

- The identification and proposal of use cases as potential candidates of 6G-NTN.
- Defining the user and technical requirements of 6G-NTN project.
- Designing 3D (multidimension) multilayered architecture comprised of interconnected terrestrial nodes, space nodes and airborne flying nodes.
- Designing more effective terminals in terms of cost, size and power consumption which would be compatible to operate with both TN and NTN access.
- Designing the flexible software defined payload intended to optimize the platform and payload resources.
- Designing and evaluation of flexible waveform using data-driven approaches. The flexibility is in terms of supporting terrestrial and non-terrestrial deployments and extends the capability of coverage into light indoor environment.
- Development of AI-enhanced RIC to provide effective Radio Resource Management (RRM) solutions, considering the multi-frequency, multi-constellation, and TN-NTN connectivity aspects to foster the high availability values and heterogenous traffic demand. D4.6 is one of the contributions to the critical design of RIC.
- Another aspect is evaluating the interference scenarios and their impact on radio access network. Based on the impact, spectrum management techniques will be proposed.
- Taking into account the latency requirements defined in Work Package 2 (WP2), another aim is to define accurate and reliable Position and Timing solutions for 6G-NTN. In

particular, it will define the enhanced NTN based user equipment (UE) positioning mechanism for regulated services.

- Providing virtualized and cloud native architectures for 6G core network. The objective is also to define the open interfaces and application programming interfaces (APIs) to configure, monitor and orchestrate the NTN edge and core networks.
- In addition, based on the assessment of cybersecurity threats and vulnerabilities for 6G-NTN, there is a need to provide solutions to counter these vulnerabilities. In this regard, a goal is to propose solutions relying on cloud native architecture and open interfaces mainly focused on securing the communication scheme.
- The final objective is to develop solid strategic approaches for effective communication, dissemination and community building at European, national, and international levels. This includes engagement and coordination with the target stakeholders from mobile, satellite and vertical industries having an interest in taking up the 6G-NTN technologies and concepts as developed.

1.2 CONTEXT

The efforts to efficiently adapt the resources in NTN are becoming a priority and requiring attention, both in non-Real Time (non-RT) and near Real Time (n-RT) scenarios. It is vitally important to enhance overall system performance while providing reliable and cost-effective services. This requirement becomes even more significant in satellite integrated networks that introduce additional complexities due to the variability of satellite links, non-uniform user traffic demand, dynamic user densities with time, and heterogenous channel conditions etc. Classical and heuristic Radio Resource Control (RRC) approaches may often struggle to effectively adapt to such heterogeneity. Furthermore, the system related constraints and the number of resources to manage at the larger scale, the classical methods exhibit limited efficacy in addressing these complexities, thus leading to sub-optimal solutions and degraded performance.

In this context, AI-driven approaches hold significant potential as key contributors to resource optimization process. It has the ability to learn complex patterns from data, anticipate network dynamics, and autonomously make decisions in both centralized and distributed environments. In particular, the advent of flexible and disaggregated architectures and the evolution of O-RAN and AI-RAN paradigms enable the flexible deployment of AI at the edge of the network or at ground level, where it can optimally react based on the resource allocation problem requirements.

Therefore, D4.6 is explicitly aligned with one of the core objectives of 6G-NTN project in terms of designing AI-driven RRM strategies that leverage flexible and disaggregated architecture solutions. The focus is on enabling AI-powered functions within both the non-RT and near RT RIC platforms to support dynamic, context aware and efficient resource allocation. This deliverable thoroughly investigates the specific network functions that can be augmented or fully managed by AI and identifies which of these network functions components are most suitable to be deployed and integrated in RIC platform to perform the resource allocation optimization. The detailed design methodologies, architectural considerations and optimization strategies are further elaborated in subsequent sections.

1.3 PURPOSE

In point of fact that efficient RRM has paramount significance to ensure Quality of Service (QoS) requirements and overall system optimization by dynamically allocating the available resources considering the aforementioned challenges. Those QoS requirements become even more stringent particularly due to ever growing user demands, global coverage, seamless connectivity, and ubiquitous communication requirements. These requirements might be fulfilled to a great extent by the intelligent decision-making capabilities of RRC, capable of continuously analyzing network conditions, learning and identifying complex patterns and features to enhance overall network performance.
To this end, this delivery provides the complete contribution towards AI-enabled RIC capable of optimizing the RMM in integrated TN-NTN framework. In particular, the focus relies on identifying the potential resource network functions, for instance,

- **Traffic Off-Loading (TOL):** The use of TOL allows to balance the load between TN and NTN depending on the dynamic user traffic i.e. by off-loading traffic to NTN. This leads to an increase of the QoS during high network load conditions and a reduced TN energy consumption when the traffic request is lower.

- **Fractional Frequency Reuse (FFR):** FFR in multibeam satellite system aims to minimize capacity deviation by efficiently utilizing the spectrum resources. This resource network function leverages the dynamic user traffic demand and density information as a function of time.

- **Beam Hopping (BH):** Beam hopping aims to sequentially activate the beams in time following a predetermined or adaptive hopping schedule. Unlike traditional fixed multi-beam systems, where all spot beams remain active simultaneously, BH systems allocate radiofrequency (RF) power and spectrum only to beams corresponding to active user demand.
  Key characteristics include:
  - **Temporal multiplexing** – Beams are activated in scheduled time slots.
  - **Spatial adaptability** – Coverage is shifted according to traffic distribution.
  - **Resource efficiency** – Power and spectrum are concentrated on active beams, thereby improving utilization.

- **Link Quality Prediction:** A distinguishing characteristic of NTN is the fast movement of Low Earth Orbit (LEO) satellites, which necessitates frequent satellite-switching by a UE. To serve the UE continuously during the switch, online radio measurement procedure in the legacy solution may cause certain inefficiencies and bottlenecks, such as power consumption, spectrum usage, potential service interruptions due to measurement gaps, early coordination at network side. Thus, to improve the efficiency during NTN mobility, an AI-based mechanism is considered to predict the UE's radio channel condition in the time-and-spatial domain to overcome the aforementioned inefficiencies.

The proposed resource network functions are set to be thoroughly investigated from the architectural aspects, to identify the appropriate AI-enabled algorithms in optimizing the overall resource management.

1.4 D4.6 RELATION TO OTHER WORK PACKAGES IN 6G-NTN

The dependencies of all work packages (WP) on each other are illustrated in **Figure 1-1**. Since this deliverable is one of the tasks of WP4 related to 6G-NTN Radio Access Technologies (RATs). It requires to take inputs from the tasks of other WPs and similarly provides outputs to accomplish the other WPs.

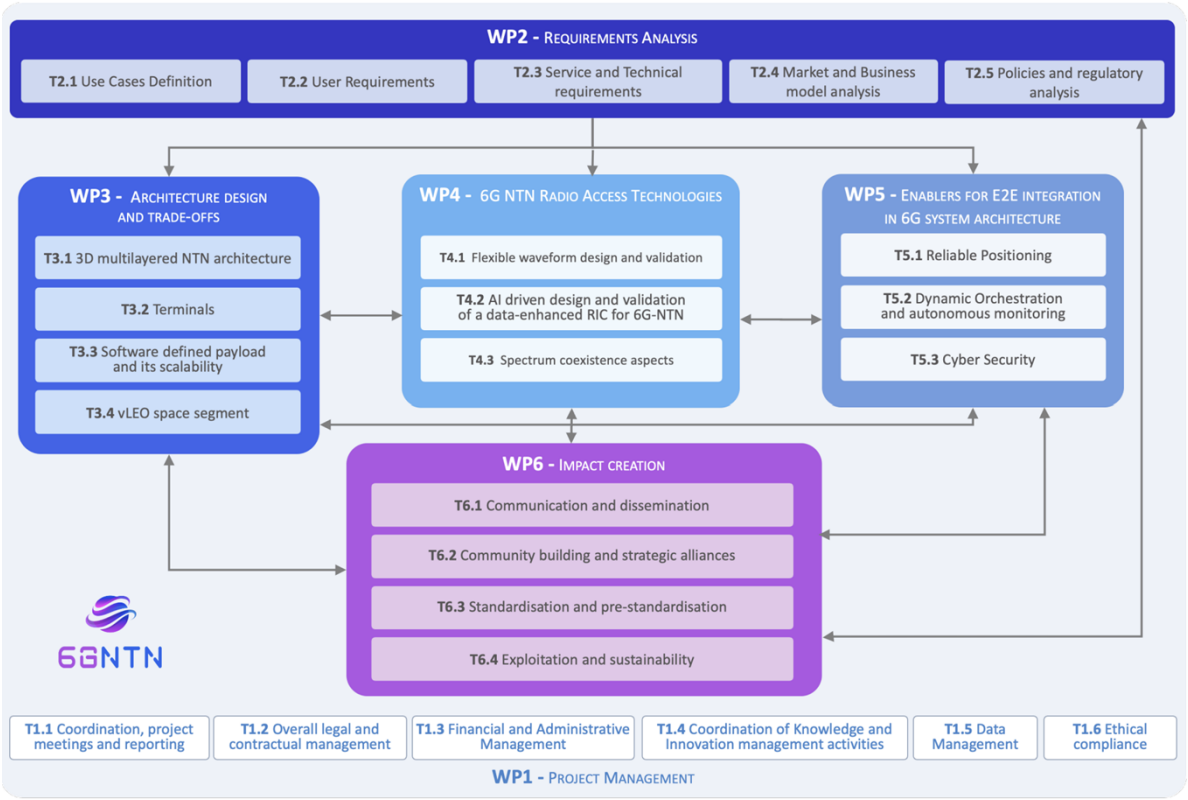

FIGURE 1-1 6G-NTN OVERALL WORK PACKAGES ORGANIZATIONS AND THEIR DEPENDENCIES

The organization of deliverable D4.6 with other defined tasks are explicitly given as,

- D4.6 is dependent on the inputs from WP2, related to the provision of service and technical requirements, D4.2 [5] related to initial definition of network functions and D4.3 [4] related to open datasets to perform the management of NTN resources for potential use cases.
- D4.6 is explicitly incorporating 3D multilayered NTN architecture investigated in T3.1, specifically in D3.5 [3] and antenna payload investigated in T3.3 of WP3, specifically in D3.3 as baseline, in order to identify the options for deploying the NTN RIC and the overall system description, respectively.
- D4.6 would contribute to dissemination task T6.1 of WP 6 (Impact Creation) by publishing the work related to RIC in international conferences, workshops, and journals.

The apparent connection of task T4.2 with the tasks of other WPs can be seen in **Figure 1-2**.

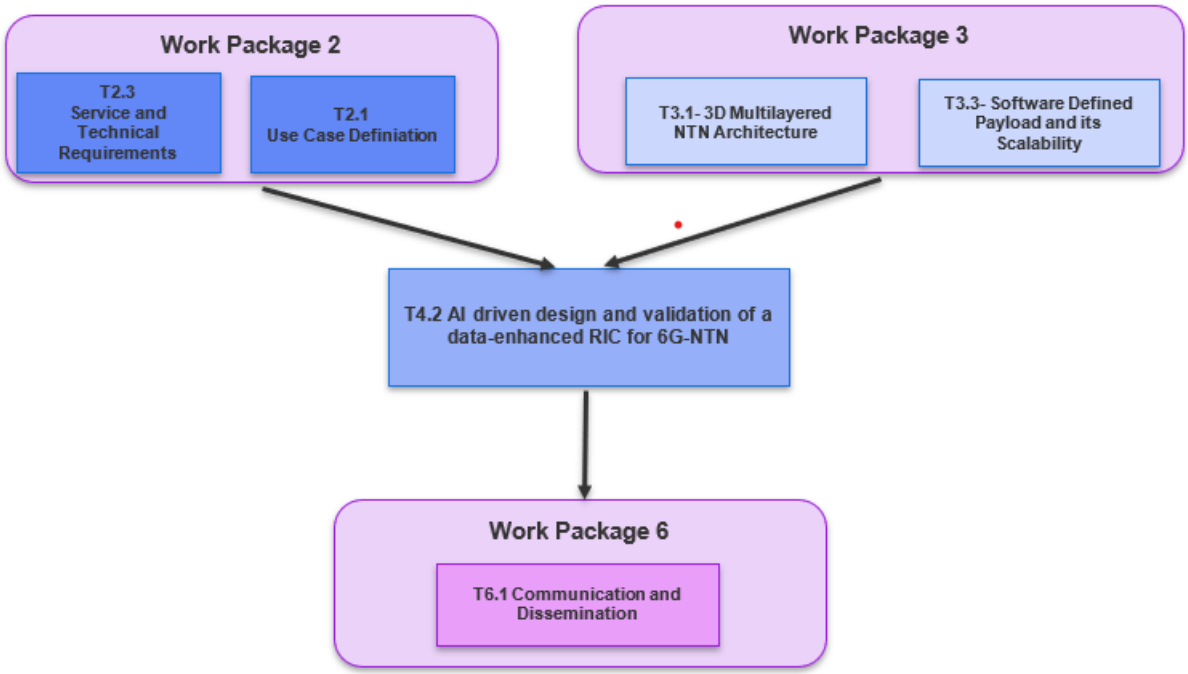

FIGURE 1-2 THE ASSOCIATION OF TASK T4.2 WITH THE OTHER TASKS IN 6G-NTN PROJECT

1.5 ORGANIZATION OF THE DOCUMENT

The organization of this document is as follows:

Section 2 provides an in-depth exploration of the architectural aspects of data-enhanced radio resource control (RRC), considering the architectures defined in WP3 as the foundation baseline. It is important to note that 3GPP AI/ML activities and the qualitative aspects of deploying AI/ML within "Conventional Architecture" (term used in D3.5 [3]) were previously analyzed in D4.2 [5]. However, the "Distributed Architecture" was only addressed at a very high level, leaving several key design and AI/ML requirements unexplored. To address this gap, at first, this document covers the global high-level qualitative AI/ML aspects of "Distributed Architecture" followed by providing the low-level architecture details relevant to the distributed split processing, open interfaces requirements and the deployment of AI/ML agent within the RIC component considering the O-RAN and AI-RAN as baseline.

Section 3 presents a detailed, network function-oriented analysis of intelligent resource management. For each network function, for instance, Traffic Off-loading, Fractional Frequency Reuse, Beam Hopping in NTN, and Link Quality Prediction, the corresponding system model is defined, capturing user dynamics, utilizing the satellite antenna payload from WP3. The associated resource allocation optimization problem is then formulated with clear objectives and system constraints. Based on the complexity and dynamic nature of each problem, AI/ML-based strategies are proposed. In particular, this section thoroughly provides the optimization of each network function deployed in RIC component, highlighting how AI/ML enables adaptive, scalable and effective resource optimization in NTN or NTN-TN environment.

It is important to note that the extensive state-of-the-art analysis, relevant to each network function, was provided in D4.2. Therefore, this report mainly focuses on the analysis and optimization results.

Section 4 draws the conclusion of this deliverable.

2. 6G-NTN ARCHITECTURE FOR DATA-ENHANCED RADIO CONTROL

This section is explicitly dedicated to AI/ML 6G-NTN architecture aspects that support the deployment of various AI/ML based resource optimization network functions. Since each resource network function operates under different objectives, time scales, data dependencies, therefore a unified one-size-fits-all deployment framework is not feasible. Instead, architecture must provide a flexible deployment environment capable of adapting to local requirements and computational demands of each function. In this regard, these adaptive AI/ML deployment functions were built on two possible architectures, within the scope of 6G-NTN, defined in D3.5 [3]: 1) Conventional Architecture 2) Distributed Architecture.

The high-level AI/ML deployment opportunities on Conventional Architecture for each network function are defined in D4.2 [5]. Therefore, this section, at first, provides the high-level AI/ML deployment aspects on Distributed Architecture. Following this, each network function provides the lower-level qualitative analysis of deploying the AI/ML functions for model training, inference and management on the architecture leveraging the functional splits and open interfaces provided by 3GPP, Open-RAN & AI-RAN Alliance [6],[7],[8]. Although the overall architectural framework is employed from D3.5 [3], the hosting and placement of AI/ML functions vary across network functions. This variation is to ensure that each function can be deployed in a more suitable manner for its specific resource optimization objective, as single deployment configuration is not appropriate to address all types of resource optimization problems.

2.1 DISTRIBUTIVE ARCHITECTURE

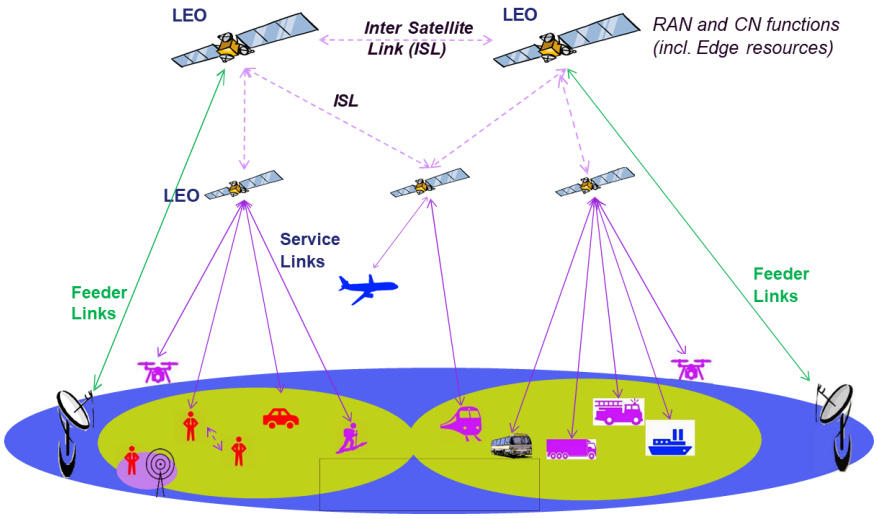

FIGURE 2-1 DISTRIBUTED ARCHITECTURE CITED FROM DELIVERABLE D3.6

Task 3.1 proposed distributed architecture for 6G NTN, wherein the constellation is composed of two interconnected satellite layers as detailed in D3.6 [9]:

- A service satellite layer – A service satellite (SS) carries radio unit (RU) with lower physical layer (PHY) functions.
- A feeder satellite layer – A feeder satellite (FS) carries the rest of RAN (and core network (CN) if needed) functions.

Accordingly, a service satellite is devoted to UE-connectivity and does not have a feeder-link. On the other hand, a feeder satellite contains a feeder link to the ground gateway, but not a service link to any UE. Multiple service satellites connect to a single FS, and nearby feeder satellites are inter-connected as shown in **Figure 2-1**. This way, link-capacity on service and feeder sides are expected to be de-coupled to a certain degree and, thus, it helps to achieve:

- Higher service link throughput by devoting all service satellite power to the service link.
- Better scalability and flexibility for deploying service satellites gradually.

Global Qualitative Analysis

However, using the service satellite carrying gNB RU with lower PHY for supporting an AI/ML function will face limitations:

- First of all, the service satellite in the distributed architecture was initially proposed for benefiting communication, and its hardware and resources will likely be designed and dedicated to communication tasks, which makes it less suitable for AI/ML purpose.

- Second, the RU with lower PHY layer onboard the service satellite will focus on providing user-plane (UP) functions in the lower PHY, and most of the control plane functions will reside in the feeder satellite. Therefore, most of the information collected by the network (NW) from the UE will not be visible by the service satellite, even though the service satellite must relay that information from the UE to the NW. In this case, the service satellite will have no visibility to the collected data from the UE, unless it obtains the data from the feeder satellite and/or the ground NW.

- Third, the onboard RU with lower PHY just provides a bottom layer in the overall end-to-end (E2E) protocol layers between a UE and a gNB/CN. Therefore, it does not have an E2E communication with the UE, e.g. for the purpose of sending a life cycle management (LCM) message, transferring a model to the UE.

In terms of the detailed AI/ML function blocks for AI/ML-based air interface design, the distributed architecture may face certain limitations:

- UE data (training, inference, monitoring) can only be collected from the layers/entities where the data is generated and/or reported to. The layers/entities used for collecting the legacy UE reports and the multi access edge computing (MEC) AI/ML server proposed in D4.2 can only be located in the feeder satellite and, thus, it is natural to use the feeder satellite as the data collection entity. E.g.
  - Layer-1 (L1) measurements/reports take place at higher PHY layer in the feeder satellite
  - Layer-2 (L2) measurements/reports take place at MAC layer in the feeder satellite

- Layer-3 (L3) measurements/reports take place at RRC layer in the feeder satellite
- Over the top (OTT) UP data collection method, as proposed in D4.2, takes place between the UE and CN/DN entity, which is located in the feeder satellite

- Model transfer has to be originated from the CN or gNB centralized unit (CU), depending on the detailed solution for transferring the AI/ML model, e.g. via UP from the OTT server, via RRC signaling/UP from gNB.
  - No matter which solution is used to transfer the model to the UE/NW, model storage is preferred to be deployed collocated/closed to the CN or gNB CU, e.g. in the feeder satellite.
- Model training can be resource-consuming, which is a limiting factor for satellite service. Therefore, it is preferred to train a model in the feeder satellite instead of the service satellite.
- Model inference may be performed at a satellite feeder and/or a UE, where:
  - The inference input data is available; and/or
  - The corresponding action is taken by using the generated inference output.

  Since the service satellite carrying onboard RU with lower PHY has limited functions, e.g. radio frequency (RF), beamforming, and inverse fast fourier transform (IFFT), only limited set of radio functions may benefit from running the inference at the service satellite.
- For performance monitoring and management
  - It is better to be collocated with the data collection entity and/or the model inference entity for obtaining real time input, especially if real-time monitoring is required for ensuring lower management latency
  - In addition, the performance monitoring and management entity is better to be collocated with the CN or gNB CU, to ease the transmission of management decisions to UE(s).

2.2 NETWORK FUNCTION-SPECIFIC QUALITATIVE ASPECTS OF AI/ML 6G-NTN ARCHITECTURE

This section provides the deployment aspects of AI/ML functionalities in disaggregated and distributed RAN architecture.

Disaggregated RAN:

With the emergence of new applications and highly dynamic service requirements to be supported by 5G and beyond, RAN architectures and protocols evolving from traditional highly centralized and vendor specific infrastructure to open, disaggregated, intelligent and virtualized ecosystems. This transformation enables flexible resource allocation to meet the dynamic and growing demands of diverse use cases. In this context, 3GPP introduced eight functional splits in the RAN, enabling an adaptable architecture that separates the deployment of gNBs functions between CUs and Distributed Units (DUs) [6]. This separation allows the flexible deployment and management of different network functions of RAN to optimize performance and resource allocation. CU(s) and DU(s) are logical nodes that include the subset of gNB functions, depending on the functional split option as shown in **Figure 2-2**. These splits range from high layer split (HLS) to lower-layer split (LLS) adoptable based on application

requirements such as latency, throughput, fronthaul bandwidth etc. In the LLS, the division between the CU and DU occurs at the lower layers of the protocol stack, for instance at the PHY (Physical Layer) level, is suitable for latency sensitive applications, where the DU can process time-sensitive tasks locally and closer to RU such as Option 7. Option 7 further splits PHY into 7.1, 7.2 and 7.3, which offloads some of the functions into DU to further ease the burden on midhaul bandwidth. To be specific, in Option 7.1, (i)FFT, Cyclic Prefix (CP) removal/additional functions reside in the DU, the rest of PHY functions reside in the CU. In Option 7.2, iFFT, CP removal/addition functions, resource mapping and precoding functions reside in the DU, the rest of PHY functions reside in the CU. In Option 7.3, only the encoder resides in the CU, and the rest of PHY functions reside in the DU. Conversely, in the HLS, the division occurs at the higher layers of the protocol stack, for instance at the RLC (Radio Link Control) layer such as Option 2 and reduces the fronthaul bandwidth requirements.

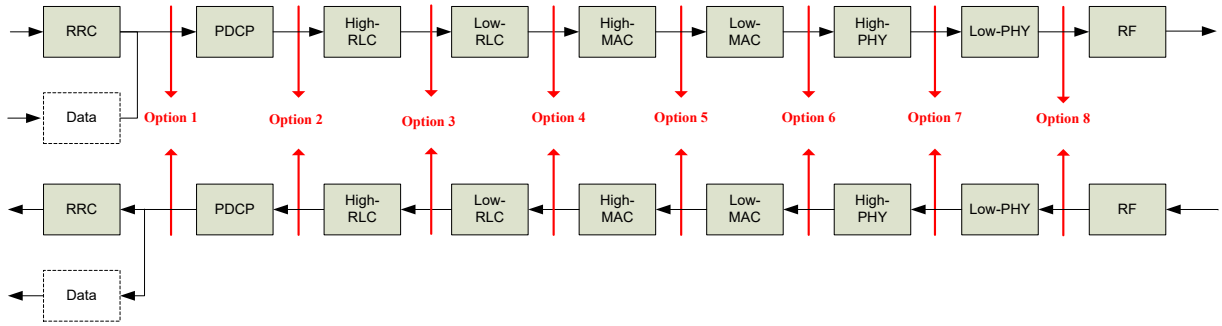

FIGURE 2-2 EIGHT RAN FUNCTIONAL SPLIT OPTIONS INTRODUCED BY 3GPP [6]

To facilitate communication between the CU and DU, 3GPP introduced the F1 interface. This interface is a standardized, open interface that terminates between CUs and DUs that ensures interoperability as in **Figure 2-3** [10]. To further separate the control and user plane, this F1 interface is divided into two components i.e. F1-C (responsible for control signaling and management between CU and DU), and F1- U handles the user data traffic between CU and DU. The other interfaces shown are NG interface terminates between NG-RAN and 5G core and Xn-C interface enables the communications between gNBs CUs.

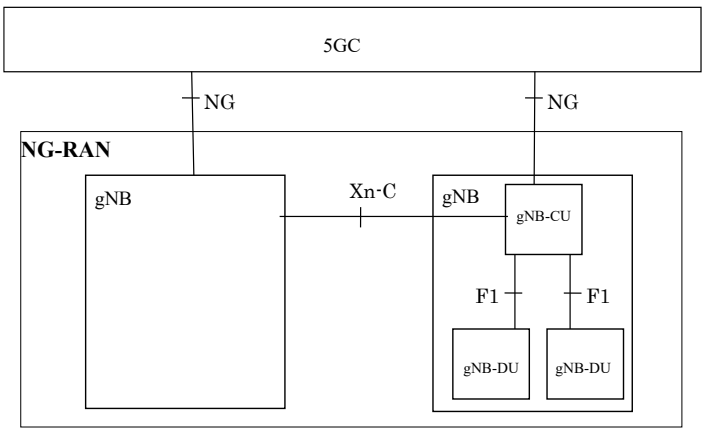

FIGURE 2-3 3GPP INTERFACES [10]

Following the 3GPP RAN standards, the ORAN is a disaggregated approach built on the 3GPP functional split motivated towards openness and interoperability by adopting the existing functional splits, some new interfaces and more specifically RICs that enable intelligence in optimizing the RRM. However, O-RAN adopts two functional splits among the ones that are defined above, such as split Option 2, which is typically referred to as HLS and split Option 7.2x, a LLS split within a PHY layer. Similarly, these splits define how RAN responsibilities are distributed across different O-RAN elements that include O-CU (O-RAN Central Unit), O-DU (O-RAN Distributed Unit) and O-RU (O-RAN Radio Unit).
Besides these processing units, a separate entity introduced in O-RAN is the RAN RIC mentioned above, which supports the intelligence to better optimize the RRM functions. RIC components are further split into non-RT RIC and Near-RT RIC that serves as software platform allowing the hosted applications (rApp, xApp) to control the RAN. The non-RT RIC operates on a time scale longer than 1 second, focusing on non-real time RRM utilizing rApps, higher layer procedure optimization, and policy optimization in RAN. Most importantly, it supports the integration of AI/ML framework into RAN components, offering policy-based guidance for applications in Near-RT RIC and providing Enrichment Information (EI) to enhance Near-RT RIC applications. In contrast, near- RT RIC is another component of RIC that also supports the control and optimization of RRM utilizing xApps but operates within a timescale of more than 10 ms and less than 1 second. This integration of AI into RAN aims to enhance the RAN performance, supports the deployment of AI-based application (xAPP, rAPP) and drives the zero-touch network and service management (ZSM) for adaptive decision making, intelligent radio resource control and automated network configuration.

The latter part of the section integrates the disaggregated RAN components, interfaces and the deployment of AI/ML functions into 6G-NTN defined architecture based on each network function requirements and also defines the specific architecture requirements.

2.2.1 Traffic Offloading

In modern mobile cellular systems, a dense network of cells is deployed to provide connectivity to a demanding number of users. However, the mobility of UE leads to a disparity in the load across the network cells, affecting the QoS for UEs and resulting in inefficient resource utilization. Additionally, the demand for high data rates by UEs and the non-uniform distribution of UEs exacerbate resource overutilization in certain cells. To address these challenges, it is necessary to distribute the workload evenly across cells, thereby ensuring efficient resource utilization. To this aim, different cell-load balancing techniques have been developed in terrestrial networks, aiming to equally distribute the load among cells to maintain a satisfactory end-user experience and efficient network operation. The detailed relevant literature review was provided in D4.2 [5].

In some scenarios, UEs may be unable to transition to neighboring cells due to resource scarcity and limited coverage. This constraint impedes efficient load balancing among cells and diminishes the quality of service for users. Leveraging NTN, the load can be balanced not only between TN cells, but also between TN and NTN, adding a second plane of load balance optimization. In this context, the quality of service of TN can be enhanced by off-loading some traffic to the NTN, increasing the overall network throughput and resource exploitation. The

optimal assignment of UEs to TN or NTN is a demanding task that must be performed in a centralized entity based on the network status information collected from UEs and cells.

The reference architecture to better support the traffic off-loading network function is shown in **Figure 2-4**, which is described below.

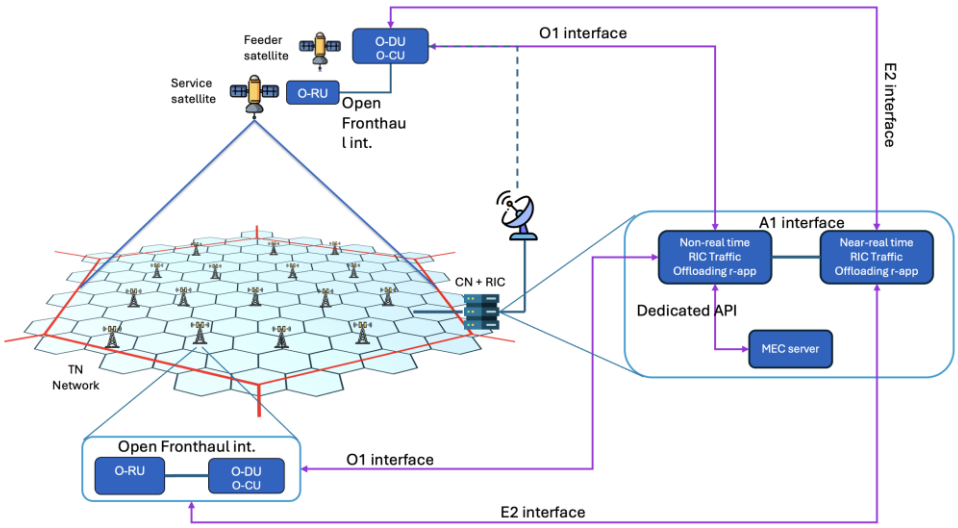

FIGURE 2-4 REFERENCE OPEN ARCHITECTURE FOR THE TRAFFIC-OFFLOADING NETWORK FUNCTION

As it is visible, the TN is composed of a set of three-sector cells, while the NTN is composed of a satellite generating a single NTN beam with overlapped service area with TN and complementing its service**.** TN and NTN are served by the same core network and by the same AI/ML server. The data that must be collected by the AI/ML server to perform model training and inference is listed in **Table 2-1**.

TABLE 2-1 INPUT DATA TO PERFORM MODEL TRAINING AND INFERENCE IN TRAFFIC OFF-LOADING

Collected data	Source	Distribution network	Usage
UEs RSRP measurements	UE	Ground network	Input to training input to management
UE's location	UE	Ground network	
UE's service requirements (included the requested resources)	UE	Ground network	
TN cells resource allocation and load status	TN cell	Ground network	

NTN cell resource allocation and load status	Satellite	Feeder link	
Satellite ephemeris	Satellite	Feeder link	

Due to the coordinated nature of this network function, training, storage, and inference of the AI/ML service should reside in centralized AI/ML servers that can be either located in the same or in different nodes of the network. From **Table 2-1**, it is clear that the majority of the data that must reach the AI/ML server is produced on-ground. Thus, to minimize the average latency of the delivery of the data in the system and the load on the transport network the AI/ML servers should be located on-ground.
More precisely, the traffic off-loading network function should manage the AI/ML functions in the following ways and the overall flow is illustrated in **Figure 2-5**.

Model Storage

For traffic off-loading network function, the model will be stored in the MEC servers located on-ground and near to the location for which the model has been trained and in which it will be exploited. A network trained for global applicability could be stored also in a geographically centralized MEC server.

Model Training

The model training and management is performed on the MEC AI/ML server and non-RT-RIC accessed through non-RT RIC using dedicated API.

Model Inference

The model inference will be performed in the near Real-time RIC. Indeed, the majority of the input data is produced in the ground network and the majority of RAN nodes using the inference output are on-ground. Moreover, the inference of a large AI/ML model is a computationally intensive task that is not easily deployable on a resource-limited satellite.

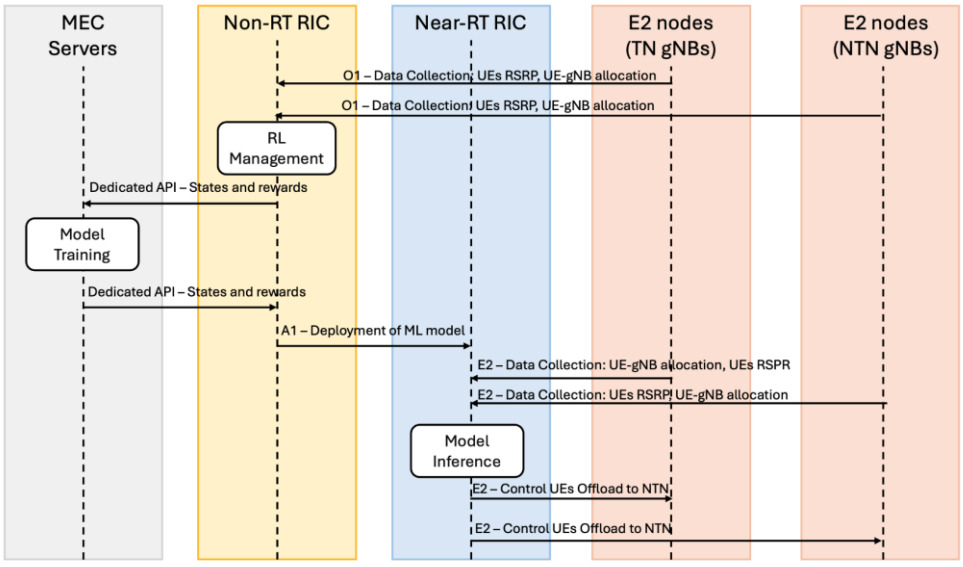

FIGURE 2-5 DATA FLOWS FOR THE PROPOSED ARCHITECTURE

The data shown in **Figure 2-5** is explicitly being considered to support AI/ML model training and inference for cell-load balancing network function and will be collected from the gNB and sent to the non-RT RIC for the training and RT-RIC for the inference via O1 and E2 interfaces respectively. In particular, the gNBs share information about the UEs reference signal received power (RSRP), the UE-gNB allocation and the gNB load status with the non-RT RIC for the management of the neural network (NN), which is then trained in the MEC server. Through a dedicated API, the states and the reward are updated. Once the DQN has been trained, the model is deployed in the near-RT RIC. The latter receives as input the data from the gNBs and outputs the users to be offloaded.
Table 2-2 shows the requirements for each interface in the system. In particular, the major requirements is on the O1 and E2 interfaces on the feeder link due to the propagation delay.

TABLE 2-2 TRAFFIC-OFFLOADING NETWORK FUNCTION ARCHITECTURE REQUIREMENTS

Network Function	Interface	Hosting entity	Requirements
Traffic Offloading	Dedicated API	Server on ground	Support the delay for the API to access the dataset stored and model training
	O1 Interface	Link between the on-ground gNB and the server Link between the feeder satellite and the server (feeder link)	Support the delay on the feeder link to make non-real time decision
	A1 Interface	Server on-ground	Provide policy guidance to the RAN to steer its operation
	E2 Interface	Link between the on-ground gNB and the server Link between the feeder satellite and the server (feeder link)	Support the delay on the feeder link to make near-real time decision

2.2.2 Fractional Frequency Reuse

The ever-growing demand for communication services comes with the immense challenge of utilizing the limited spectrum resources efficiently in both terrestrial and non-terrestrial networks.

One major challenge in satellite communication is inter-beam interference, which appears when a certain frequency bin is reused among the coverage area. Taking this constraint into account, it is very likely that the users end up using the same frequency bands in adjacent beams which results in inter-beam interference. In essence, this results in difficulties to meet the quality of service (QoS) requirements. Firstly, from the perspective of TNs, Frequency Reuse (FR) schemes have been widely adopted where different adjacent cells utilize different frequency resources to mitigate the inter-cell interference. These have been found as viable

solutions to enhance the overall system performance. In the context of FR-n, where n = {1,2,3…N} is the frequency color scheme in FR, for instance, when n=3, each of the three adjacent cells utilizes frequency bands orthogonal to the frequency bands of each other. Although this can alleviate the intercell interference, it comes with the cost of extreme spectral inefficiency since the available bandwidth is not being fully utilized and thereby impacts the system throughput. Considering that problem, FR can be further categorized as Fractional Frequency Reuse (FFR) where further different orders of FR schemes are used within each main cell. This divides each cell into sub-regions with different FR orders, to balance the interference reduction and efficient spectral utilization. For instance, in **Figure 2-6**, FFR-1 is used in the central region of cells (using same frequency resources) while FFR-3 is used at the cell edge (using distinct frequency resources).

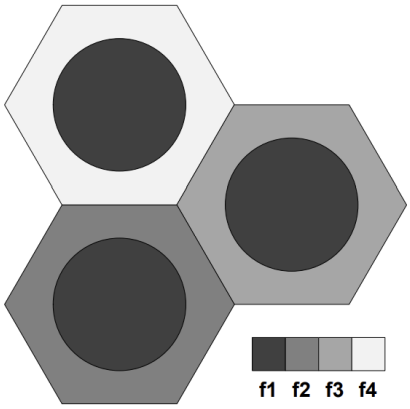

FIGURE 2-6 FFR WITH FR-1 IN CELL CENTRAL REGION AND FR-3 IN CELL EDGE REGION [11]

However, while FFR has been envisaged for wide adoption in traditional TN to address such challenges, it has not yet been extensively explored in NTN due to constraints posed by legacy satellite payloads constructed by array-fed reflectors. In the forthcoming broadband satellite missions, and thanks to the 6G-NTN antenna, direct radiating arrays (DRA) will enable flexible operation of the beams, allowing irregular deployments and mimicking the FFR schemes. Hence, at the satellite system level, the evolution of multi-beam satellite systems makes them compatible with FFR techniques. However, traditional FFR schemes are predominantly static or fixed and unable to cope with the temporal and spatial variability characteristics of NTN such as satellite movement, fluctuating traffic mobility over time etc. This limited adaptability highlights the necessity of intelligent and flexible FFR mechanism capable to reacting to real-time network conditions. This importance of this capability increases in 6G networks, which are intended to accommodate dynamic service requirements across various use cases. This motivates the integration of AI to dynamically tune the FFR for effective resource optimization. In particular, as previously discussed, advanced beamforming strategies enable the creation of concentric beams, allowing the inner and outer beam radii to be adjusted according to user demand instead of having the same all the time. Furthermore, the joint optimization of beamwidth and power allocation between inner and outer regions may introduce an additional degree of freedom, facilitating throughput optimization in response to the time-varying nature of user traffic.

Architecture Aspects

In D4.2, FFR framework provided a high-level description of AI deployment, answering the concerns relevant to where the model training, inference and data collection etc. should take place. Therefore, this subsection extends the analysis to low-level and designs the distributed agentic framework that effectively optimizes the FFR framework in terms of architectural aspects. In this perspective, following this O-RAN paradigm, it is imperative that FFR network

functions can be deployed as xAPP and rAPP for different stages of optimization depending on the AI management task. Since the FFR framework can be exploited by manipulating the antenna and beamforming capabilities residing in RU, thereby, it is envisaged that the FFR AI framework will be deployed at the edge in space, as detailed in the architecture given in **Figure 2-7**.

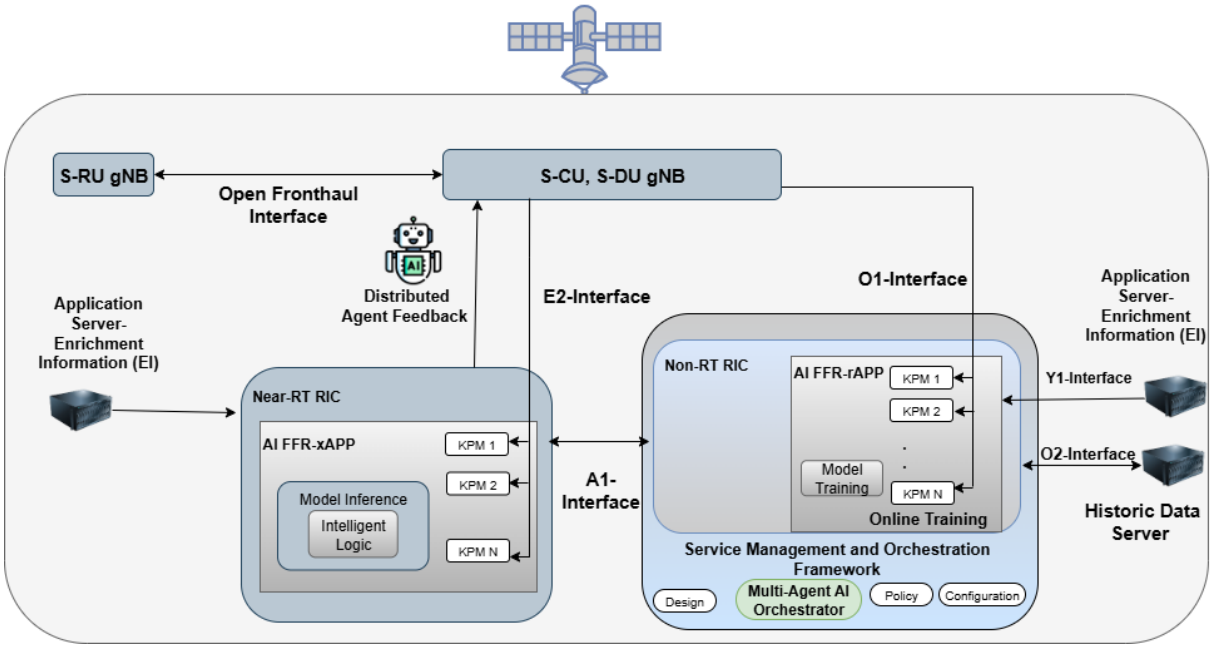

FIGURE 2-7 EDGE DEPLOYED FFR AI IN THE CONVENTIONAL ARCHITECTURE

The provided architecture configuration is relevant for scenarios, when the satellite moves to a remote area or when connectivity is required in the middle of an ocean or during a natural disaster when on-ground infrastructure is severely compromised. Therefore, it is important to emphasize that the architecture follows the configuration of regenerative satellite payload with at least a full gNB and both near-RT and non-RT RICs on-board. The provided architecture follows the space-based O-RAN split as S-RU, S-CU and S-DU where S-RU connects to S-DU using Open fronthaul interface. S-CU and S-DU, potentially considered as E2 nodes, communicate with each other by using the F1 interface while the communication between E2 nodes and near RT RIC enables using E2 interface and hence E2 nodes functionality can be controlled through deploying the external xAPP and rAPP applications. Moreover, the A1 interface terminates between near RT RIC and non-RT RIC and is used to transfer the trained ML models and policies. In addition, unlike E2 interface between near-RT RIC and E2 nodes, the O1 interface potentially provides communication between E2 nodes and non-Real time RIC directly, typically used to provide the key performance measurements (KPMs) data for model training. The ``Intelligent Logic`` component is responsible for making decisions of allocating the power and the inner beamwidth given the network state at time *t*. Furthermore, Y1 interface can be utilized to gather information from external application servers. The architecture supports both online training in case the conditions are dynamic, and the offline training by using the historical dataset stored in the database of server. Optionally, the MEC server or O-cloud can be further deployed to host intensive model training and can be assessed through rAPP by using O2 interface. The flow of information in the proposed architecture in optimizing the FFR network function can be seen in **Figure 2-8**.

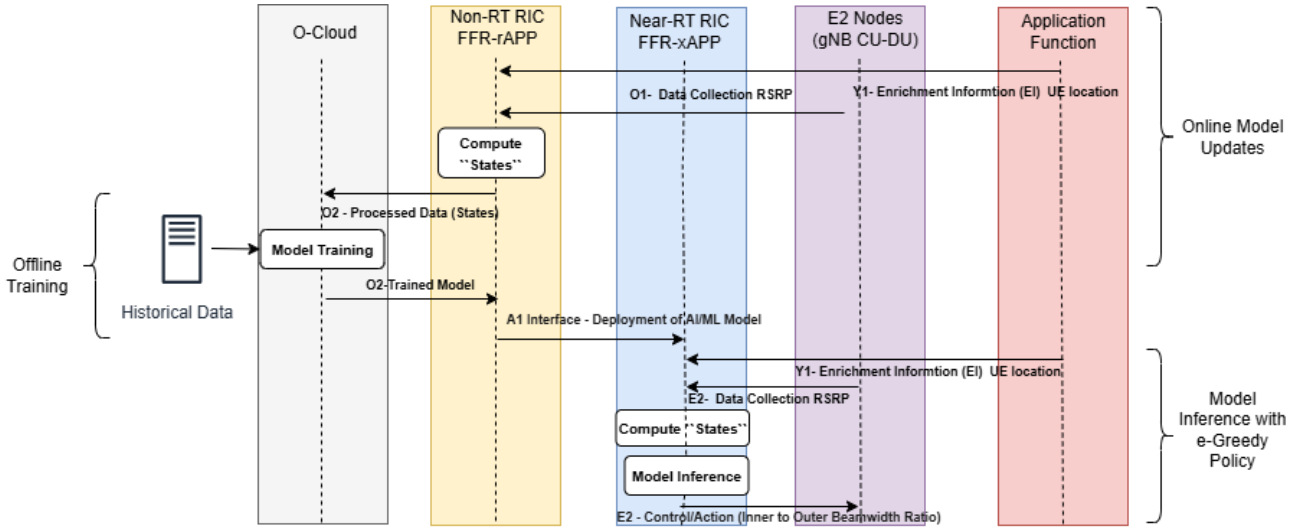

FIGURE 2-8 THE INFORMATION FLOW LEVERAGING THE FFR-AI FRAMEWORK IN THE PROPOSED ARCHITECTURE.

After carefully designing edge-deployed FFR AI, this network function would manage the AI/ML functions in the following ways:

Model Training:

- The model training for FFR AI can be possibly performed on O-cloud/MEC or in non-RT RIC depending on the computational requirements. Model training is associated with FFR-rAPP due to the fact that model training is a computationally intensive process and operates with a time scale of more than 1 sec. Also, the model may require historical data information during the training.
- To perform the model training, the data or KPMs required are collected from E2 nodes (S-CU, S-DU) using O1 interface for online training and historical data from a dedicated server and send them to rAPP in non-RT RIC where a part of FFR application is deployed. This application primarily observes State Space and sends it to AI/ML O-cloud MEC server through O2 for model training.

Model Storage:

For the FFR network function, the model is stored on-board and supports the transfer of trained model to near-RT RIC for model inference through A1 interface.

Model Inference:

- Once the model is trained and stored, this is collected by FFR-rAPP and subsequently by FFR-xAPP deployed in near-RT RIC through an A1 interface for model inference. This software platform allows the hosted applications to control the RAN. Therefore, the model inference would utilize the information collected and processed in relevant elements through dedicated interfaces as depicted in **Figure 2-8** and decide one of the greedy actions using the intelligent logic and apply that action on E2 nodes to dynamically adjust the FFR resources.

Distributed Architecture:

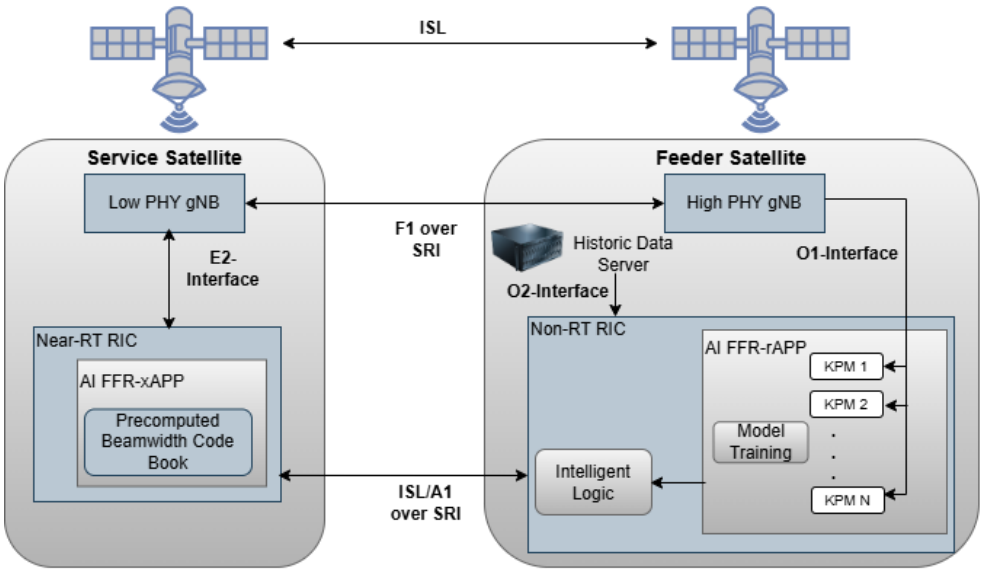

FIGURE 2-9 DISTRIBUTED OPEN ARCHITECTURE WITH TRAINING CAPABILITIES ON FEEDER SATELLITE

The distributed architecture comprises 4 service satellites and 1 feeder satellite. The service satellite hosts the lower layer split while the core network and edge computing capabilities are hosted by feeder satellite. To perform real time actions regarding beamforming, the dedicated FFR-xAPP within near-RT RIC is hosted by service satellites that could communicate through standardized E2 interface. As the edge computing capabilities are in feeder satellite, the FFR-rApp in non-Real Time RIC is integrated into feeder satellite as shown in **Figure 2-9**.

Model Training:

Similar to conventional architecture, model training will be performed in FFR-rAPP in non-Real Time RIC but on feeder satellite. The data for model training is collected from satellite service through F1 over SRI.

Model Inference:

After training the model on feeder satellite, the model can be transferred to service satellite for inference, or the intelligent logic can transfer only the decision to FFR-xAPP through ISL where beamforming codebook is precomputed corresponds to specified beamwidths.

Data Collection:

The data listed in **Table 2-3** is used to support AI/ML model training and inference for the FFR network function to dynamically adjust the inner beamwidth and adaptive power allocation.

TABLE 2-3 INPUT DATA TO PERFORM MODEL TRAINING AND INFERENCE IN FFR NETWORK FUNCTION

Collected and Calculated Metrics	Source	Usage
UE Geocoordinates	Historical Data Server	Input to training and inference model onboard
UE distance to beam Center	NW	Input to training and inference model onboard
UE traffic density in each beam	NW	Input to training and inference model onboard
Dynamic Inner Beamwidth	NW	Output from inference
Dynamic Power Allocation	NW	Output from inference
UE Capacity	UE	To evaluate the system performance

Architecture Requirements

Table 2-4 shows the architecture requirements for FFR with respect to the distributed architecture

TABLE 2-4 DISTRIBUTED ARCHITECTURE REQUIREMENTS FOR FFR NETWORK FUNCTION

Network Function	Interface	Hosting Entity	Requirements
Fractional Beamforming	Model-rApp and KPMs (Internal API)	In feeder Satellite (Gbs)	Synchronization with API to access the dataset stored and model training
	High PHY and rApp (O1)	In feeder Satellite	Synchronization with O1 to access the KPMs for online updates.
	Intelligent Logic-rApp and High PHY (O1)	In feeder Satellite	Synchronization with O1 to make decisions on real time KPMs
	rApp and xApp (A1)	In feeder satellite and service satellite	Synchronization with F1.
	xAPP and Low PHY (E2)	In service satellite	Synchronization with F1

Power budget envelope:

For the SS in the distributed architecture, D3.6 [9] reports total ~8.01–8.18 kilowatt (kW) power consumption and ~229 kilogram (kg) mass, with RF FE & beamforming ≈ 7.8 kW (≈95–97% of total), RU 60–120 W (~1%), and two 20 mm OISLs ≈ 75–130 W each (≈2–3%). For the FS, totals are ~1.97–4.37 kW (8 OISLs) or ~2.37–5.17 kW (12 OISLs) and ~228–388 kg, with Base-station (DU/CU without RU) 0.8–2.4 kW, RF FE ~370 W, and 80 mm OISLs 100–200 W each (8–12 terminals depending on topology). These values cement the rule that continuous

compute (training and, by default, inference) belongs to the FS, while the SS remains RF-dominated.
Indeed, training (re-training, fine-tuning, continual learning) is not recommended on the SS because the SS power is already committed to RF + OISLs + RU; the C-band SS table shows only tens of watts—at best—for any additional continuous compute, and that margin can vanish at high OISL load. Conversely, the FS is explicitly dimensioned with 0.8–2.4 kW for DU/CU and a multi-kW total budget, which comfortably accommodates graphic processing unit (GPU)/system on chip (SoC) training windows (mixed precision, scheduled outside peak forwarding). In practice, training power on the FS sits within the DU/CU envelope, i.e., hundreds of watts depending on epochs/batch and concurrency of xApps.

However, there are regimes where placing a light-weight inference policy on the SS is not merely admissible but advantageous. The case rests on four pillars—control-loop stability, autonomy under backhaul impairment, beam agility, and data-movement efficiency—bounded, always, by the SS's power and mass budgets.

When the control period of the network function such as the one we are considering is sub-second and the FS path introduces delay or jitter comparable to that budget, local inference suppresses ISL/fronthaul variability and permits more assertive control gains. Under ISL congestion, irregular ground contact—a minimal edge policy on the SS provides fail-operational continuity and graceful degradation, avoiding oscillations from tardy centralized decisions. With high beam-reconfiguration cadence, where the set of active beams changes swiftly and coherence windows are short, a compact SS-resident policy plus per-beam caches avoids control chatter and reduces ISL pressure. Finally, when the function's features already reside on the SS (RSRP/ signal to interference and noise ratio (SINR)/physical resource block (PRB), Low-PHY buffers), it is more efficient to decide locally and uplink (UL) only succinct actions/telemetry than to export feature vectors each tick—an operation that also adds queuing and jitter.

This preference is conditional on power and mass viability. Let

$$P_{\text{margin}}(N_b) = P_{\text{SS,tot}} - P_{\text{RU}} - P_{\text{OISL}} - P_{\text{RF}}(N_b) \qquad (2.2.1)$$

be the continuous SS margin available for non-RF compute at an active-beam load N_b. SS-side inference is enabled only if

$$P_{\text{margin}}(N_b) \;\geq\; (1+\beta)\, P_{\text{inf}}, \qquad (2.2.2)$$

with βa 10–20% thermal/reliability guard and P_{inf} the continuous power of the local policy.

Practically, this calls for compressed models in the 20–30 W band mapped onto existing FPGA/DSP/SoC; adding a new accelerator typically breaks mass neutrality and complicates thermal integration. As $N_b$approaches the payload's beam ceiling, $P_{\text{RF}}(N_b)$ rises and the margin collapses; in that regime, inference should remain on the Feeder Satellite (FS), which retains electrical/thermal headroom and centralizes model lifecycle.

Hence a disciplined placement policy: by default, run inference on the FS and apply on the SS; “switch on” SS-side inference when timing demands it, backhaul is uncertain, beam agility rewards it, and the margin test is passed without touching the mass line. As traffic or beam count grows—or the backhaul stabilizes—revert to the default to preserve SS budgets while keeping training, validation, deployment, and audit under FS governance.

Beyond power and mass, beam management itself naturally calls for a centralised view at feeder level rather than at the individual service satellite. Each feeder satellite supervises multiple SS and sees a global picture of traffic, interference and handover needs across its footprint and its cluster; many of the key decisions (which beams to light, how to shape them, where to move power) depend on relative load and geometry between beams on different SS, not just on local measurements. Performing inference only at SS level would fragment this view, forcing complex coordination protocols between SS and risking incoherent or oscillatory behaviour when beams overlap or compete for the same users.

Locating the AI-driven beam management in the feeder domain also simplifies inter-feeder coordination. Feeder satellites can exchange aggregate load and interference information over high-capacity ISLs and jointly decide how to share traffic, how to hand over users between clusters, and how to implement redundancy when a feeder is degraded or temporarily unavailable. In this setting, the beam-management logic becomes a distributed but globally aware controller running on a small number of powerful feeder nodes, rather than a collection of loosely coupled policies embedded in dozens of SS. This not only improves the quality of the optimisation (better interference coordination, smoother cross-SS handovers, more robust load-balancing), but also makes upgrades, rollbacks and algorithmic experimentation far easier: updating a handful of feeder-based AI engines is manageable, whereas updating and re-qualifying code on every SS would be costly and slow.

In short, keeping beam management and its AI components on feeder satellites ensures that decisions are taken from a constellation-level perspective, with access to multi-satellite context and redundancy mechanisms, while leaving service satellites focused on the efficient generation of RF power and the faithful application of the commands they receive.

While at the time of writing this deliverable the power budget for the Q/V band case is not available, it is expected that same conclusions can be obtained.

Storage budget envelope:

The FFR model’s lifecycle benefits from a clean split between primary persistence and operational caches. Primary persistence (datasets, replay buffers, checkpoints, and the model registry) should live on the ground by default, where capacity scales more economically, data governance (audit, retention, export control) is simpler, and deployment traceability is stronger. From there, immutable snapshots are synchronized to the constellation at a cadence dictated by drift and update milestones.

When operations demand autonomy—wide contact gaps—a working set belongs on the FS: recent checkpoints, small replay buffers for continual learning, and the minimal registry needed for safe roll-forward/rollback without feeder dependence. This orbital storage should be sized

with energy/thermal sobriety considering that the model and datasets for this network function are estimated to be below 1 TB. The SS, in contrast, should host only tiny, ephemeral caches (per-beam coefficients/tables) to avoid eroding its already tight non-RF budget.

2.2.3 Beam Hopping

We provide in the following table a summary of the essential data needed for training and inference in beam hopping models, including information about the entities from which these data are sourced. This includes measurements collected from the NW/UE.

Required Data

TABLE 2-5 INPUT DATA TO PERFORM MODEL TRAINING AND INFERENCE FOR BEAM HOPPING

Data	Source	Usage
The DL data volume in the buffer	This information can be delivered in real time through the gNB-CU-UP's internal counters or protocol-specific statistics, such as those available via the gNB's management interface	Input for model training and inference
Frequency resource	The available PRB for use (over PDCCH from gNB-CU-CP to UE), for example from Downlink (DL) Channel Information (DCI) message	Input for model training and model inference
SINR at each user in each cell	RRC signaling at gNB-CU-CP. For example from CSI-SINR or SS-SINR.	Input for model training and model inference

Architecture Requirements

The effectiveness of beam hopping is influenced by the placement of Virtual Network Functions (VNFs) within NTN networks, with data collection mechanisms further constraining these placement decisions.

Below are the architectural requirements necessary to achieve optimal performance for the beam hopping solution:

TABLE 2-6 ARCHITECTURE REQUIREMENTS FOR BEAM HOPPING NETWORK FUNCTION

Network function	Hosting Entity	Requirements
Data collection for model inference	gNB-CU-CP gNB-CU-UP	The gNB-CU-CP and gNB-CU-UP should be co-located with the near-RT RIC—where the inference model runs—to enable timely and efficient decision-making for beam hopping

Model Training	non-RT RIC (potentially offline)	There are no specific placement constraints for the non-RT RIC concerning the gNB-CU for data collection
Model inference Execution of beam hopping decisions	Near RT-RIC	Since model inference governs the beam illumination strategy and radio resource management across beams, the resulting decisions should be directed to the gNB-CU, RU, and DU for full execution. Consequently, it is advisable to integrate the gNB-RU and gNB-DU alongside the gNB-CU.

2.2.4 Link Quality Prediction

In NTN, a UE would expect frequent satellite switch due to the movement of satellites. In order to manage that, the UE's radio conditions towards the serving satellite and the other neighbour satellite(s) need to be accounted for determining a proper satellite switch. Thus, instead of performing real measurements by the UE as in the legacy networks, it is proposed for the UE to run an AI/ML model to estimate and predict the NTN radio channels, which can reduce the number of UE measurements as well as to enable early coordination and preparation between the two cells served by the leaving and the upcoming satellites. It is noted, the focus here is to run the AI/ML model at the UE, which can benefit the UE's mobility management in both RRC_IDLE/RRC_INACTIVE and RRC_CONNECTED states.

The gNB RU with lower PHY onboard the satellite provides only a relay layer dedicated to communication with limited functionalities. As mentioned in **Section 2.1,** the AI/ML function design involving gNB and/or AI/ML MEC server can be deployed to the feeder satellite, e.g.

- Data is collected by the feeder satellite from the UE via the service satellite.
- Model training is performed at the AI/ML MEC server at the feeder satellite, by leveraging the data collected by the feeder satellite.
- AI/ML model can be stored at the feeder satellite and transferred to the UE from the feeder satellite via the service satellite.
- Perform model monitoring and LCM tasks at the feeder satellite, and the LCM command generated by the feeder satellite can be transmitted to the UE via the service satellite

➲ Model inference is performed at the UE.

➲ If different service satellites connected to the same feeder satellite have different implementations, e.g. with different RF parameters, different models may be considered for different service satellites, though the different service satellites may connect to the same feeder satellite. Thus, the UE switches the model when switching to the upcoming satellite service, though the feeder satellite is unchanged.

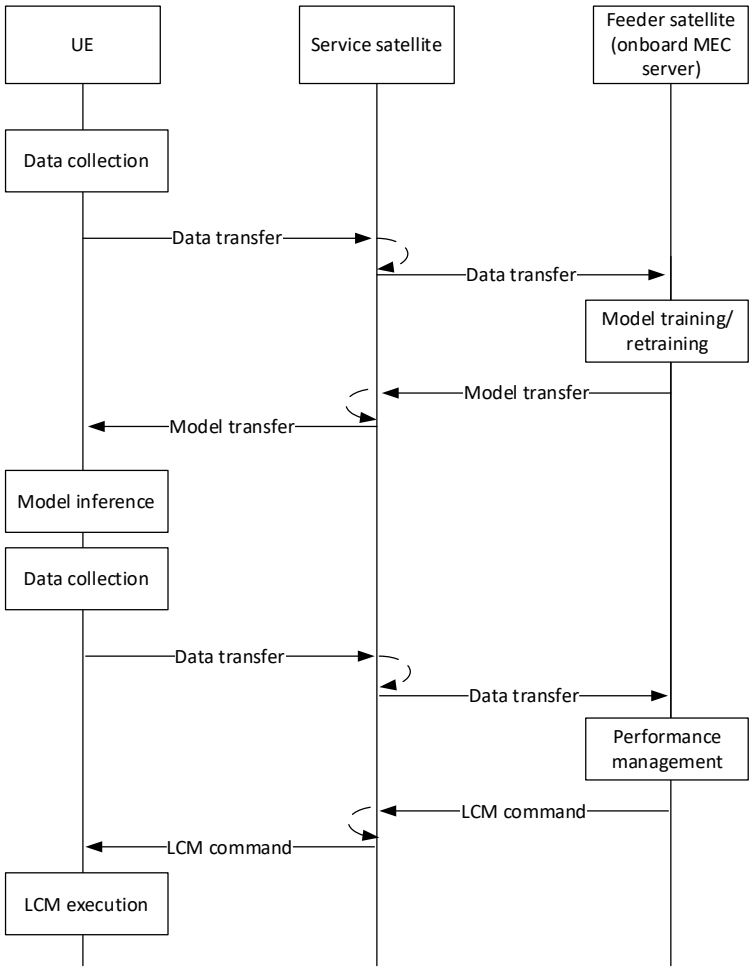

FIGURE 2-10 AN EXAMPLE TO ILLUSTRATE THE INTERACTION BETWEEN THE UE AND THE MEC SERVER VIA USER PLANE SOLUTION IN THE CONSIDERED LINK QUALTIY PREDICTION TECHNOLOGY

Figure 2-10 illustrates an example flow chart for the interaction between the UE and the onboard MEC server regarding the proposed link quality prediction technology. For example, the UE may send its collected data to the MEC server for model training/retraining. Afterwards, the trained/retrained model(s) can be provided by the MEC server to the UE, such that the UE may perform model inference locally. In addition, the data collected by the UE may be sent to the MEC server for performance management. Thus, the MEC server can oversee the performance of the AI/ML function based on the data collected from the UE, such that it may determine if any LCM action needs to be taken. If yes, the MEC server generates the LCM command and sends it to the UE for execution, e.g. to fall back to convention non-AI/ML-based solution or switch to another AI/ML model.

As mentioned in D4.2 [5], all the interaction data/message in **Figure 2-10** will be sent between the UE and the MEC server in the user plane, e.g. via an application server. Therefore, the 3GPP NW only needs to transport the data message between the UE and the MEC server. In other words, the interaction data and message (e.g. the model structure, data structure, and the message definition) do not need to be specified by 3GPP and they can be left to the vendor(s) that provides the AI/ML solution. This solution can largely reduce the specification effort to support AI/ML operation, since it avoids/reduces the need to modify the 3GPP-defined radio interface(s). Thus, it also has the benefit to potentially reduce the time duration for bringing AI/ML solution(s) to practical implementation.

After a model has been trained/retrained, the management function may decide to deliver the model to one or multiple UE(s). Two options may be considered for model delivery:

- **Option 1:** The management function may decide to proactively transmit the model, e.g. by using a broadcast/multicast session from the MEC server to all UEs. In this case, the transferred model is stored at the UE besides at the MEC server. After that, the management function can decide if and when the model should be activated for local inference at the UE, and it may send such an instruction to the UE and/or the base station. In this option, there is a possibility that the model may not be used before it becomes outdated, which brings a low efficiency on UE's internal storage usage, energy efficiency, and traffic load. However, the model activation procedure can be fast, since the model is already available at the UE.
- **Option 2:** The model may be transferred based on the real-time need for activating the model. For example, the UE may first announce its capability for supporting the model. Afterwards, if management function decides a need to activate the model for the considered UE, the management function may then request to transfer the model to the UE. Therefore, in this case, once the model is received and available at the UE, the UE may start to apply the model for inference. It is noted, in this option, after the model has been received by the UE, the UE may have to inform the base station and/or the management entity, such that the base station and/or the management entity has a synchronized understanding regarding when the UE activates the model inference. In addition, the model may or may not be kept at the UE after being deactivated, e.g. if the UE has to switch to another model. Thus, if the deactivated model is kept at the UE and it is not outdated, there is no need for the MEC server to transmit the same model to the UE in future.

It is further noted, **Figure 2-10** shows the interaction between the UE and the AI/ML server deployed onboard the satellite, i.e. the MEC server. However, it can also be used if the AI/ML server is deployed on the ground. As shown in Section 2 of the previous deliverable D4.2 [5], different AI/ML functions, e.g. data collection, model training, model storage, management such as performance monitoring and LCM, can take place in the MEC server, in the on-ground AI/ML server, and by a joint cooperation between the MEC server and the on-ground AI/ML server. **Table 2-7**shows the pros and cons of using MEC server and the on-ground server for the different AI/ML functions, by considering the operation of the AI/ML-based NTN channel prediction described in this subsection.

TABLE 2-7 COMPARISON OF DEPLOYING A CONSIDERED AI/ML FUNCTION BETWEEN AT THE MEC SERVER AND AT THE ON-GROUND SERVER

AI/ML function	MEC AI/ML server	Onground AI/ML server
Data collection	Pros: • Low latency • Reduced load over feeder link Cons: • Need to transfer the local collected data from the leaving satellite to the upcoming satellite	Pros: • Data remains in place, i.e. no need to transfer data between satellites Cons: • High latency • Higher load over feeder link
Model training	Pros:	Pros:

	• Able to use the data generated/collected by the MEC server for local training Cons: • Higher requirement for satellite complexity	• Lower requirement for satellite complexity • Able to obtain a generalized model that is applicable for UEs under multiple satellites Cons: • Training data needs to be available at the on-ground server
Model storage	Cons: • May need to transfer a region-specific model between the leaving and the upcoming satellites	Pros: • Model remains in place, i.e. no need for model transfer between satellites
Management such as performance monitoring and LCM	Pros: • Reduced latency for performance monitoring and management action • Capability to directly interact with and manage the onboard NW components	Cons: • Higher latency for performance monitoring and management action • Need to use feeder link to interact with the onboard NW components for management actions

Architecture Requirements

TABLE 2-8 ARCHITECTURE REQUIREMENTS FOR LINK QUALITY PREDICTION NETWORK FUNCTION

Network Function	Interface	Hosting entity	Requirements
Link Quality Prediction	Over-The-Top via user plane traffic	In feeder satellite (with onboard MEC server)	Enable the UE and the MEC server for AI/ML-related interactions, e.g. to transport the collected data, the trained model, and the LCM command via an application
	Internal API	In UE	Enable a UE-internal application to interact with the UE modem, e.g. to collect the UE data, to deploy a trained AI/ML model, and/or to manage the AI/ML operation.

3. OPTIMIZATION TECHNIQUES

As described in previous section about hosting each network function in the RIC component of Open architecture. Therefore, this section is dedicated to the implementation of AI/ML to optimize each network function itself.

3.1 TRAFFIC OFFLOADING

This subsection provides the system description, the mathematical model, and the simulation results of the traffic offloading function.

3.1.1 System description

We consider a rural area served by a uniform deployment of terrestrial 6G Base Stations (BSs). As recommended in 3GPP TR 38.901, for the considered Rural Macro (RMa) scenario, we assume a hexagonal grid layout of $N_{BS} = 19$ macro sites deployed at an Inter-Site Distance (ISD) of 5000 m. Each macro site has a height h_{BS} of 35 m and three sectors per site. All BSs reuse the same terrestrial frequency band B_{TN}. The same rural area is assumed to be served by a 6G NTN beam on an orthogonal band B_{NTN} of the same width.
The users are uniformly distributed in the reference area and generate uplink traffic and each user in the network is continuously active. The number of UEs in the coverage area is configured to ensure that the requested user traffic exceeds the terrestrial system capacity.
In order to assign the UEs to a gNB an important measurement report is the reference signal received power (RSRP). A network allows the UEs to report the signal quality of the current gNB, i.e., the serving one, and the neighboring gNB.

In addition, proper load measurement of the cell is fundamental for optimizing the performance of a network through load balancing. Thus, as in [12] we use the radio resource usage ratio (RRUR) as a load measurement metric. The RRUR is the ratio of the bandwidth already used in gNB over the total available bandwidth at the same gNB. It is calculated as follows:

$$\beta_n = \frac{1}{\omega_n} \sum_{k=1}^{K} \gamma_k \zeta_k \qquad (3.1.1)$$

Where ω_n is the total bandwidth of the $n-th$ gNB, γ_k and ζ_k represent the number of PRBs and the resource block bandwidth allocated to the user k. The resource block bandwidth depends on the numerology. RRUR values can be between 0 and 1, with 1 meaning that the gNB is saturated.

In this network function, each UE is allocated to the gNB which provides the strongest RSRP. Based on the common load measure, i.e., the RRUR, the load distribution is performed. In particular, a higher RRUR of a gNB indicates that the cell has a higher load to serve and fewer available resources. Thus, new UEs in an overloaded cell will reduce the per UE data rates. Therefore, it is necessary to reduce the load of the overloaded cell by offloading a UE to another gNB.

3.1.2 Problem formulation

In a network, if the RRUR of a gNB is close to 1, a user that moves into the cell will either be dropped or will experience a low data rate. Therefore, a new user in an overloaded cell will reduce the per user data rate, affecting the quality of service of the UEs. To reduce the RRUR of a gNB, load balancing among cells is necessary, from overloaded cells to underloaded neighboring cells (including the gNB at the satellite). In load balancing, the total network load is known by the RIC. Leveraging the broad and centralized view of the RIC, the traffic off-loading network function aims at increasing the system throughput of the integrated 6G network. To this regard, the network function can optimize the system throughput by selecting which UEs to offload from the TN to the NTN. Then the optimization problem could be formulated as:

$$P: T = \max_{x_k} \sum_{k=1}^{N_{UE}} x_k^{TN} \zeta_k \sum_{j=1}^{\gamma_k} log_2\left(1 + \frac{h_k^{TN} P_k}{N + \sum_{i,i\neq k} \xi_{i,j} P_i h_i^{TN}}\right) + x_k^{NTN} \gamma_k \zeta_k log_2\left(1 + \frac{h_k^{NTN} P_k}{N}\right) \quad (3.1.2)$$

$$s.t.:$$

$$C1: x_k^{TN} + x_k^{NTN} \leq 1 \; \forall k = 1, \dots, N_{UE}$$

$$C2: \sum_{k=1}^{N_{UE}} a_{k,g} \gamma_k \leq \Gamma_g \forall g = 1, \dots, N_{BS}$$

$$C3: \xi_{k,j} = \begin{cases} 1, & if\ \gamma_k \geq j \\ 0, & otherwise \end{cases} \forall k = 1, \dots N_{UE}; \forall j = 1, \dots, \max_{\mathrm{g}} \Gamma_g$$

Where T is the throughput, $x_k^{TN}\ and\ x_k^{NTN} \in [0,1]$ indicates the users assigned to a gNB terrestrial and non-terrestrial, respectively, h_k^{NTN} and h_k^{TN} are the channel coefficient between the UE and the NTN gNB and the TN gNB respectively, and P_k is the transmitted power, Γ_g is the number of PRBs associated to a gNB, $\xi_{k,j}$ is a binary variable that is 1 if user k transmits over the j-th PRB, and $a_{k,g}$ is a binary variable that is 1 if user k is served by the g-th gNB., and $\xi_{i,j} \in [0,1]$ is an indicator to check on which PRB there is the interference. Constraints C1 guarantees that a served user is served either by one BS or one satellite, while C2 guarantees the total number of PRB allocated to the served UEs does not exceed the total number of resource available at the gNBs.

3.1.3 Optimization framework

The optimization technique selected to implement this network function is based on Deep Q-Learning (DQL). DQL is an advanced Reinforcement Learning (RL) algorithm that leverages the power of deep neural networks to solve problems with high-dimensional state spaces, such as the traffic off-loading network function that is being evaluated. This approach extends the classic Q-Learning algorithm by using a neural network to approximate the Q-value function, which predicts the expected cumulative reward for taking a given action in a given state and following the optimal policy thereafter.

DQL has been selected for this network function because its learning process involves an agent interacting with an environment and does not require large datasets. The agent perceives the environment's state and takes actions that affect the state, receiving rewards as feedback. The state space $\mathcal{S}$ represents all possible situations the agent can encounter. The action space $\mathcal{A}$ includes all possible actions the agent can take. These actions might be discrete or continuous. The reward function $R(s, a)$ provides feedback from the environment, signaling how good or bad the action taken by the agent was in a specific state. This reward is used to guide the learning process, pushing the agent towards actions that maximize

cumulative rewards. The Q-function $Q(s,a)$ estimates the expected cumulative reward of taking action a (i.e., which UE will be offloaded) in state s and following the optimal policy thereafter. The policy π is the strategy the agent uses to decide its actions based on the current state.

The DQL framework is leveraged in this optimization function to address the high dimensionality of the considered environment. Specifically, the DQL has a state space $\mathcal{S}$ that provides the agent with a snapshot of the network's condition and is composed of several key metrics for each base station, which is composed by: i) the allocation of each UE to the gNB; ii) the RSRP; iii) the requested traffic from each UE; the load status of the gNBs. The action space $\mathcal{A}$ is the set W_A that identifies the allocation of the UEs to the NTN. The reward is computed to reflect the optimization objective of maximizing the system throughput, as the throughput gain obtained by applying the selected action:

$$R(s_n, a) = T_{S_t,a_t} - T_{greedy_t} \quad (3.1.3)$$

where T_{greedy_t} is the throughput obtained through the greedy algorithm, which identifies the best users to be offloaded.

3.1.4 Network Function Simulation

The simulation reflects the system description implementing the optimization framework and defining the scenario with which the optimizer interacts. As described in section 3.1.1, for the considered RMa scenario we assume a hexagonal grid layout of 19 macro sites with three sectors per site. In the complete scenario, the network function would need to optimize the off-loading of all the UEs in the satellite coverage area considering the radio resources they are allocated to. Users ask for different number of resources (in terms of PRB), which can be allocated considering only one OFDM symbol. At each iteration of the simulation, users are dropped in a random position that reflects the UE-gNB allocation as shown in **Figure 3-1**. As mentioned above, the DQL decides the optimal UE to be offloaded based on the state space $\mathcal{S}$.

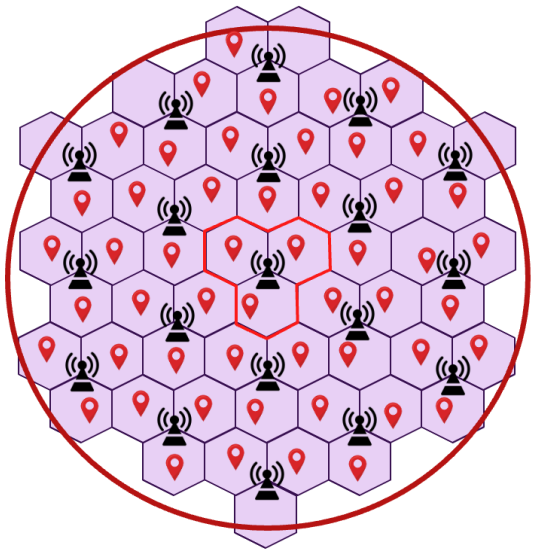

FIGURE 3-1 SYSTEM MODEL

We use the configuration parameters from 3GPP TR38.901 regarding the scenario and the path loss model. The relevant information about the base station antenna model is reported in **Table 3-1** from 3GPP TR 38.921 **[14].** For the UE antenna a gain of 0 dB is considered in line with Rep. ITU-R M.2412-0. Additionally, a gNB receiver noise figure of 5 dB and a thermal noise level of -174 dBm/Hz are considered. [ITU-R M.2412-0].

TABLE 3-1 3GPP BS ANTENNA MODEL [38.921]

Parameter	Description	Value
A_m (dB)	Front to back ratio	**30**
SLA_v(dB)	Side lobe suppression	**30**
φ_{3dB} (deg.)	Horizontal HPBW	**90**
θ_{3dB} (deg.)	Vertical HPBW	**90**
$G_{E,max}$(dBi)	Array element peak gain	**5.5**
L_E (dB)	Element loss	**2.0**
(M, N)	Number of radiating elements rows and columns	**(16, 8)**
Number of supported polarizations, P	-	**2**
d_h(m)	Horizontal element separation	**0.5λ**
d_v(m)	Vertical element separation	**0.5λ**
Horizontal coverage range (deg.)	-	**+/- 60**
Vertical coverage range (deg.)	-	**90 to 120**

The path loss model for the rural macro area selected from TR 38.901 is expressed as:

$$PL_{\text{RMa-LOS}} = \begin{cases} PL_1 & 10\text{m} \le d_{2D} \le d_{BP} \\ PL_2 & d_{BP} \le d_{2D} \le 10\text{km} \end{cases}$$

$$PL_1 = 20\log_{10}(40\pi d_{3D} f_c / 3) + \min(0.03h^{1.72}, 10)\log_{10}(d_{3D}) - \min(0.044h^{1.72}, 14.77) + 0.002\log_{10}(h)d_{3D}$$

$$PL_2 = PL_1(d_{BP}) + 40\log_{10}(d_{3D} / d_{BP}) \quad (3.1.4)$$

Where d_{3D} is the distance between the UE and the gNB, f_c is the carrier frequency, d_{BP} is the break point distance, the average building height is h = 5 m and the average street width is W = 20m. The UE are considered to be always in line of sight. We also consider the uplink budget at the satellite according to the parameters in D3.10.

The results are presented in **Figure 3-2** in the form of the cumulative distribution function (CDF) of the system throughput. In particular, the results are compared with a random allocation. It can be noted that the TN throughput obtained by means of the DQN is higher than that computed by the random allocation. Indeed, by offloading the users to the satellites, resources on the gNB on-ground were made available to other users, thus increasing the throughput of the TN.

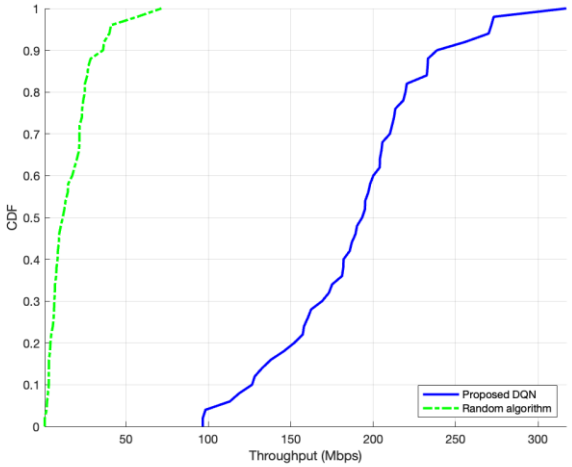

FIGURE 3-2 THROUGHPUT ANALYSIS FOR TN

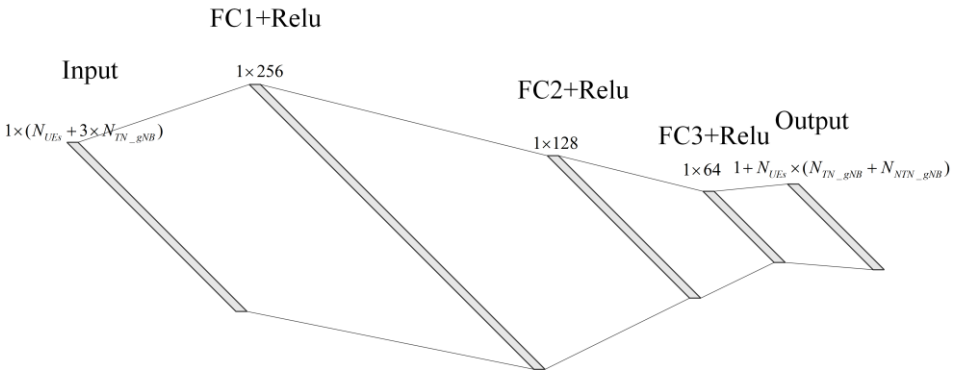

FIGURE 3-3 DQN STRUCTURE FOR TRAFFIC OFFLOADING

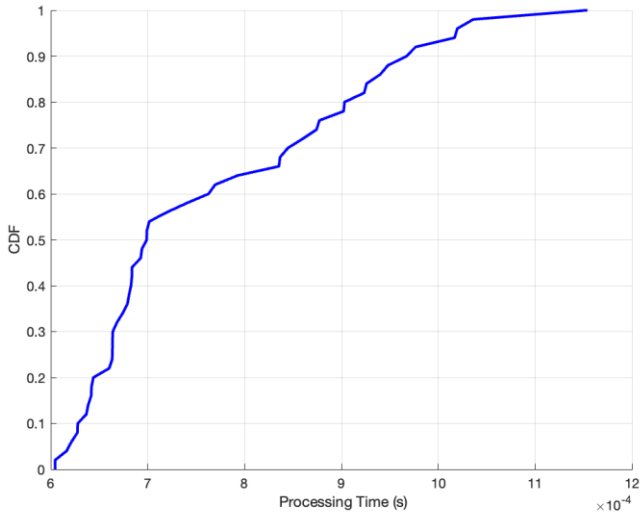

FIGURE 3-4 INFERENCE TIME

Regarding the DQN, its architecture is provided in **Figure 3-3**. As seen above, it consists of three fully connected layers, each one followed by the Rectified Linear Unit (ReLU) activation function.

From the computational complexity perspective, the number of multiply-and-accumulate units (MACs) is 170650 per inference. Clearly, given a specific computing platform, complexity metrics are directly related to inference latency and power consumption. Considering the MACs of the proposed DQN, the processing time is reported in **Figure 3-4** as a CDF. It is obtained considering different number of Monte Carlo simulations and different number of users, which can be between 24 and 49. It should be noted that the latency is smaller than 1 ms in 93% of the cases.

While it is a result of several simplifications, not accounting for real-world factors, this figure hints at the possibility of achieving real-time inference. It is worth emphasizing that these simulations have been carried out on a laptop.

Since the DQN is implemented on-ground, the processing time, which is negligible with respect to the feeder link round trip time, is not a bottleneck to the system performance in terms of latency.

Conclusion

This work focuses on a 6G NTN system with overloaded traffic and proposes a DQL-based optimization method for traffic offloading. Simulation results show that it is possible to maximize the throughput by offloading users to the satellite. The network function assumes one core network for both TN and NTN, going into the direction of a unified network. From the computational complexity perspective, the number of multiply-and-accumulate units (MACs) is 170650 per inference. Clearly, given a specific computing platform, complexity metrics are directly related to inference latency and power consumption. It should be noted that the latency is smaller than 1 ms in 93% of the cases. While it is a result of several simplifications, not accounting for real-world factors, this figure hints at the possibility of achieving real-time inference. It is worth emphasizing that these simulations have been carried out on a laptop. Since the DQN is implemented on-ground, the processing time, which is negligible with respect to the feeder link round trip time, is not a bottleneck to the system performance in terms of latency.

3.2 FRACTIONAL FREQUENCY REUSE

This section provides the system description, the mathematical model, and the simulation results of the dynamic FFR network function

3.2.1 System Description:

Considering the multi-beam satellite system that intends to mimic the dynamic FFR framework. For this purpose, there is a global model with B beams and each beam $b, b = 1, \dots, B$ is further split into local models to comprehend FFR framework as K outer beams with beamwidth β_k and there exist N inner beams inside with beamwidth β_n, where $N = K$ at any time t. Each outer beam $k, k = 1, \dots, K$ and inner beam $n, n = 1, \dots, N$ is serving the heterogeneous non-uniform user terminals (UTs) $U, u = 1, \dots, U$ with geo-coordinates (x_u, y_u). Each outer and inner beam intends to use distinct frequency resources depending on the adopted FFR scheme as defined later. Adopting the phased array system inherently involves the multi-antenna setup, each beam is formed by manipulating the phase and amplitude of the signals transmitted by M antenna elements. Hence, in general for a global model, the received signal r_n at the user u in b_{th} beam can be represented as,

$$r_b = \mathrm{h_u}^H \mathrm{w_b} \mathrm{s_u} + \sum_{b \neq \ell} \mathrm{h_u}^H \mathrm{w}_\ell \mathrm{s_u} + a_u \tag{3.2.1}$$

where H is Hermitian transpose, $\mathrm{s_{u,b}}$, $\mathrm{s_{u,\ell}} \in \mathbb{C}$ are the transmitted signals to user u in beam b and interfered signal from beam ℓ, respectively, $\mathrm{w_b}$, $\mathrm{w}_\ell \in \mathbb{C}^{M \times 1}$ are the adopted downlink beamforming vector forming the beam b and interfered beam ℓ, respectively, and $\mathrm{h_u} \in \mathbb{C}$ is the channel vector associated with users in a beam. Finally, a_u is the received Additive While Gaussian Noise (AWGN) at the user u, characterized by normal distribution with zero mean and variance σ_n^2.

Now extend this global model with B beams by splitting into local model with K outer and N inner beams to comprehend the FFR framework, where channel vectors, beamforming vectors and signal components are independently associated to inner and outer beams. Moreover, a criterion is to define to allocate the users to inner and outer beam as u_n and u_k, respectively. Hence, the received signals r_n and r_k at the user u_n and u_k, respectively can be represented as,

$$r_b = \begin{cases} r_n = \mathrm{y_n} + \mathrm{i_n} + a_{u_n} \\ r_k = \mathrm{y_k} + \mathrm{i_k} + a_{u_k} \end{cases} \tag{3.2.2}$$

The first term in (3.2.2) as $\mathrm{y_n}$ and $\mathrm{y_k}$ represent the desired received signal in inner and outer beam respectively, whereas $\mathrm{i_n}$ and $\mathrm{i_k}$ refer to the interference received signal from other beams allocated with identical frequency resources.

The $\mathbf{y_n}$ in (3.2.3) further comprises on the channel vectors, beamforming vectors and signal components independently associated to user u_n in desired inner beam and the interference $\mathbf{i_n}$ to user u_n from the other beams.

$$\begin{cases} y_n = \mathrm{h_{u_n}}^H \mathrm{w_n} \mathrm{s_{u_n}} \\ I_n = \sum_{\ell \neq n, f(n)=f(\ell)} \mathrm{h_{u_n}}^H \mathrm{w}_\ell \mathrm{s_{u_n}} \end{cases} \tag{3.2.3}$$

Similarly $\mathrm{y_k}$ and $\mathrm{I_k}$ aspects are associated to user u_k in desired outer beams as given in (3.2.4)

$$\begin{cases} y_k = \mathrm{h_{u_k}}^H \mathrm{w_k s_{u_k}} \\ I_k = \sum_{\ell \neq k, f(k)=f(\ell)} \mathrm{h_{u_k}}^H \mathrm{w_\ell s_{u_k}} \end{cases} \quad (3.2.4)$$

where $f(.)$ in (3) and (3.2.4) is a beam frequency mapping formula that assigns outer and inner beam to a specific set of frequency resources, depending on the adopted FFR scheme. Therefore ℓ could be any other outer or inner beam utilizing the identical frequency resources as of n in (3.2.3) and k in (3.2.4).

3.2.1.1 Beamforming

Consider a phased-array with M elements located at coordinates (x_m, y_m), operating at wavelength λ. For each beam b, steered toward elevation–azimuth (Θ_b, ϕ_b), the standard effective circular aperture that achieves a desired beam-width β_b is approximated by,

$$D_b = \frac{58.4\,\lambda}{\beta_b}, \qquad R_b = \frac{D_b}{2} \quad (3.2.5)$$

This defines a circular active region of radius R_b. Corresponding to the effective aperture, the active set of elements is computed as,

$$\mathcal{S}_b = \{m: \| (x_m, y_m) - (x_c, y_c) \| \leq R_b\} \quad (3.2.6)$$

where (x_c, y_c) is the antenna center coordinates. At the beam level, the normalized beamforming vector for beam b is then,

$$\mathrm{w}_b(\beta_b) = \frac{1}{\sqrt{|\mathcal{S}_b|}}\, \mathrm{z}_b . \exp(-j\frac{2\pi}{\lambda}(x_m \sin\theta_b \cos\phi_b + y_m \sin\theta_b \sin\phi_b)) \quad (3.2.7)$$

where $\mathbf{z}_b \in \{0,1\}^M$ is a binary mask vector with $[\mathbf{z}_b]_m = 1$ if $m \in S_b$. This relation in (3.2.7) ensures the unit-norm power distribution across the active elements i.e. $||w_b||_2 = 1$.

For a discrete set of beam-widths $\{\beta^{(v)}\}$, the beamwidth-specific codebook is,

$$\mathcal{C}(\beta^{(v)}) = \{\mathrm{w}_b(\beta^{(v)})\}_{b=1}^{B}, \quad (3.2.8)$$

In addition, at the user level, the array response ξ toward the user direction (θ_u, ϕ_u) is,

$$\xi_{u,b} = \mathrm{w}_b(\beta_b)^H (\mathrm{m}_i . a(\theta_u, \phi_u)). \quad (3.2.9)$$

This compact formulation packages beamforming vectors into codebooks indexed by beamwidth that can be call up later when reinforcement learning (RL) framework apply action. The beam pattern is considered regular one, such as an example is drawn in **Figure 3-5**.

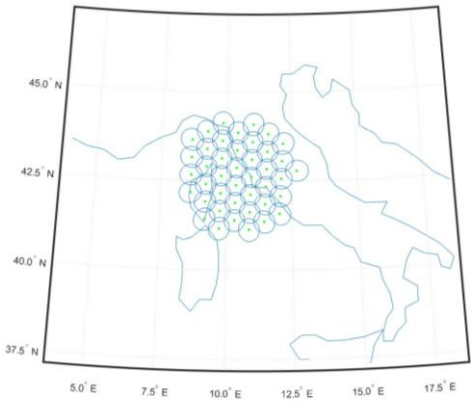

FIGURE 3-5 BEAM LAYOUT WITH REGULAR BEAM CENTER POSITIONS

3.2.1.2 User Classfication

The splitted local model in (3.2.3) and (3.2.4) explicitly requires the classification strategy to assign users to inner and outer regions in order to evaluate the terms $\mathbf{y_n}$ and $\mathbf{y_k}$. To do so, a distance-based metric is used to indicate whether a user u is classified as u_n or u_k. Since the location coordinates of beam centers and the users are known, a threshold $\gamma_n(\beta_n)$ is defined based on the haversine distance between the beam center and inner beam boundary. This threshold is then used to classify each user into the corresponding region everytime inner beamwidth is updated, as follows,

$$u_n = \begin{cases} 1 & if \quad d_{((x_b,y_b),(x_u,y_u))} \leq \gamma_n\,(\beta_n) \\ 0 & otherwise \end{cases} \tag{3.2.10}$$

3.2.1.3 Channel State Information

Following the user allocation to beams, the subsequent step is to model the channel state information (CSI) defined using channel vectors $\mathrm{h_{u_n}}$ and $\mathrm{h_{u_k}}$ in (3.2.3) and (3.2.4) for both inner and outer beams, respectively. Considering, at system level and concatenating all CSI vectors, it represents the system channel matrix for all inner and outer beams such as $\mathbf{H_n^*} \in \mathbb{C}^{U_n \times N}$ and $\mathbf{H_k^*} \in \mathbb{C}^{U_k \times K}$. For better understanding, the channel matrices are further split as $\mathbf{H}^* = DR$ for both inner and outer systems, where the matrix R determines by the satellite antenna radiation pattern, transmission gain, receiver antenna gain, free space path loss and the noise power while D represents the phase-slant due to different propagation path between the users. Considering this, any entry of the matrices $\mathrm{H_n^*}$ and $\mathrm{H_k^*}$ can be given as,

$$\begin{cases} h_{u_n,n} = \dfrac{\sqrt{G_{t(u_n,n)} G_{r(u_n,n)} G_{tx(u_n,n)}}}{(4\pi \frac{d_{u_n n}}{\lambda})\sqrt{\kappa B w_n}} \\ h_{u_k,k} = \dfrac{\sqrt{G_{t(u_k,k)} G_{r(u_k,k)} G_{tx(u_k,k)}}}{(4\pi \frac{d_{u_k k}}{\lambda})\sqrt{\kappa B w_k}} \end{cases} \tag{3.2.11}$$

where G_t and G_r represent the transmission gain and receiving gain to user in beam respectively while G_{tx} is the gain defined by transmission radiation pattern at the user location, can be acquired from array response ξ in (3.2.9), d is the distance between satellite antenna and the user, λ is the wavelength, Bw is bandwidth allocated to beam and κ is Boltzmann constant.

If the users u_n and u_k are located in the desired inner and outer beams, respectively then $|h_{u_n,n}|^2$ and $|h_{u_k,k}|^2$ represent the channel gain that the users are experiencing in their desired inner and outer beams. The users in adjacent beams may utilize the common frequency bands, therefore, the signal received by any user should also include the interference caused by the other users using the same frequency sub-bands in other beams. Hence, the SINR of this local model for any user in its desired beam can be formulated as,

$$\begin{cases} SINR_{u_n,n} = \frac{P_n|h_{u_n}|^2}{\sum_{n\neq\ell, f(n)=f(\ell)} P_\ell|h_{u_n}|^2+N_o} \\ SINR_{u_k,k} = \frac{P_k|h_{u_k}|^2}{\sum_{k\neq\ell, f(k)=f(\ell)} P_\ell|h_{u_k}|^2+N_o} \end{cases} \quad (3.2.12)$$

where P_n and P_k are transmitting powers allocated to inner and outer desired beams respectively and N_0 is white Gaussian noise.

Following the SINR relation formulated in (3.2.12), the offered capacity C^{off} to any user u in beam n and k, can be formulated as,

$$C^{off}_{(\forall u,b)} = \begin{cases} \frac{1}{U_n}\left(\sum_{u\in U_n} Bw_n \log_2(1+SINR_{u,n})\right) \\ \frac{1}{U_k}\left(\sum_{u\in U_k} Bw_k \log_2(1+SINR_{u,k})\right) \end{cases} \quad (3.2.13)$$

Bw donates the bandwidth allocated based on FFR scheme.

3.2.1.4 Fractional Frequency Reuse Scheme

In this work, the well-established strict fractional frequency reuse (S-FFR) scheme is adopted, which strategically manages frequency allocation across beam regions to enhance system performance. Specifically, the inner region of each beam employs a frequency reuse (FR) factor of 1, meaning that users within the inner beam region share the common frequency sub-bands. This approach is in-fact suitable for the inner region, where users generally experience high SINR for being near to beam center. In contrast, the outer beam region employs a FR factor of 4, ensuring that the users in adjacent beams are assigned disjoint frequency resources as shown in **Figure 3-6**. This stringent allocation is necessary because the SINR in the outer beam region is more vulnerable to co-channel interference for being at the beam edge. Thereby, assigning the disjoint frequency resources enhances the user experience at the beam edges to an extent.

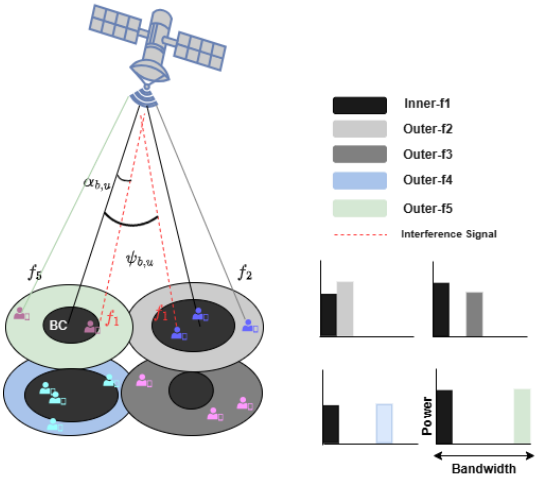

FIGURE 3-6 THE DYNAMIC STRICT FFR SCHEME FOR MULTI-BEAM SATELLITE SYSTEM

Further interpreting **Figure 3-6**, $\alpha_{b,u}$ donates the azimuth angle of the desired user terminal u with respect to the center of desired beam b.This angle is used to calculate the relative beam gain for desired user. Similarly, $\psi_{b,u}$ represents the azimuth angle of an interfering user terminal with respect to the same beam center. This angle is used to determine the relative beam gain from the interfering user to desired beam center, as used in (3.2.12).

3.2.2 Problem Formulation

Since the objective of this paper is to 1) tune the power control of inner and outer beam independent to each other, under the total power constraint 2) adjust the inner beamwidth β_n as a function of heterogeneous user terminal distribution. The ultimate goal is to minimize the normalized capacity deviation (NCD) as the difference between the throughput request C^{req} and offered throughput C^{off}. Hence, the optimization problem is formulated as,

$$\begin{aligned} \underset{\{P_n\},\{P_k\},\{\beta_n\}}{\text{minimize}} \quad & \mathcal{NCD} = \frac{\sum_{b\in B}\sum_{u\in U}|C_{u,b}^{req}-C_{u,b}^{off}|}{\sum_{b\in B}\sum_{u\in U}C_{u,b}^{req}} \\ \text{subject to} \quad & c_1: \sum_{n\in N} P_k \le P_{\max} \\ & c_2: \sum_{k\in K} P_k \le P_{\max} \\ & c_3: Bw_n + Bw_k \le 2\frac{Bw}{4+1} \quad \forall n,k \end{aligned} \tag{3.2.14}$$

and equivalently formulated as,

$$\mathcal{NCD}\,(P_n,P_k,\beta_n) = \frac{\sum_{b\in B}|C_b^{unmet}+C_b^{unused}|}{\sum_{b\in B}\sum_{u\in U}C_{u,b}^{req}}$$

$$C_{\text{unused},b} = \sum_{u\in U}\begin{cases} C_{u,b}^{off}-C_{u,b}^{req}, & \text{if} C_{u,b}^{off} > C_{u,b}^{req}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.2.15}$$

$$C_{\text{unmet},b} = \sum_{u \in U} \begin{cases} C_{u,b}^{req} - C_{u,b}^{off}, & \text{if} C_{u,b}^{req} > C_{u,b}^{off}, \\ 0, & \text{otherwise.} \end{cases} \quad (3.2.16)$$

In this problem, the search variable is P_n, P_k, β_n while the objective function is to minimize the NCD. Constraints c_1 and c_2 represent the power allocation to inner and outer beam n and k respectively, should not exceed maximum available power resources P_{max}. Constraints c_3 dedicates to S-FFR where the total bandwidth resources per beam partitioned into 5 sub-bandwidths since inner beam region entitled to use FR 1 and outer beam region sets to use FR 4.

3.2.3 Simulation Framework

3.2.3.1 Case 1: Beamwidth Optimization using Deep Q Learning

In order to solve the FFR optimization problem, at first only the beamwidth is optimized, the idea herein is to investigate the Deep Reinforcement Learning (DRL) based Deep Q Learning (DQL) framework. The motivation for selecting DRL for optimization task is due to the fact that the state and action space in the FFR network function are considered discrete, and DQL is an appropriate choice for handling the discrete states and action spaces.

Typically, in DQL, an agent receives state s_t from state space $\mathbb{S}$ and performs an action a_t from action space $\mathcal{A}$. After executing action a_t, the agent receives a reward r_t and moves to the next new state s_{t+1}. The agent continues the process until reaching the terminal state. To execute an action, the agent follows a policy $\pi(a_t|s_t)$ for mapping from given state s_t to action a_t as shown in **Figure 3-7**. The ultimate objective of the agent here is to maximize the total accumulated reward. In Q learning, there exists action value functions typically expressed as $Q_\pi(s,a)$ and $Q^*(s,a)$, where the former depicts the expected return for selecting action a in state s, while the latter represents an optimal action value function i.e. $Q^*(s,a) = max_a Q_{target}(s,a)$ by following any policy for state s and action a. Meanwhile, in DQL, a Deep Neural Network (DNN) is used to approximate the optimal action value function.

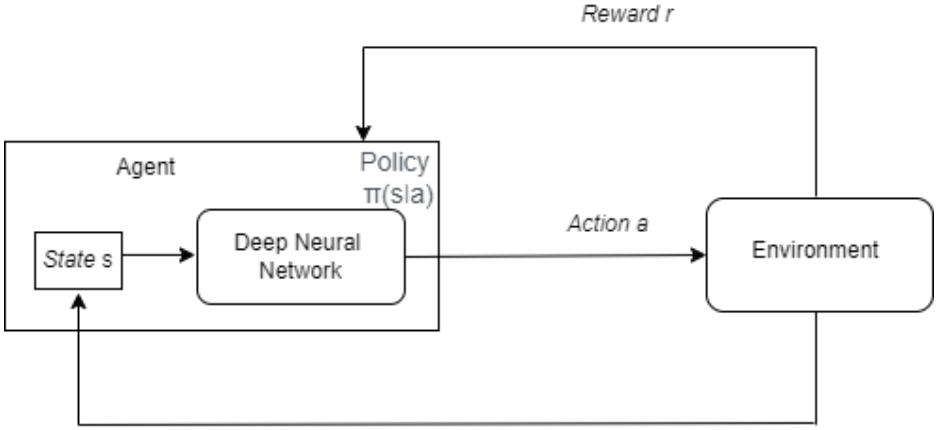

FIGURE 3-7. TYPICAL DQL WORKFLOW

To map this DQL framework into the problem of FFR defined above, the DQL model uses the five attributes as *states* for each beam b at time t such as, $s_b^t = \{U_n, U_k, C_b^{unmet}, C_b^{unused}, C_b^{off}\} \in \mathbb{R}$. The FFR problem defined herein is a centralized training problem, which means that the

action of changing inner beamwidths for one beam will impact the overall system performance. So, the state is defined as system level instead of per beam.

Given the state's representation, the state vector $\boldsymbol{s_t}$ will be B x 5, where B is equal to number of beams.

Action Space: The action space is the discrete value of inner beamwidths β_n. In this case, the action space uses 3^o, 4^o and 5^o of beamwidths, which means the inner beam can take any of these values while the outer beamwidth is set to 6^o

Reward: Since the objective is to minimize the NCD given in (3.2.14), and in DRL, the algorithm maximizes the reward, therefore the reward is given in negative such that,

$$r = -NCD \tag{3.2.17}$$

DQL Implementation Steps and Parameters:

1) The first step of DQL is defining two networks
    a. Q-Network
    b. Another Q network, referred to as Target Network having exactly same attributes as main Q-network

The defined networks contain the number of layers and neurons per layer and the activation functions. The input layer size is typically the same as the state size i.e. B x 5 in our case and output layer size is same as the total number of actions i.e. 3^B, as we are using 3 possible beamwidths values.

2) The Bellmen Equation:

$$Q^*(s,a) = r + \gamma \, max \, Q_{target}(s`,a) \tag{3.2.18}$$

This equation is the heart of DQL and the main character in the training of DQL. The neural network is used to approximate the optimal action-value function $Q^*(s,a)$, r is the reward and γ is the parameter to prioritize the future reward since $Q_{target}(s`,a)$ is based on next state not the current state.

Environment Simulation

The *Environment* simulation considers the LEO satellite operating at an altitude of 600km, equipped with a phased array antenna system designed for flexible beamforming. The antenna employs $M = 512$ radiating elements arranged in a circular aperture with element spacing $\delta_M = 0.7\lambda$. The antenna system provides high directivity through a large number of elements and supports adaptive control of radiation patterns for simultaneous generation and steering of multiple Earth-fixed beams with dynamically adjustable beam-widths. The system operates in the Q/V frequency band, with the downlink configured in Q band at a carrier frequency $f_c = 40$GHz. The outer beamwidth is kept fixed at 6^o while the inner beamwidth intend to dynamically adjust to minimize the objective function in (3.2.14), but in case 1, the optimization variable is only inner beamwidth while the power is fixed. The resulting joint action of inner beamwidth to this *Environment* determines the reward as designed in (3.2.17).

TABLE 3-2 FFR SYSTEM PARAMETERS FOR DQL ENVIRONMENT

Parameters	Values
Satellite Altitude	600 km
Operating Frequency f_c (downlink)	40 GHz
Radiation Elements M	512
Element Spacing δ	0.741λ
Antenna Geometry	Circular
Element Gain	5.31 dB
Total Bandwidth	2 GHz
Receiver G/T	10 dB.K^{-1}
Outer Beamwidth	6°
Possible Inner Beamwidths	$\{3,4,5\}^{\circ}$

Model Training and Testing:

The training process is organized into episodes, each representing the defined time steps. During the model training, the agent iteratively interacts with the established *Environment*, exploring actions at each time step and returns the reward and next state. Over the episodes, this process enables the agent to converge towards strategies that maximize long-term objective. To compute the current and next states, the training required the user traffic dataset and the corresponding requested demands. To this end, one of the preeminent datasets comprising on the realistic terminal positions for the maritime, provided in D4.3 [4] is used. **Figure 3-8** illustrates a snapshot of the normalized aggregated user traffic distribution, depicting the variations across consecutive time steps. The demand for each user is assigned in a probabilistic manner and modeled as discrete random variable D, where the probability of a user requiring a demand level is given by $P(D = C_{u,b}^{req}) = p_j$ with $\sum_j p_j = 1$. This probabilistic assignment captures heterogeneous user by associating each demand level with a predefined probability.

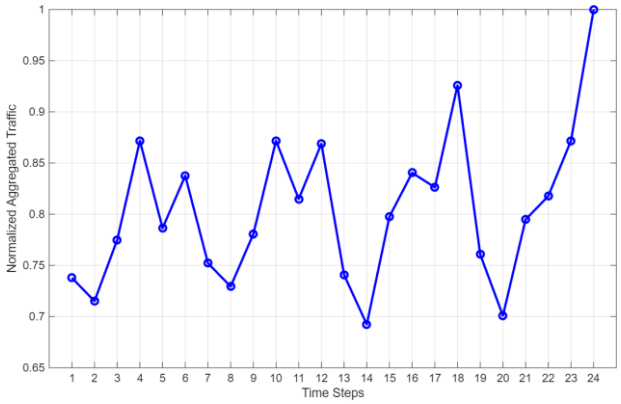

FIGURE 3-8 NORMALIZED AGGREGATED USER TRAFFIC ACROSS CONSECUTIVE TIME STEPS

After setting up the *Environment* and datasets, the model training process is initialized using an ϵ-greedy exploration strategy. The training relevant parameters are given in **Table 3-3**.

TABLE 3-3 TRAINING RELEVANT PARAMETERS

Parameters	Values	Explanation

Number of Hidden Layers	2	The hidden layers between the input and output layer of network that optimizes the weights and activates the neurons.
No. of neurons	128	Each layer contains 128 neurons except input and output layer.
Replay Memory Size	5000	Replay Buffer stores the experience. Actually, DQL algorithm trains the network and learn through this Replay Buffer where the values of states, actions and next states and rewards are stored every time. The DQL randomly picks a batch of experience and calculates the MSE loss between target Q network and the main Q network and try to minimize the difference.
Batch Size	64	This batch size is 64 which means DQL takes randomly 64 experiences from Replay buffer for training
Target Network Update	Soft Update (0.01)	After every few episodes, the Target network copies the weights from main Q network. The reason behind doing this is the need of target network to slowly follow the main Q-network's learning progress without being too reactive.
Learning Rate	1e-4	Learning rate controls how quickly the Q-network adjusts its predicted Q-values based on the error.
Epsilon Decay	0.9817	The epsilon decay is related to exploitation vs exploration concept. The epsilon value goes from 1 to 0.01 with the given decay value. During the initial state when epsilon has higher value, the agent will randomly choose the action (which is called exploration). With the epsilon decay, the agent becomes greedy and start exploiting the learnt policy.
Step Size	24	In DQL, there are training episodes. In each episode, there are steps where the agent takes actions based on epsilon decay value. In our FFR case, 24 is chosen.
Episodes	300	The total no. of training episodes. During each step of the training episode, I provided the random hours from any of the 6 days to maintain the randomness of dataset so that network doesn't overfit just by learning the traffic trend. So, each step in each episode takes a random hour from any of the 6 days

The mean square loss (MSE) and the episode reward (NCD) are depicted in **Figure 3-9** and **Figure 3-10**.

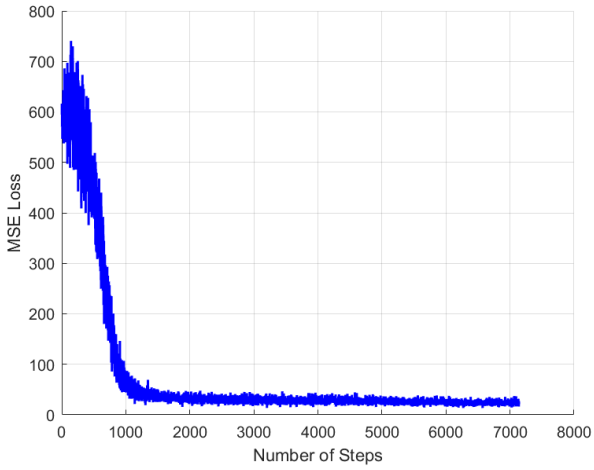

FIGURE 3-9 MSE LOSS DURING DQL TRAINING AS A FUNCTION OF NUMBER OF STEPS

The MSE loss updates between the target Q network and the main Q network. The x-axis is relevant to each step. There are 300 episodes, and each episode contains 24 steps so the total number of steps are 300 x24 = 7200. It can be seen that the loss is stabilizing nearly 3000 steps which means Q network learns to meet the target network expectations but it doesn't mean that the agent is leant the policy at this stage since agent is still in the process exploration and policy learning which we can see in **Figure 3-10** .

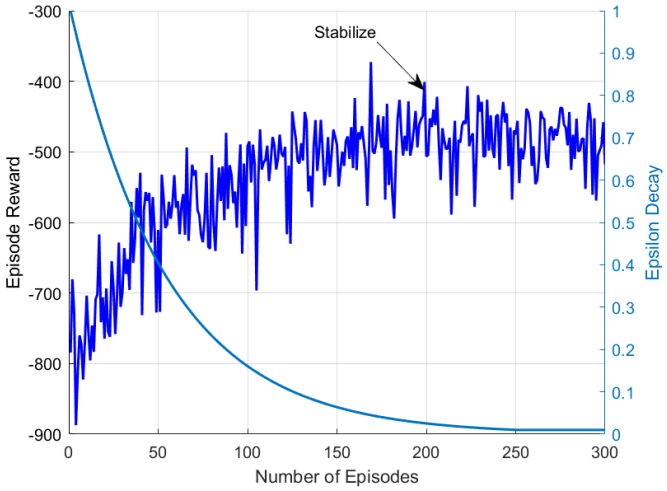

FIGURE 3-10 THE EPISODE REWARD

The episode reward herein is the sum of -NCD acquired from 24 steps. As in each step, the agent applies action and in return we get -NCD reward. So, each episode contains 24 steps which means in each episode we will have the sum of all 24 -NCD values of actions applied by agent on random sample. Technically, the episode reward should be minimized as the agent starts exploitation. It can be seen that the episode reward is nearly -900 in the start and then starts improving with epsilon decay. Lower the epsilon value, the agent means applying its learnt policy. After 200 episodes, when epsilon reaches less than 0.1, the agent uses the greedy action learnt during exploration stage. Since in each episode, we have a random

combination of samples, we will have different minimum -NCD based on user traffic. However, eventually, the stable episode reward provides insights on learning the correct policy.

The network is then tested with unseen data for 24 samples. The comparison is made among,

1) When agent takes random action (100 Monto Carlo Trails)
2) When all inner beams use beamwidth 3^o
3) When all inner beams use beamwidth 4^o
4) When all inner beams use beamwidth 5^o
5) Brute Force as Benchmark (By testing each optimal action through exhaustive search, 100% optimal action)
6) DQL (By testing the trained agent)

The DQL performance is near to Benchmark (Brute Force) which tells the efficient selection of actions on unseen data. The average NCD over 24 hours for Brute force is 17.15 and the average NCD over 24 samples for DQL is 14.32 which takes around 85% accuracy on real dataset as shown in **Figure 3-11**.

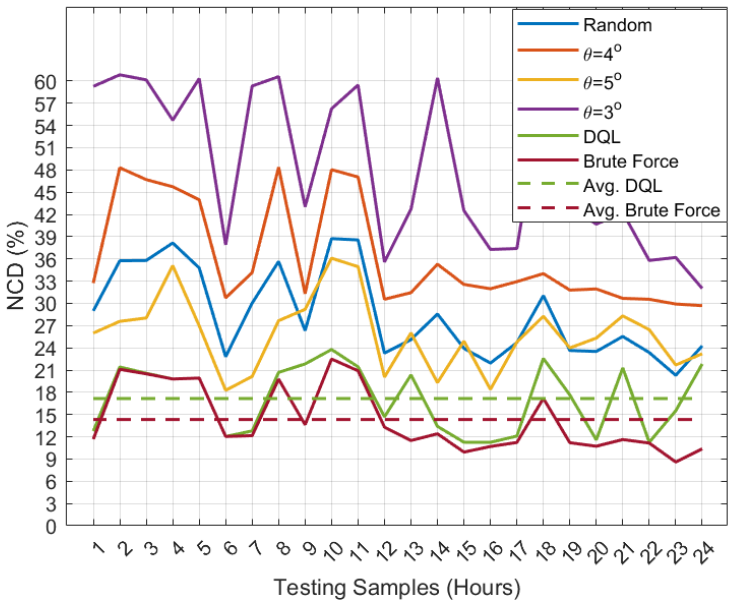

FIGURE 3-11 MODEL INFERENCE OF DQL ON FFR INNER BEAMWIDTH OPTIMIZATION

3.2.3.2 Case 2: Joint Optimization using Multi-Agent Framework

As seen in previous subsection, DQL is a straightforward approach for value-based optimization, but its scalability limitations render it less suitable for *joint optimization* of parameters in the multi-beam scenarios. The key limitations arise from the fact that DQL must select the greedy action from the entire joint action space, whose cardinality grows exponentially with the number of beams and parameter choices. Formally, consider a system with B beams, each associated with 3 possible values of inner-to-outer beamwidth ratio and 3 discrete values power levels for inner and outer beams each, the resulting action space $\mathcal{A}$ is of size $27^5 \approx 1.4 \times 10^7$. This exponential growth of joint action space makes DQL

computationally intractable for multi-beam joint optimization problems, motivating the adoption of more scalable learning framework such as multi-agent reinforcement learning (MARL).

The MARL framework adopted here follows the principle of centralized learning and decentralized execution (CTDE). Under this paradigm, it allows each agent to learn its own action-value function, while ensuring that the individual action-value function Q_i optimization collectively contributes to the optimization of the joint action value function Q_{jt} for all agents, which means that the action space $\mathcal{A}$ reduced to per agent action space i.e. the size of 27^1, regardless of number of beams.

The adopted architecture is composed of two distinct neural networks, each capturing different aspects of value functions:

1) **Agent-Centric Network**: For each agent $i \in \{1,2,\dots,B\}$, an action-value function Q_i is estimated from its local state $s_i^t \in \mathbb{R}^{d_s}$ and local action $a_i^t \in \mathcal{A}$. The mapping can be written as f_i: (s_i^t, a_i^t) $Q_i \in \mathbb{R}$. Here Q_i outputs a scalar value representing the utility of choosing action a_i^t in the given s_i^t during step t. As seen, this agent-centric network is employed by each agent i to determine its own action by calculating the action value for a given state s_i^t. This network subsequently produces a unified output by aggregating the individual action-value estimates Q_i of all agents, expressed as $Q' = \sum_{i=1}^{N} Q_i(s_i^t, a_i^t) \in \mathbb{R}$.

2) **Joint Q-Network**:The joint Q-network is responsible for estimating the single joint Q_{jt} value. This networks takes as input the optimal action vector sampled by all agents from their respective individual action-network and outputs the corresponding joint Q_{jt} values. The mapping can be written as f_{jt}: (S^t, A^t) $Q_{jt} \in \mathbb{R}$. The selection of optimal action within each individual network is performed using decentralized policies, where each agent determines its action through a linear-time argmax operations over its local action-value estimates.

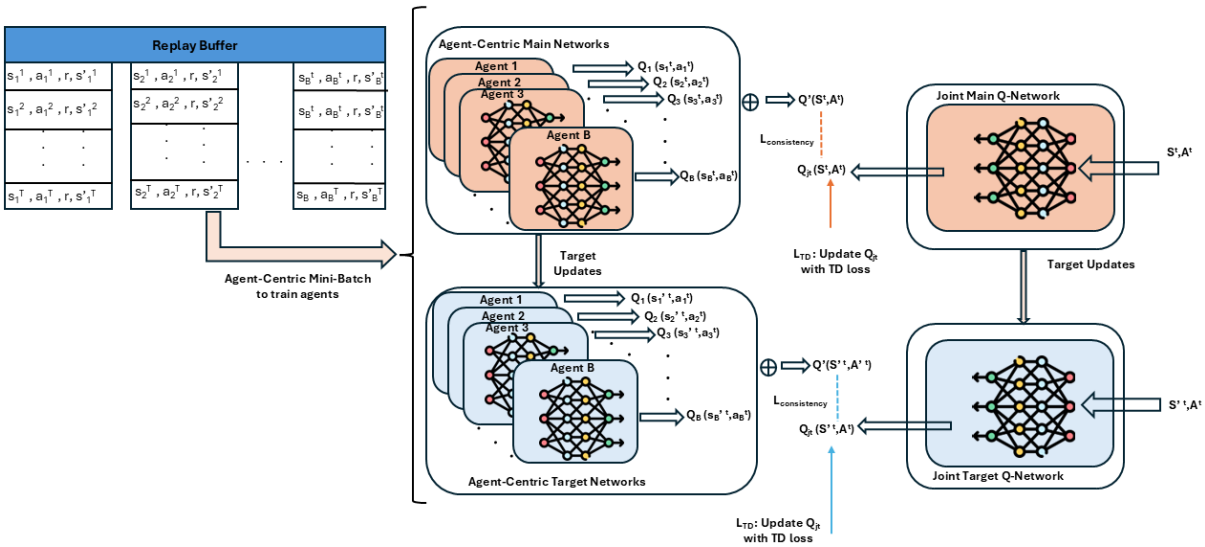

FIGURE 3-12 CTDE MULTI-AGENT ARCHITECTURE

To enable centralized training in the adopted MARL framework, two key challenges must be addressed. First, the joint action function must accurately approximate the true action-value. This is typically achieved using the conventional behaviour of DQN, where the temporal-difference (TD) is minimized by aligning the predictive Q-values with those generated by target network. Secondly, the action-value function Q' acquired from individual action-value network must be constrained to track the joint action-value function Q_{jt} in such a way that their optimal actions remain equivalent. These consistency requirements ensure that decentralized policy at the agent level aligns with global optimum obtained through centralized training. Without loosing the generality, the global loss function is used as formulated for QTRAN [13],

$$L(S^t, A^t, r^t, S'^t, \theta) = L_{TD} + L_{consistency} \tag{3.2.19}$$

where θ represents the learnable parameters of the neural network. L_{TD} corresponds to the loss used to approximate the true action-value by minimizing the TD as in DQN given in (3.2.20),

$$L_{TD} = (Q_{jt}(\mathrm{S^t}, \mathrm{A^t}) - y^{dqn}(r, \mathrm{S'^t}; \theta^-))^2 \tag{3.2.20}$$

where y^{dqn} represents the Bellman equation in (3.2.18)

$$y^{dqn}(r, \mathrm{S'^t}; \theta^-) = r^t + \gamma Q_{jt}(\mathrm{S'^t}, \bar{\mathrm{A}}'^t, \theta^-), \quad 0 \le \gamma \le 1 \tag{3.2.21}$$

where $\bar{\mathrm{A}}'^t = [\mathrm{argmax}_{a_i^t} Q_i(s'^t, a_i^t; \theta^-)]_{i=1}^B$, γ is discounted factor that determines how much the agent values the future reward, and $\boldsymbol{\theta^-}$ is periodically copied from the learnable parameters θ of the main network to the target network to ensure training stability. The consistency loss can be computed as follows,

$$L_{\mathrm{consistency}} = \begin{cases} L_1 = \lambda_1(\hat{Q}_{jt}(S^t, \bar{A}^t) - Q'(S^t, A^t))^2, \\ L_2 = \lambda_2(\min\{\hat{Q}_{jt}(S^t, A^t) - Q'(S^t, A^t), 0\})^2, \end{cases} \tag{3.2.22}$$

$\hat{Q}_{jt}(S^t, \bar{A}^t)$ donates the fixed value of Q_{jt} obtained by stopping the gradient flow in order to stabilize training. By stopping the gradient, $\hat{Q}_{jt}$ is treated as constant target during the loss computation.

This CTDE based MARL framework is adopted for joint inner-to-outer beamwidth ratio and power optimization in FFR with the following parameters:

Environment: The *Environment Simulation* is similar to the one provided for Case 1.

State Space: The State Space is similar to the one formulated for Case 1 for DQL.

Action Space: In this case, the joint action action space $\mathcal{A}$ is now defined as a composite action space constructed by combining the beam-width and power decisions:

$$\mathcal{A} = \{\mathcal{W} \times \mathcal{P} \times \mathcal{P}\} \in \mathbb{R}^{3 \times d_a} \tag{3.2.23}$$

where $\mathcal{W}$ is the set of possible inner beamwidth values, $\mathcal{P}$ is the set of discrete possible transmit power levels that inner and outer beams can take and d_a is the dimensionality of action space, equal to the total number of available actions in joint action space $\mathcal{A}$ is therefore:

$$|\mathcal{A}| = |\mathcal{W}| \, . \, |\mathcal{P}^2| \tag{3.2.24}$$

Model Training

The training process of the adopted MARL framework for joint optimization follows the general principles of DQL, incorporating an experience replay buffer $\mathcal{D}$ and the periodic updates of target networks for each local agent. The distinction lies in the inclusion of a consistency loss, formulated in (3.2.19) that enables the CTDE paradigm. The training steps are as follows:

Step 1: Define the individual action-value network for each agent that maps its local state to action values. The input dimension of the individual network corresponds to the local state size of agent i i.e. $\mathbb{R}^{d_s}$, and the output dimension equals to the total possible actions in action space $\mathcal{A}$ i.e. $\mathbb{R}^{3\times d_a}$.

Subsequently, define the joint action-value network with the input dimension of $\mathbb{R}^{|h|\times B}$, where $|h|$ donates the size of hidden vector from individual network since the joint action-value network shares the parameters at the lower layer of individual action-value network. The output is a single scalar joint Q-value.

Define the corresponding target networks similar to main networks and initialize with the random weights. In addition, initialize the replay buffer $\mathcal{D}$ with pre-defined capacity.

Step 2: Specify the step size, and apply a randomly generated action vector in order to obtain the initial state, which serve as the input to each individual network.

Step 3: Employ a ϵ −greedy policy for action selection, whereby each agent either select a random action index $a_i^t \in \mathcal{A}$ with probability ϵ or exploits its individual Q-network to compute the action-values and choose the action index corresponding to maximum Q-value with probability $1-\epsilon$. Extract the action from the action space $\mathcal{A}$ using the selected action index and apply within FFR environment to obtain the tuple $(s_i^t, a_i^t, r_t, s_i'^{t+1})$, and is then stored in the replay buffer until a sufficient number of experiences are accumulated for training.

Step 4: Sample a mini-batch of experiences of size $\mathcal{M}$ from the replay buffer $\mathcal{D}$ and input the mini-batch into the individual action-value network, and the joint action-value network, then updates the targets of that mini-batch and compute the loss function given in (3.2.19).

Step 5: Perform a gradient descent step on the loss function to update the learning parameters θ via back propagation.

Step 6: Repeat the Step 2 to 5 until reaching the pre-defined terminal episode.

Follwing the training step provided, the episode reward is provided as,

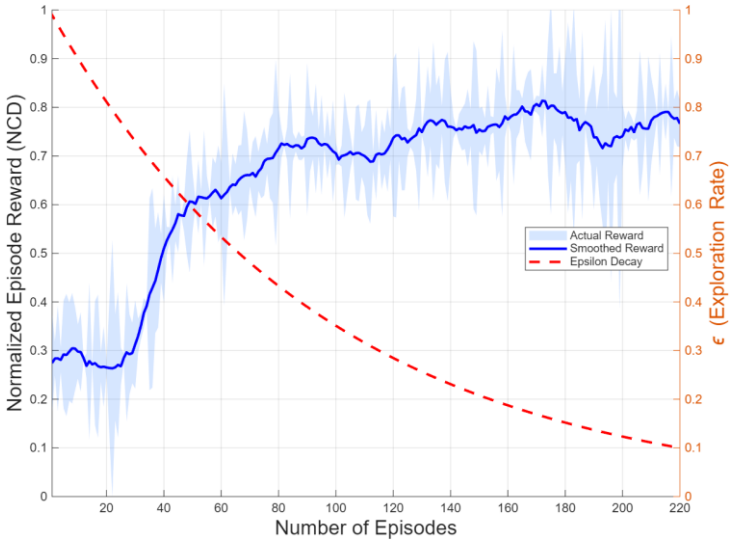

FIGURE 3-13 OVERALL EPISODE REWARD USING MARL FRAMEWORK

As illustrated in **Figure 3-13**, at the initial stages of training, the agents predominantly explores the environment through random actions, which results in relatively lower episode reward. As the training progresses, the exploration rate gradually decays, the agents begin to adopt more greedy policy, leading to steady increase in reward until approximately 170 episodes. Beyond this point, the reward stabilizes, indicating that the network has begun to converge towards a meaningful set of actions. The fluctuation observed in the reward during training is relevant to the heterogeneity of the environment i.e. different states may have different maximum achievable rewards at each time step. Thereby, the reward function naturally varies as the agents encounter different states throughout the episodes.

Moreover, this behaviour also aligns with the expected dynamics, indicating that all agents are cooperatively adapting their actions. This impact is further illustrated in **Figure 3-14** , where each agent exhibits the same trend and progressively maximizes its individual reward. The outcomes in **Figure 3-13** and **Figure 3-14** depict that the agents are not only learning individually but also adapting their policies in a cooperative manner under the centralized training framework, ensuring that the training process converges towards the robust cooperative policy.

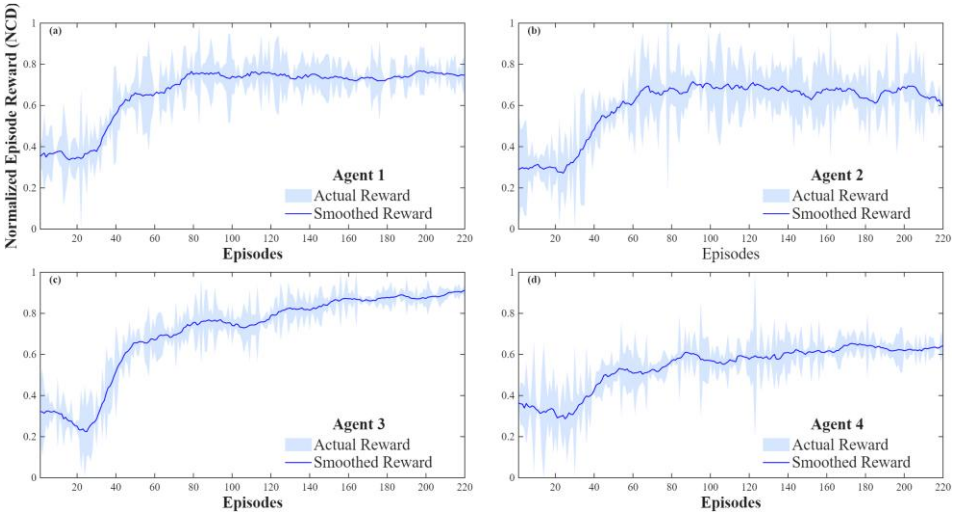

FIGURE 3-14 PER-AGENT NORMALIZED NCD AS A FUNCTION OF NUMBER OF EPISODES

Model Inference

After training the multi-agent model, the performance comparison is performed with multiple benchmarks to evaluate its inference capabilities in optimizing FFR framework as follows:

Genetic Algorithm (GA): To identify the near-optimal combination of actions for comparison, genetic algorithm is employed. This algorithm iteratively evolves a population of candidate solutions through biologically inspired operations such as selection, cross-over and mutation, guided by our FFR reward function i.e. -NCD. This evolutionary process converges toward high-performance configurations with reduced computational cost compared to exhaustive search (Brute Force). GA is thus employed as benchmark to assess the efficiency and inference capabilities of the adopted multi-agent learning model.

Independent DQL (I-DQL): is a non-cooperative multi-agent scheme, wherein each agent performs decentralized learning and independently selects action following the same ε-greedy policy. Unlike the cooperative multi-agent framework, I-DQL operates without inter-agent cooperation or shared state information, and each agent optimizes its policy solely based on local observations.

Fixed Beamwidth (FB): The FB represents a static baseline scenario in which each beam maintains a constant beamwidth throughout operation, independent of network dynamics, user distribution or traffic dynamics, while adhering to the total power constraint.
Random Actions: The random actions refer to the scheme in which each agent selects beamwidth and power allocation randomly at every time step.

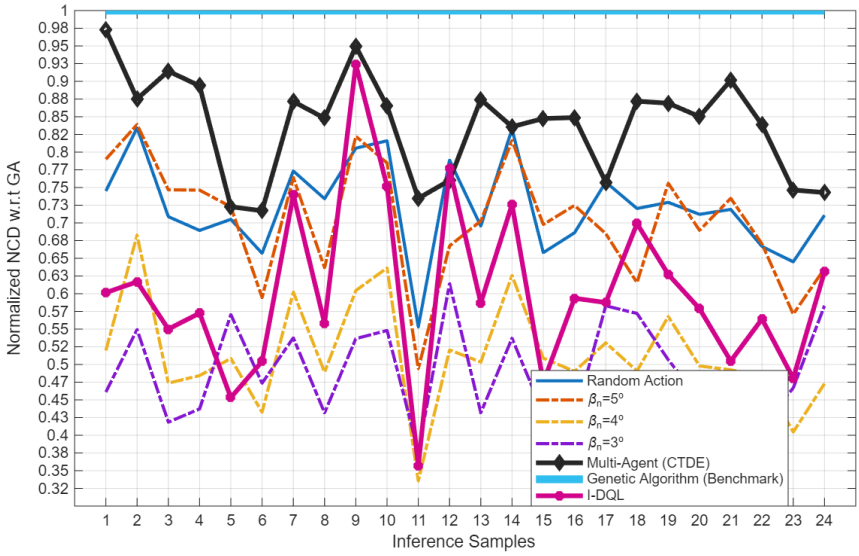

FIGURE 3-15 MARL (CTDE) MODEL INFERENCE PERFORMANCE COMPARISON

Figure 3-15 depicts that the CTDE outperforms the fixed beamwidths and non-cooperative I-DQL in most of the testing samples. Similarly, Figure 3-16 shows the accuracy of all the approaches with respect to the optimal genetic algorithm. This gives insights on accuracy in terms of percentages such as CTDE framework provides around 84% accuracy compared to near-optimal benchmark while non-cooperative I-DQL performance is around 60% which is due to the fact that the agents are independently selecting action leading to poor performance.

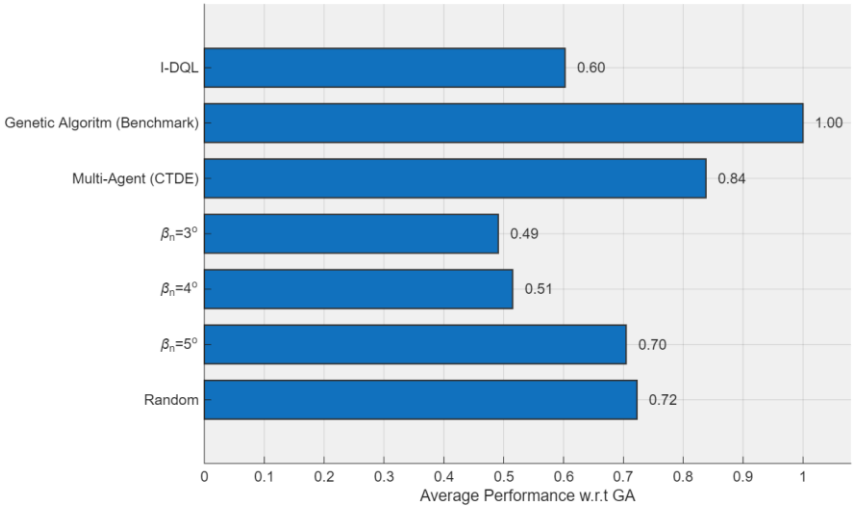

FIGURE 3-16 AVERAGE NORMALIZED NCD PERFORMANCE COMPARISON

Conclusion

FFR AI network function is designed to deploy as xApp and rAPP within AI-enabled RIC. Its core objective is dynamically controlling the inner beamwidth and power allocation in accordance with user-specific traffic requirements and spatial distribution. During simulations, it was observed that jointly optimizing the beam radius and transmit power levels substantially mitigates capacity imbalance across beams, thereby preventing system from over and under-provisioning radio resources. To investigate this, two simulation scenarios are considered. In the first case, the transmit power kept fixed, while only beamwidth was adaptively tuned to minimize the NCD using traditional DQL. Although effective for single variable control, the DQL framework struggled to efficiently handle the joint optimization of beamwidth and power allocation due to exponentially large action space inherent in multidimensional control problems.

To overcome this limitation, as second case, the simulation extended to the centralized training and decentralized execution (CTDE) paradigm, enabling coordinated learning across multiple agents during training while preserving independent execution during inference, made the solution more scalable. The outcomes are promising by achieving 84% accuracy compared to heuristic optimal solution.

3.3 BEAM HOPPING

3.3.1 Beam hopping Overview

3.3.1.1 Introduction

NTN includes satellite constellations (LEO, MEO, GEO) and high-altitude platforms (HAPs), which are becoming integral components of beyond-5G and 6G systems. A central challenge in NTN lies in the efficient utilization of constrained spectrum and power resources to meet dynamic and geographically diverse user demands. Beam Hopping (BH) has emerged as a promising technique to improve flexibility, spectral efficiency, and capacity in such networks.

3.3.1.2 Concept of Beam Hopping

Beam hopping is a dynamic resource allocation approach where a satellite's multi-beam footprint is not continuously illuminated. Instead, beams are sequentially activated in time following a predetermined or adaptive hopping schedule. Unlike traditional fixed multi-beam systems, where all spot beams remain active simultaneously, BH systems allocate radio-frequency (RF) power and spectrum only to beams corresponding to active user demand.
Key characteristics include:

- **Temporal multiplexing** – Beams are activated in scheduled time slots.
- **Spatial adaptability** – Coverage is shifted according to traffic distribution.
- **Resource efficiency** – Power and spectrum are concentrated on active beams, thereby improving utilization.

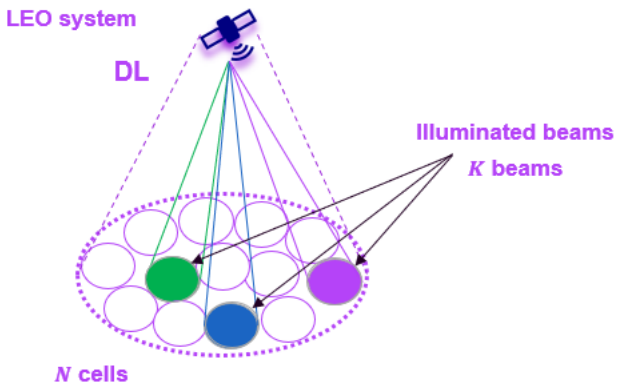

FIGURE 3-17 BEAM HOPPING SYSTEM SCENARIO

3.3.1.3 Role in NTN

Beam hopping directly addresses several inherent NTN challenges:

- **Traffic variability** – User demand across NTN regions (e.g., urban vs. oceanic) is highly uneven. BH enables dynamic adaptation to these spatio-temporal fluctuations.

- **Spectrum scarcity** – By enabling temporal reuse of spectrum across non-overlapping beams, BH significantly improves spectrum efficiency.
- **Energy efficiency** – Concentrating transmission power only on active beams reduces energy waste in low-demand regions.
- **QoS and latency support** – Hopping sequences can be designed to prioritize latency-sensitive or high-throughput services, facilitating differentiated QoS provisioning.

3.3.1.4 System Design Considerations

The design of beam hopping systems in NTN involves optimization across multiple dimensions:

- **Hopping pattern design** – Defines which beams are illuminated and when, using either static schedules or adaptive, traffic-aware strategies.
- **Frame structure alignment** – Hopping cycles must align with physical-layer frame timing to minimize signaling overhead.
- **Power and spectrum allocation** – Requires balancing throughput maximization with fairness across beams.
- **Gateway–satellite coordination** – Efficient feeder link design and backhaul management are essential for real-time hopping control.

3.3.1.5 Technical Challenges

Despite its benefits, BH introduces several technical complexities:

- **Synchronization** – Ground terminals must remain synchronized with intermittent beam illumination.
- **Handover management** – Frequent switching between beams can increase signaling load and complicate mobility management.
- **Interference control** – Overlapping hopping patterns across beams or satellites may lead to inter-beam interference.
- **Optimization complexity** – Real-time dynamic BH involves solving large-scale resource allocation problems under tight latency constraints.

3.3.1.6 Current Research Directions

Ongoing research is exploring novel enhancements to beam hopping for NTN, including:

- **AI/ML-driven BH optimization** – Using traffic prediction and reinforcement learning to dynamically adapt hopping patterns.
- **Joint beamforming and beam hopping** – Combining spatial beamforming with temporal hopping to further boost spectral efficiency.
- **Integration with terrestrial 6G** – Leveraging BH for seamless interoperability between NTN and terrestrial networks.
- **Standardization efforts** – Ongoing 3GPP work in Releases 17 and 18 positions BH as a critical feature of NR NTN architectures.

3.3.2 System model

3.3.2.1 Notations and Definitions

The table below presents the notations associated with the variables considered in the beam hopping scheme, along with their corresponding definitions.

TABLE 3-4 BEAM HOPPING: VARIABLE NOTATIONS AND DEFINITIONS

Notation	Definitions
$n \in \{1, \ldots, N\}$	Cells
$k \in \{1, \ldots, K\}$	Beams
N_u	Number of antennas at the user u
N_s	Number of antennas elements at the satellite
T_D	Duration of a time slot
$m \in \{1, \ldots, M\}$	Number of time slots for which a packet could be stocked in the satellite buffer, the packet will be discarded if it has been waiting in the buffer more than M time slots
Λ_t^n	The amount of arrival traffic in cell n at time t $\overline{\Lambda_t^n}$ for RT data, and $\overline{\overline{\Lambda_t^n}}$ for non-RT data
$d_{t,m}^n$	The amount of packet arrival in cell n at time t, that has been waiting in the buffer for m time slot. $\overline{d_{t,m}^n}$ for RT data, and $\overline{\overline{d_{t,m}^n}}$ for non-RT data
$\emptyset_t^n$	The amount of packet arrival in cell n at time t, $\emptyset_t^n = \sum_{m=1}^{M} d_{t,m}^n$ $\overline{\emptyset_t^n}$ for RT data, and $\overline{\overline{\emptyset_t^n}}$ for non-RT data
D_t^n	The amount of packet arrival and the amount of arrival traffic in cell n at time t . $D_t^n = \emptyset_t^n + \Lambda_t^n$ $\overline{D_t^n}$ for RT data, and $\overline{\overline{D_t^n}}$ for non-RT data

3.3.2.2 System dimension

We consider one LEO satellite covering N cells and able to illuminate up to K beams. Each satellite has 7 antennas, and each antenna can generate instantaneously 4 to 8 beams, so from 28 to 56 beams in total.

- Number of cells covered by the LEO satellite: 499
- Altitude of the satellite: 600 km
- Elevation: 45°
- Surface of the cell: 45 km
- Number of antennas at each satellite: 7 antennas
- Number of simultaneously generated beams by antenna: 4 to 8

3.3.2.3 Representation of the buffer data and delay values across cells

The table below presents the data stored in the buffer along with the corresponding delays across different cells.

TABLE 3-5 REPRESENTATION OF BUFFER DATA AND DELAY VALUES ACROSS CELLS

	Arrival traffic	Packet arrival waiting in the buffer				Total data to be served
Packet delay at t	**0**	T_D	…	$(M-1)\,T_D$	$M\,T_D$	Total stored data
Cell 1	Λ_t^1	$d_{t,1}^1$	…	$d_{t,(M-1)}^1$	$d_{t,M}^1$	$D_t^n = \Lambda_t^n + \sum_{m=1}^{M} d_{t,m}^n$
Cell 2	Λ_t^2	$d_{t,1}^2$	…	$d_{t,(M-1)}^2$	$d_{t,M}^2$	$D_t^n = \Lambda_t^n + \sum_{m=1}^{M} d_{t,m}^n$
⋮						
Cell N	Λ_t^N	$d_{t,1}^N$	…	$d_{t,(M-1)}^N$	$d_{t,M}^N$	$D_t^n = \Lambda_t^n + \sum_{m=1}^{M} d_{t,m}^n$

3.3.2.4 Bandwidth allocation scheme

- Total BW B_{tot} divided into Msub-band, each having equal BW of B_{sub}
  - Each beam can utilize one or more consecutive sub-bands.
  - There are $\frac{M(M+1)}{2}$ available allocation schemes.
  - The BW allocation for each slot t can be represented by:

$$BW_t = \left\{ \left(B_t^1, B_t^2, \ldots, B_t^k, \ldots, B_t^K\right) \middle| \, B_t^k = 1,2, \ \ldots, \ \frac{M(M+1)}{2} \ and \ k \in K \right\}$$

  - The notation $B_t^k = i$ signifies that the i^{th} BW allocation schema is applied to beam k during time slot t Beam occupancy.
  - Example of bandwidth scheme:
    For 4 Sub-bands, we can have 10 available bandwidth allocation schemes.

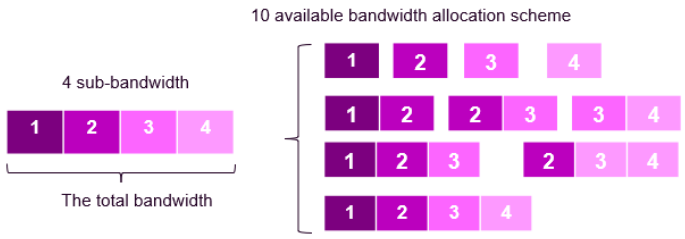

FIGURE 3-18 THE AVAILABLE BANDWIDTH ALLOCATION SCHEME FOR 4 SUB-BANDS

3.3.2.5 Beam occupancy

Let x_t^n be a variable representing the illumination status of a beam. When $x_t^n = 0$, it indicates that the cell n has not been illuminated at time t, while $x_t^n = 1$ signifies that the cell has been illuminated. So the beam occupancy representing all the cells can be expressed as:

$$X_t = \{(x_t^1, x_t^2, \dots, x_t^n, \dots, x_t^N) |\, x_t^n = 0{,}1 \; and \; \sum_{i=1}^{N} x_t^i = K\} \tag{3.3.1}$$

3.3.2.6 Channel Model

Here we consider the user link

- $H_k \in \mathbb{C}^{N_s \times N_u}$Matrix Channel gain from beam k to cell n
- The satellite forward link loss $H = [H_1 \dots H_K] \in \mathbb{C}^{N_s \times K\, N_u}$
- The channel coefficient from beam k to cell n: $h_{k,n} = G_k^{Tx}.P_L.G_n^{Rx}$
  - G_k^{Tx} / G_n^{Rx} Transmit/ receive beam antenna gain, P_L Pathloss

3.3.2.7 Co-channel interference

The overlap factor quantifies the co-channel interference from beam j to beam i : $\alpha_t^{i,j} = \frac{|B_t^i \cap B_t^j|}{|B_t^i|}$

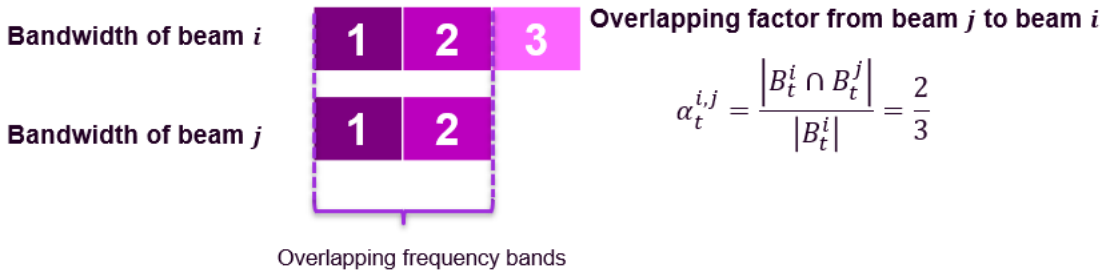

FIGURE 3-19 OVERLAP FACTOR FOR CO-CHANNEL INTERFERENCE.

3.3.2.8 SINR

- SINR of the user at cell n when illuminated by beam k

$$SINR_t^{n,k} = \frac{P_b \mathbb{E}[\|H_k s\|^2]}{N_0 |B_t^K| B_{sub} + \sum_{k' \in K, k' \neq k} P_b \alpha_t^{k,k'} \mathbb{E}[\|H_{k'} s\|^2]} \tag{3.3.2}$$

  - P_b The transmit power of each beam, N_0 the noise power spectral density, and B_T^K the number of sub-bands

3.3.2.9 Channel Capacity

- The channel capacity is represented by the following equation:

$$C_t^n = x_t^n |B_t^k| B_{sub} log_2(1 + SINR_t^{n,k}) \tag{3.3.3}$$

3.3.2.10 Performance metrics

- The non-RT data throughput in cell n at time t:

$$Th_t^n = min\{C_t^n, D_t^n\} \tag{3.3.4}$$

- The non-RT data throughput in all cells at time t:

$$Th_t = \sum_{n=1}^{N} Th_t^n \tag{3.3.5}$$

- The average delay of RT packet of cell n at time t:

$$\tau_t^n = \frac{\sum_{m=1}^{M} mT_D d_{m,t}^n}{\sum_{m=1}^{M} d_{m,t}^n} \tag{3.3.6}$$

3.3.3 Optimization problem

Based on the definitions of the aforementioned parameters, the beam hopping scheme can be formulated as an optimization problem, as expressed in the following equation,

$$opt. \max_{x_t^k, B_t^k} \wp = \beta \sum_{n=1}^{N} Th_t^n - (1-\beta) \sum_{n=1}^{N} \tau_t^n \tag{3.3.7}$$

$$C_1: \sum_{k=1}^{K} P_b^k < P_{tot}, \quad \forall k \in K$$

$$C_2: \sum_{k=1}^{K} P_b^k < P_{max}, \quad \forall k \in K$$

$$C_3: \left|B_t^k\right| B_{sub} < B_{tot}, \quad \forall k \in K$$

$$C_4: B_t^k = 1,2, \ \ldots, \ \frac{M(M+1)}{2}, \qquad \forall k \in K$$

$$C_5: \sum_{i=1}^{N} x_t^i = K \ and \ x_t^n = 0,1.$$

- The above objective function represents a compromise between maximizing the throughput of non-RT users and minimizing the latency of RT users.
- β is a **weighting parameter** that balances the influence of the throughput and latency in the model, constrained between 0 and 1.
- Constraint 1 corresponds to the limitation on the total available power at the satellite.
- Constraint 2 ensures that the sum of power allocated to individual users does not exceed the maximum allowable limit.
- Constraint 3 guarantees that the bandwidth assigned to users does not exceed the total available system bandwidth.
- Constraint 4 addresses the rules governing the bandwidth allocation schemes across users.

- Constraint 5 pertains to the occupancy constraints of the beams.

3.3.3.1 Beam Hopping Problem as Sequential Decision-Making

The data packet in the buffer at time t can be expressed by the following equation:

$$D_t^n = D_{t-1}^n - x_{t-1}^n \, C_{t-1}^n + \Lambda_t^n \quad (3.3.8)$$

- This indicates that the stored data packet at time t is equal to the stored data packet at time $t-1$, minus the amount served by the illuminated beams, plus the traffic arrival at time t.

3.3.4 Deep Reinforcement learning solution for Beam Hopping (Deep Q learning)

3.3.4.1 Advantages of Deep Q-Learning for Efficient Beam Hopping

Scalability

- Beam hopping entails a high-dimensional decision space, encompassing variables such as beam configurations, resource block allocation, traffic demand and type, as well as diverse QoS requirements.
- Traditional tabular Q-learning and exact optimization approaches are inadequate for managing this level of complexity and scalability.
- Deep Q-learning leverages neural networks to approximate the action-value function, enabling superior scalability and adaptability in complex environments.

Adaptivity to dynamic traffic

- User demand exhibits temporal variability, driven by periodic patterns (e.g., diurnal cycles) as well as unpredictable traffic surges.
- Heuristic approaches, such as greedy algorithms, lack adaptability, while optimization-based models require frequent re-computation to remain effective.
- Deep Q-learning is capable of learning adaptive policies that dynamically respond to variations in the observed system state.

Fast online decision-making

- Convex optimization techniques can, in principle, yield optimal resource allocations; however, their computational cost and time requirements render them impractical for real-time decision-making.
- In contrast, once trained, a Deep Q-learning agent can produce decisions efficiently through a single forward propagation step (i.e., matrix multiplications) within the neural network.

Long-term optimization

- Greedy and heuristic strategies typically prioritize maximizing instantaneous throughput.
- However, beam hopping necessitates a trade-off between current and future allocations to prevent service deprivation across users or regions.
- Deep Q-learning explicitly optimizes cumulative long-term rewards, thereby promoting both fairness and efficiency in resource allocation. Formally, the action-value function (Q-function) is defined as the expected discounted return:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} | s_t = s, a_t = a] \quad (3.3.9)$$

where $\gamma \epsilon [0,1)$ is the discount factor. The optimal policy then selects the action that maximizes this value:

$$\pi^{*}(s) = arg \max_{a} Q^{\pi}(s,a) \quad (3.3.10)$$

Flexibility for multi-objective goals

- Beam hopping requires a joint consideration of throughput, latency, fairness, QoS, and interference constraints.
- Within the Deep Q-learning framework, these diverse objectives can be systematically incorporated into the reward function.
- This enables simultaneous optimization of multiple key performance indicators (KPIs), offering greater flexibility compared to rigid heuristic strategies or single-objective optimization methods.

3.3.4.2 Problem Summary

- **Goal**: Dynamically select which K out of N cells to illuminate using satellite beams and allocate radio resources among them.
- **Inputs**:
  - Traffic demand in each cell (arrival rate, type: real-time or non-real-time)
  - Number of beams K
  - Total radio resource blocks available
- **Output**:
  - Selected K cells to serve (beam hopping decision)

- Resource allocation per beam

- **Reward**:
  - Tradeoff of:
    - Minimizing latency for real-time traffic
    - Maximizing throughput for non-real-time traffic

3.3.4.3 Deep Q-Learning Design

State: represents the amount of the data packet and arrival traffic at the buffer waiting to be served.

$$S_t = D_t^n = D_{t-1}^n - x_{t-1}^n \, C_{t-1}^n + \Lambda_t^n \tag{3.3.11}$$

Action: represents the chosen beam occupancy and bandwidth allocation scheme

$$(X_t, B_t) \tag{3.3.12}$$

Reward: tradeoff between the non-RT data throughput and the RT data delay

$$\begin{aligned} R_t &= w_{th}.Th_t + w_l.(1 - \tau_t^n) \\ with: &\quad w_{th} + w_{th} = 1 \\ and: &\quad w_{th}, w_{th} \geq 0 \end{aligned} \tag{3.3.13}$$

3.3.4.4 Flowchart of beam hopping solution with DQL

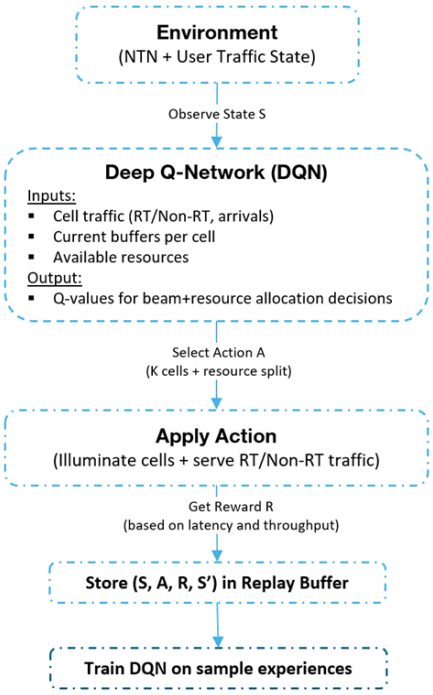

FIGURE 3-20 FLOW CHART OF BEAM HOPPING OPTIMIZATION AS A DQL ALGORITHM

3.4 LINK QUALITY PREDICTION

3.4.1 Free Space Path Loss Simulation

3.4.1.1 Motivation

In cellular NTN, as the service is provided by satellites orbiting the earth, it is common for the service coverage of a fixed location on earth surface to switch from one satellite to another on a regular basis. For example, if the NTN service is provided by LEO satellites, the switch can happen every few minutes. From a UE's perspective, even in a stationary scenario, constant measuring of service and neighbor satellite link quality is required to coordinate the service switch to avoid the loss of service.
Traditionally, link quality measurement on UE is done through collecting wireless signal samples and then computing link quality metric, i.e. Reference Signal Received Power (RSRP). Collecting neighbor satellite signal samples requires UE to tune its RF operation to neighbor satellite frequency, which is typically different from the service satellite frequency due to large Doppler shift caused by the satellite movement. Tuning RF frequency away from the service satellite frequency creates a service gap, which leads to lower data rate. Constant neighbor satellite measurement can also lead to higher power consumption, as UE needs to enable RF operation constantly, even during a long period of time with no data transmission.
Predicting neighbor satellite's link quality can be a way for UE to reduce the number of measurements required to maintain the service, therefore, increasing the data rate and lowering power consumption from operating RF. This research will focus on predicting satellite signal's Free Space Path Loss (FSPL), a dominating factor of satellite signal's RSRP, at an arbitrary time in the future, given the satellite's current position and velocity. We also assume stationary UE with a known position.

3.4.1.2 Dataset Generation

The dataset used for training, testing, and validating the FSPL prediction model consists of LEO satellite orbits generated by a satellite simulator. These orbits include 600km and 1200km altitude with five- and fourteen-minutes satellite visibility, respectively. In addition, different inclinations have been considered. Each orbit is represented by a series of Kepler elements sampled every four seconds within the visible duration of the orbit. The Kepler elements are then converted to position and velocity of the satellite in ECEF frame. Finally, each satellite orbit is associated with a unique UE position, which is the projection of the satellite zenith position on the Earth surface, assuming a stationary UE scenario. The UE is assumed stationary in this study.

To increase the amount of data per orbit, the satellite's position and velocity are propagated with a fixed step size between two Kepler elements, resulting to T, a natural number, position and velocity states for an orbit. With satellite and UE position known, FSPL can be calculated. Eventually, satellite orbit in the dataset consists of a series of satellite position and velocity, a UE position, and a series of FSPL.

For satellite orbit n, all satellite position states is written as $P_{sat,n} = [Px_{sat,n}, Py_{sat,n}, Pz_{sat,n}]_{Tx3}$, where $Px_{sat,n}$, $Py_{sat,n}$, $Pz_{sat,n}$ are length T vectors whose elements are values (in meter) on x, y, and z axis of ECEF frame. For integer t and $0 \leq t < T$, $P_{sat,n}[t] = \left[Px_{sat,n}[t], Py_{sat,n}[t], Pz_{sat,n}[t]\right]_{1x3}$ represents the satellite's position state at time instance t in the orbit.

Similarly, the velocity states of orbit n can be written as $V_{sat,n} = [Vx_{sa,n}t, Vy_{sat,n}, Vz_{sat,n}]_{Tx3}$, where $Vx_{sat,n}$, $Vy_{sat,n}$, $Vz_{sat,n}$ are length T vectors whose elements are velocity (m/s) along x, y, and z axis of ECEF frame. For time instance t and $0 \leq t < T$, the satellite's velocity state can be written as $V_{sat,n}[t] = \left[Vx_{sat,n}[t], Vy_{sat,n}[t], Vz_{sat,n}[t]\right]_{1x3}$.

UE position associated with satellite orbit n is represented by $p_{ue,n} = [px_{ue,n}, py_{ue,n}, pz_{ue,n}]_{1x3}$, where $px_{ue,n}$, $py_{ue,n}$, $pz_{ue,n}$ (in meter) are corresponding to values on x, y, and z axis of ECEF frame respectively.

Assuming both UE and satellite use isotropic antennas with 0dB antenna gain, FSPL in dB at time instance t of satellite orbit n can be calculated by the following equation:

$$f_{FSPL}(P_{sat,n}[t], p_{ue,n}) = 20log_{10}(d) + 20log_{10}(freq) + 20log_{10}\left(\frac{4\pi}{c}\right) \quad (3.4.1)$$

where d is the Euclidean distance (in meter) between $P_{sat,n}[t]$ and $p_{ue,n}$, and $c = 3x10^8\ m/s$ is the speed of the light. Therefore, for satellite orbit n, FSPL of all time instances can be written as a length T vector L_n, and $L_n[t] = f_{FSPL}(P_{sat,n}[t], p_{ue,n})$.
To increase the data variety for a satellite orbit, randomizing UE position is also considered. For satellite orbit n, a randomized UE position can be generated by sampling positions centered around the UE position $p_{ue,n}$ within radius D ($D \in N$), and M such randomized UE positions, $p_{ue,n,m}$, are generated:

$$p_{ue,n,m} = [px_{ue,n,m}, py_{ue,n,m}, pz_{ue,n,m}]_{1x3} \quad (3.4.2)$$

where $m = 1, 2 \ldots M$ and the Euclidean distance $\|p_{ue,n} - p_{ue,n,m}\| < D$. The FSPL associated $p_{ue,n,m}$ can be written in a length T vector $L_{n,m}$, where $L_{n,m}[t] = f_{FSPL}(P_{sat,n}[t], p_{ue,n,m})$. After applying UE position randomization, each satellite orbit has $M + 1$ sets of data, consisting of satellite position and velocity states, a unique UE position, and associated FSPL vector:

$$\{P_{sat,n}, V_{sat,n}, p_{ue,n,m}, L_{n,m}\} \quad (3.4.3)$$

where $m = 0,1 \ldots M$, $p_{ue,n,0} = p_{ue,n}$, and $L_{n,0} = L_n$.

3.4.1.3 Data Preprocessing

The following procedure is used to create input and output data for the prediction model:

1. Set values for D and M, select training orbits,
2. Keep the other orbits for validation and set $M = 0$
3. For each orbit, apply UE position randomization and generate all $M + 1$ sets of data

4. For all sets of data, including training and validation data set, apply the following terrestrial frames transformation and elemental rotations:
   a. Transform the relative position $P_{rel,n,m}$ and velocity vector $V_{sat,n}$ from ECEF to East-North-Up (ENU) frame with transform matrix R. Define:

$$P_{rel,n,m} = \left[Px_{rel,n,m}, Py_{rel,n,m}, Pz_{rel,n,m}\right]_{T\times 3} \quad (3.4.4)$$

where $Px_{rel,n,m}, Py_{rel,n,m}, \ and \ Pz_{rel,n,m}$ are length T vectors and $P_{rel,n,m}[t] = P_{sat,n}[t] - P_{ue,n,m}$.

$$R = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ \cos\phi\cos\lambda & \cos\phi\sin\lambda & \sin\phi \end{bmatrix} \quad (3.4.5)$$

where ϕ and λ and are the geodetic latitude and longitude of UE position $P_{ue,n,m}$.

The relative position in ENU frame at time instance t can then be written as,

$$P'_{rel,n,m}[t] = \left(R \times P_{rel,n,m}[t]^T\right)^T = \left[Pe_{rel,n,m}[t], Pn_{rel,n,m}[t], Pu_{rel,n,m}[t]\right]_{1\times 3} \quad (3.4.6)$$

where $P'_{rel,n,m}$ is a $T \times 3$ matrix, and $Pe_{rel,n,m}, Pn_{rel,n,m}, and \ Pu_{rel,n,m}$ are length T vectors whose elements are values (in meter) on axis E, N, and U of ENU frame.
The velocity in ENU frame at time instance t can then be written as,

$$V'_{sat,n}[t] = \left(R \times V_{sat,n}[t]^T\right)^T = \left[Ve_{sat,n}[t], Vn_{sat,n}[t], Vu_{sat,n}[t]\right]_{1\times 3} \quad (3.4.7)$$

where $V'_{sat,n}$ is a $T \times 3$ matrix, and $Ve_{sat,n}, Vn_{sat,n}, and \ Vu_{sat,n}[t]$ are length T vectors whose elements are velocity (m/s) along E, N, and U axis of ENU frame.

   b. Apply elemental rotation to $P'_{rel,n,m}$ and $V'_{sat,n}$ around E axis so that the post rotation relative position $P''_{rel,n,m}$ on N axis is 0 at the mid-point of the orbit:

$$P''_{rel,n,m}[t] = \left(R_E \times P'_{rel,n,m}[t]^T\right)^T = \left[Pe'_{rel,n,m}[t], Pn'_{rel,n,m}[t], Pu'_{rel,n,m}[t]\right]_{1\times 3} \quad (3.4.8)$$

$$V''_{sat,n}[t] = \left(R_E \times V'_{sat,n}[t]^T\right)^T = \left[Ve'_{sat,n}[t], Vn'_{sat,n}[t], Vu'_{sat,n}[t]\right]_{1\times 3} \quad (3.4.9)$$

and R_E is the rotation matrix,

$$R_E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_E & -\sin\theta_E \\ 0 & \sin\theta_E & \cos\theta_E \end{bmatrix} \quad (3.4.10)$$

$$\theta_E = arccos\left(\frac{u_{mid}}{\sqrt{n_{mid}^2+u_{mid}^2}}\right) \quad (3.4.11)$$

$$n_{mid} = Pn_{rel,n,m}[\lfloor T/2 \rfloor] \quad (3.4.12)$$

$$u_{mid} = Pu_{rel,n,m}[\lfloor T/2 \rfloor] \quad (3.4.13)$$

such that $P''_{rel,n,m}[\lfloor T/2 \rfloor] = \left[Pe'_{rel,n,m}[\lfloor T/2 \rfloor], 0, Pu'_{rel,n,m}[\lfloor T/2 \rfloor]\right]_{1\times3}$.

c. Apply elemental rotation to $P''_{rel,n,m}$ and $V''_{sat,n}$ around U axis so that the post rotation relative position $P'''_{rel,n,m}$ on E axis is 0 at the last position of the orbit:

$$P'''_{rel,n,m}[t] = \left(R_U \times P''_{rel,n,m}[t]^T\right)^T = \left[Pe''_{rel,n,m}[t], Pn''_{rel,n,m}[t], Pu''_{rel,n,m}[t]\right]_{1\times3} \quad (3.4.14)$$

$$V'''_{sat,n}[t] = \left(R_U \times V''_{sat,n}[t]^T\right)^T = \left[Ve''_{sat,n}[t], Vn''_{sat,n}[t], Vu''_{sat,n}[t]\right]_{1\times3} \quad (3.4.15)$$

, and is the R_U rotation matrix

$$R_U = \begin{bmatrix} \cos\theta_U & -\sin\theta_U & 0 \\ \sin\theta_U & \cos\theta_U & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.4.16)$$

$$\theta_U = arccos\left(\frac{n_{last}}{\sqrt{e_{last}^2+n_{last}^2}}\right) \quad (3.4.17)$$

$$e_{last} = Pe'_{rel,n,m}[T-1] \quad (3.4.18)$$

$$n_{last} = Pn'_{rel,n,m}[T-1] \quad (3.4.19)$$

such that $P'''_{rel,n,m}[T-1] = \left[0, Pn''_{rel,n,m}[T-1], Pu''_{rel,n,m}[T-1]\right]_{1\times3}$.

d. Since the transformation and elemental rotations do not alter the Euclidean distance between the satellite and UE at any given time t, the set of data for satellite orbit n becomes $\{P'''_{rel,n,m}, V'''_{sat,n}, L_{n,m}\}$

5. Scale all position, velocity and FSPL values to $[0,1]$ based on the maximum and minimum of $P'''_{rel,n,m}$, $V''_{sat,n}$, and $L_{n,m}$ for all n and m.
6. For UE position m in of a satellite orbit n at time instance t and a random time duration Δt, the input data of the prediction model is $\left[P'''_{rel,n,m}[t], V'''_{sat,n}[t], \Delta t\right]_{1\times7}$, and the output data of the model is $L_{n,m}[t+\Delta t]$

7. 2 orbits are selected for training. For each orbit and associated UE position m, 100 input and output pairs are randomly generated. This leads to 3100 per training orbit or 9300 training samples.
8. All orbits are used for testing. For each orbit, use the UE position projected from satellite zenith position and randomly select 30% of the time instance to generate input and expected output pairs, resulting to around 10000 testing samples.

3.4.1.4 Prediction Model

As shown below, the prediction model architecture is Deep Neural Network (DNN) with ReLu activation between layers:

FIGURE 3-21 DEEP NEURAL NETWORK (DNN) ARCHITECTURE AS PREDICTION MODEL

To train the DNN model, training criterion is the MSE loss, and Adam optimizer is used with learning rate=0.001, and 100 epochs.

For testing, a predicted FSPL is generated after inputting each test sample to the model. The prediction error of the test sample is then calculated as the absolute value of the difference between predicted FSPL and the expected output.

3.4.1.5 Prediction Result

There are 9 datasets created from 9 orbits. Dataset-3 and dataset-7 are created from orbits selected for training, and the remaining datasets are used for testing the prediction model's performance. Histogram of the prediction error is used to evaluate the performance of the prediction model. All histograms' x axis uses prediction error (dB). As shown below. The prediction model performs well as most prediction errors are below 0.5dB.

Following is the result with $D = 500km$, $M = 30$, and $max(\Delta t) = 180$ seconds:

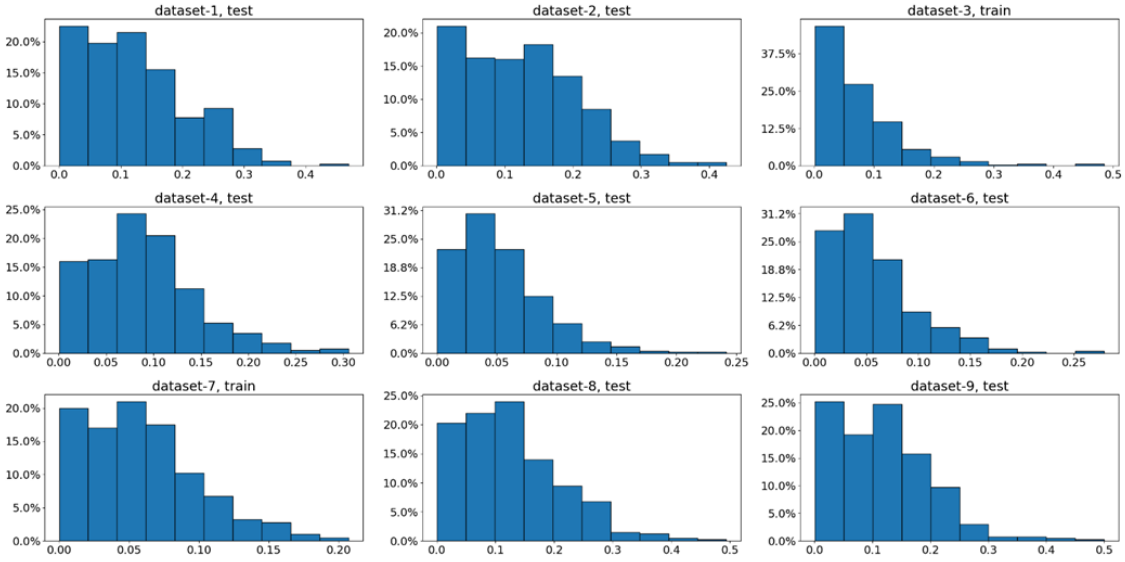

FIGURE 3-22 PREDICTION RESULTS FOR D=500KM, M=30, AND MAX(ΔT)=180 SECONDS:

Following is the result with $D = 500km$, $M = 30$, and $max(\Delta t) = 300$ seconds:

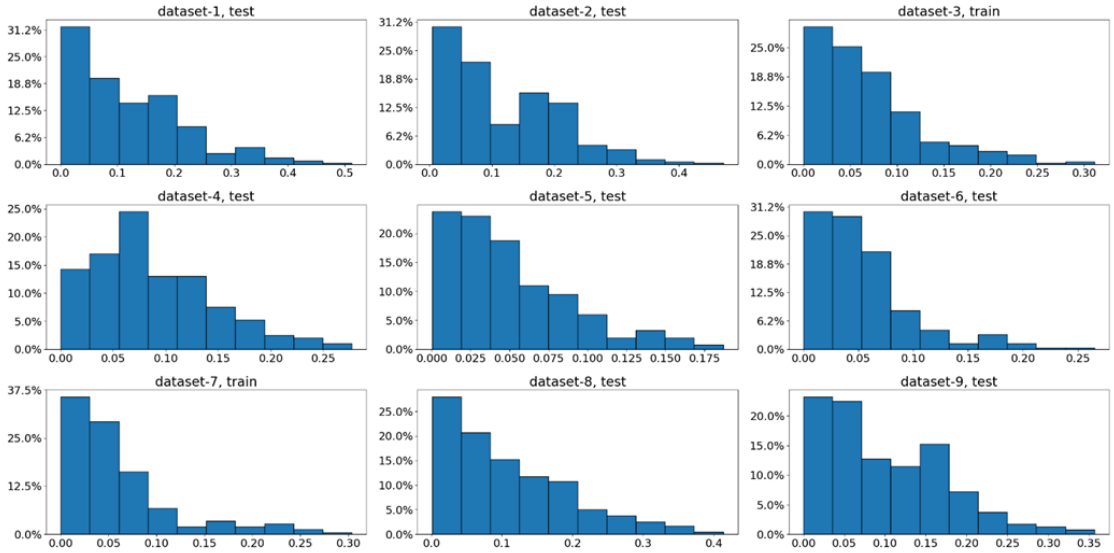

FIGURE 3-23 PREDICTION RESULTS FOR D=500KM, M=30, AND MAX(ΔT)=300 SECONDS:

3.4.2 NTN Ray-tracing Simulation

In order to test the pathloss prediction with a more realistic channel characterization, we built an internal NTN ray-tracing simulator on top of a third-party tool (i.e., MATLAB).

The standard MATLAB ray-tracing tool adopts the shooting and bouncing rays (SBR) method, which supports up to 100 path reflections and two edge diffractions, and takes into account the 3D geometry and surfaces characteristics to determine the path loss and phase shift of each ray (including tracing the horizontal and vertical polarizations of a signal through the propagation path).
However, it presents key limitations, such as a maximum distance between Tx and Rx of 500 km and a maximum angular separation of 0.05 degrees, which often does not give a multipath for satellite altitudes greater than 400 km (see **Figure 3-24**). Moreover, it is not possible to increase the ray density in a certain direction and exchanging the role of Tx and Rx does not improve the number of components.

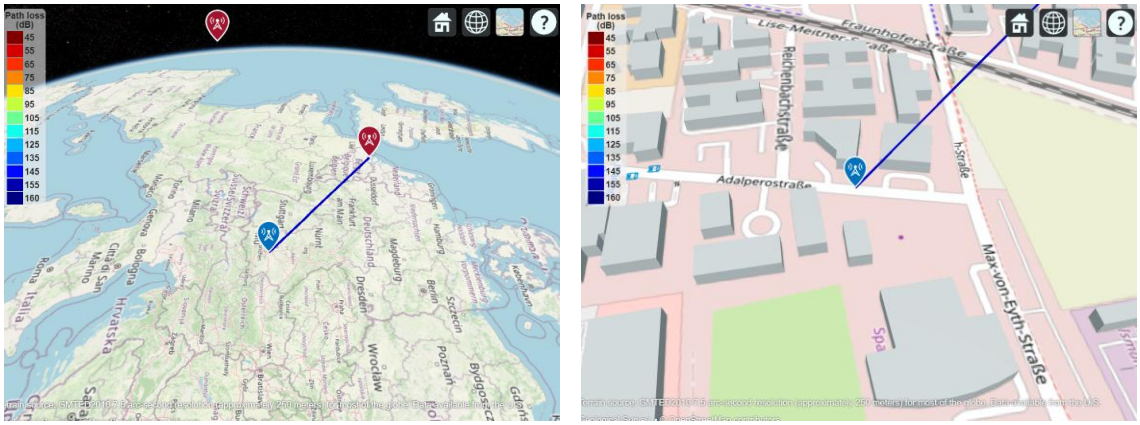

FIGURE 3-24 EXAMPLE OF DEFAULT MATLAB RAY-TRACING TOOL THAT DOES NOT PRODUCE MULTIPATHS

To solve this problem, we decided to split the ray-tracing into two parts: the first one from the Tx (i.e., satellite) to a virtual site (in the line of sight (LoS) link between Tx and Rx) and the second one from the virtual site to the Rx (i.e., UE). The first path (Sat-virtualSite) is computed by considering a free-space pathloss propagation, whereas the second path (virtualSite-UE) is computed using the full ray-tracing tool (see **Figure 3-25** for an example of NTN ray-tracing propagation). Note that the second path is much shorter to permit the computation of the multipath. Finally, the rays are combined by first separating the contributions of reflections/diffractions from free-space propagation in the second-path rays, and then by recomputing the free-space propagation in the overall path composed of first and second-paths.

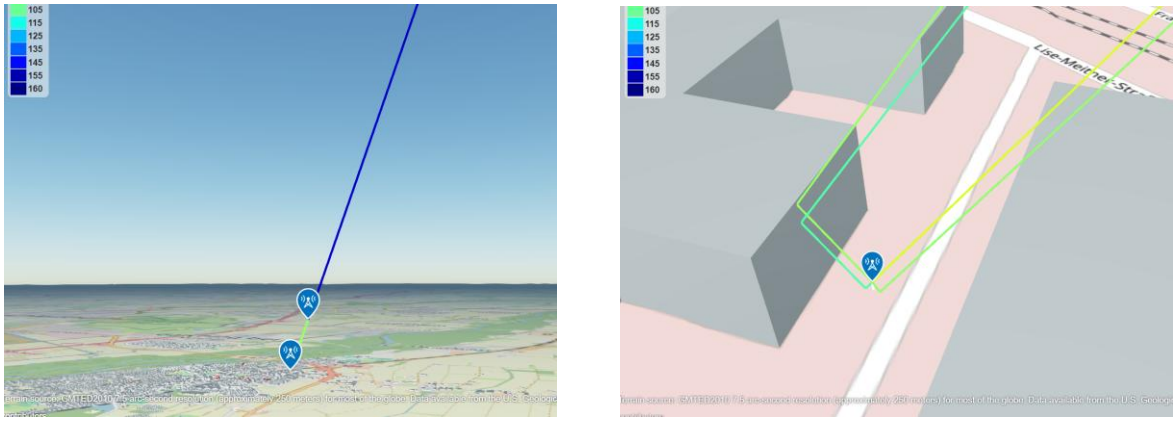

FIGURE 3-25 EXAMPLE OF CUSTOM NTN RAY-TRACING TOOL THAT DOES PRODUCE MULTIPATHS

The output of the NTN ray-tracing tool, concerning the task of pathloss prediction, is the power delay profile, composed of pathloss and path delays for each path of the channel. In case the signal is not received at the UE, we set the pathloss as 300 dB, signifying the complete lack of signal. Indeed, the buildings in the environment around the UE create particular patterns in the power delay profile that can be learned by a ML model to predict the future pathloss in time. An example can be seen in **Figure 3-26**, where out of 5 paths, only 2 are in LoS. However, among the 3 in non-line of sight (NLoS), the highlighted blue path is not received at all, whereas the highlighted pink one is still received by means of a back reflection.

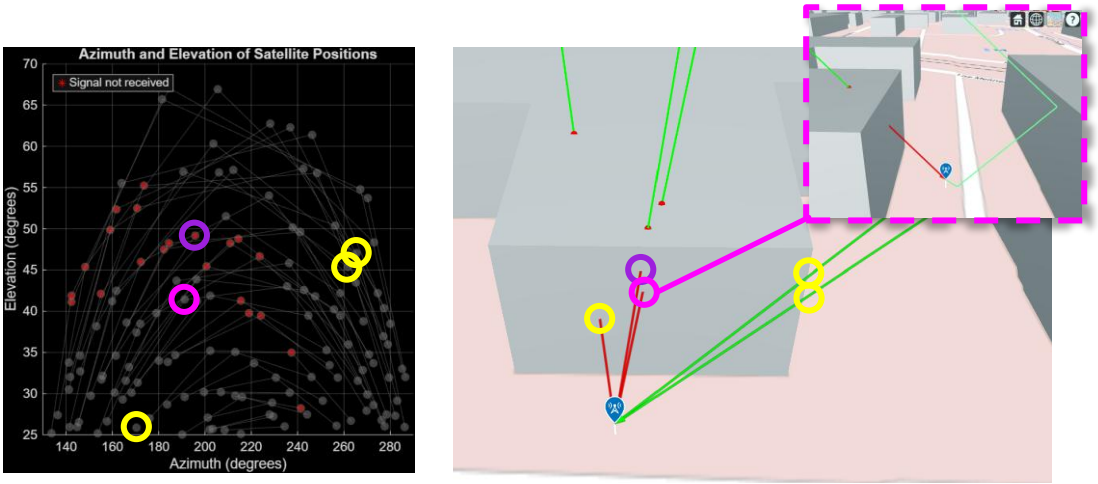

FIGURE 3-26 EXAMPLE OF MULTIPATH RECEIVED IN A SINGLE POSITION BY THE UE, WITH HIGHLIGHTED LOS (GREEN) AND NLOS (RED) PATHS. ON THE LEFT, THE CORRESPONDING ELEVATION AND AZIMUTH ANGLES ARE SHOWN

Regarding the specific dataset adopted for pathloss prediction, we created a single and a multi-UE dataset. In both cases, the UEs are static, and we adopted two constellations:

- Starlink constellation comprising 403 satellites, with 25 degree of minimum user elevation
- WP3-proposed constellation comprising 1269 satellites, with 45 degree of minimum user elevation

In both cases, we adopted a timestamp interval duration of 1 second and a carrier frequency of 1.5 MHz (ARFCN 1542, 308400). We point out that the carrier frequency does not influence the rays, as they are geometrically computed by the developed NTN-raytracing tool. Thus, the performances on pathloss prediction can be generalized to other carrier frequencies, e.g., C and Q/V bands. The transmitted power, as well as the antenna array pattern (i.e., isotropic antenna), are not considered since the ray-tracer computes the channel characteristics entirely based on the environment, and not on estimations from communication signals. Therefore, the received power at the UE could be eventually reconstructed from the predicted pathloss and the deterministic transmitted power and antenna array pattern.

The scenario is an urban environment in the area of Munich, where the 3D building geometries are extracted from OpenStreetMap. In **Figure 3-27**, we report the 10 multi-UE positions in the urban environment. The single-UE case is the position Pos1. In the single-UE case, we gather samples for 1591 timestamps (about 27 minutes) and 2700 timestamps (about 45 minutes), for the Starlink and WP3 constellation, respectively. In each timestamp the power delay profile with all the satellite in view is collected. The number of simultaneous satellites in view may vary significantly among the two constellations because of different minimum elevation angle. In the multi-UE case, for each UE, we gathered samples for 301 timestamps (about 5 minutes) and 1500 timestamps (about 25 minutes), for the Starlink and WP3 constellation, respectively.

FIGURE 3-27 MULTI-UE SCENARIO IN THE AREA OF MUNICH. THE 10 UE POSITIONS HAVE BEEN CHOSEN TO REPRESENT DIVERSE CONDITIONS AND ENVIRONMENT CHARACTERISTICS

3.4.2.1 NTN Ray-tracing Single-UE Dataset

In the two single-UE datasets created from WP3 and Starlink constellation, there are 50 and 42 satellites respectively. For convenience, the datasets are referred to as WP3 constellation and Starlink constellation in this and the next section. For a satellite ID n ($n \in \{1,2,\dots,N\}$, $N = 50$ for WP3 and $N = 42$ for Starlink constellation), its visible time from the UE's perspective is T_n seconds. The ray-tracing simulator samples along the trajectory of satellite n every second, and each sample contains various types of data, including the satellite's position and velocity in ECEF frame, elevation, Azimuth, propagation delay, line of sight status, and pathloss (dB). Therefore, the number of samples collected from satellite n is equivalent to T_n. There are 8547 and 5664 such samples for WP3 and Starlink constellation respectively. Notice, propagation delay, line of sight status, and pathloss are represented as row vectors with length equal to the number of multipath received by the UE at the sampled time instance. For example, the pathloss of a sample from satellite n to UE at time t can be presented as:

$$pathloss_n[t] = [pl_1, pl_2, \dots, pl_l]_{1xl} \tag{3.4.20}$$

where $0 \le t \le T_n - 1$, l is the multipath number at sample time t, and $pl_1, pl_2, \dots, pl_l$ are pathloss (in dB) associated with path 1 to l.

To simplify the pathloss prediction task, we choose to predict the pathloss of the strongest signal path of a sample and represent such pathloss from satellite n to UE as,

$$PL_n[t] = min(pathloss_n[t]) \tag{3.4.21}$$

where $0 \le t \le T_n - 1$, and PL_n is a row vector with length T_n. For convenience, we will refer pathloss of the strongest signal path as pathloss in the rest of the paper.

To facilitate the analysis of the dataset and the pathloss prediction, we further classify every sample according to its line-of-sight status to LOS, NLOS, and NS (no signal). The difference between a LOS and an NLOS sample is that UE receiving the direct NTN signal in the further and reflected or diffracted signal in the later. Following **Figure 3-28** shows the distribution of signal status in the two datasets:

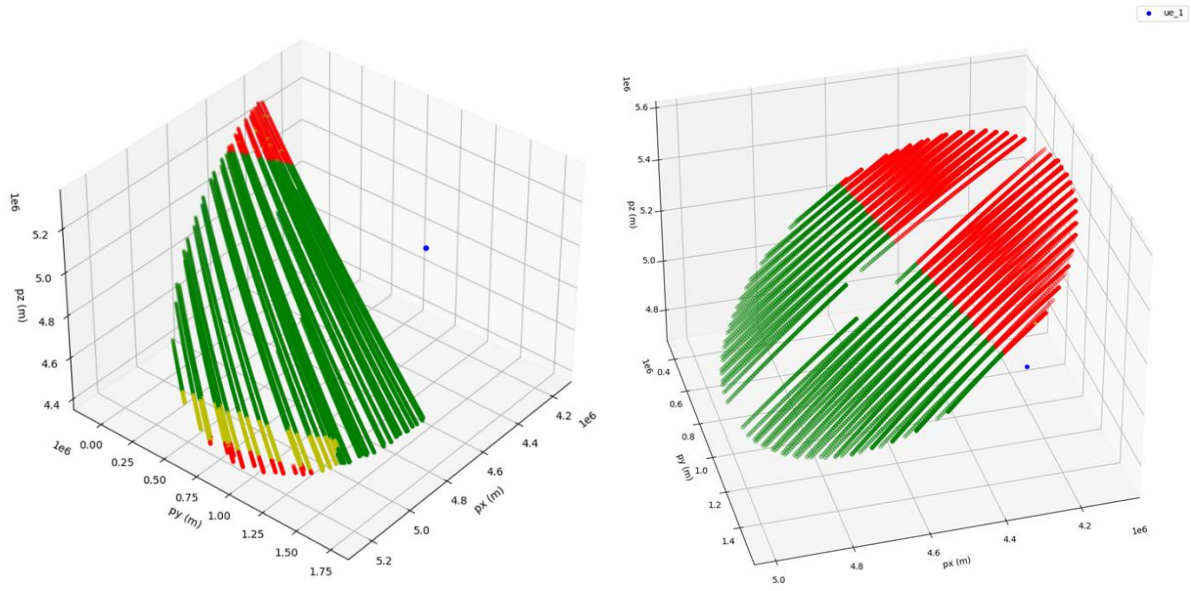

FIGURE 3-28 SIGNAL STATUS DISTRIBUTION IN ECEF FRAME. EACH SAMPLE IS PLOTTED ACCORDING TO THE SATELLITE POSITION AND COLOR CODED ACCORDING TO ITS SIGNAL STATUS: LOS (GREEN), NLOS (YELLOW), AND NS (RED). THE BLUE DOT IS UE POSITION. LEFT: STARLINK CONSTELLATION. RIG

In the Starlink constellation, there are 86.6% of the samples marked as LOS, 6% of NLOS, and 7.3% of NS. For samples classified as LOS, their pathloss range from 151dB to 157dB. Pathloss of NLOS samples range from 162 to 165dB. In the WP3 constellation, 58.9% of the samples are marked as LOS with pathloss ranging from 151 to 155dB. There is only one NLOS sample with 195dB pathloss, and the remaining 41.1% samples are marked as NS. See the pathloss histograms in **Figure 3-29** below,

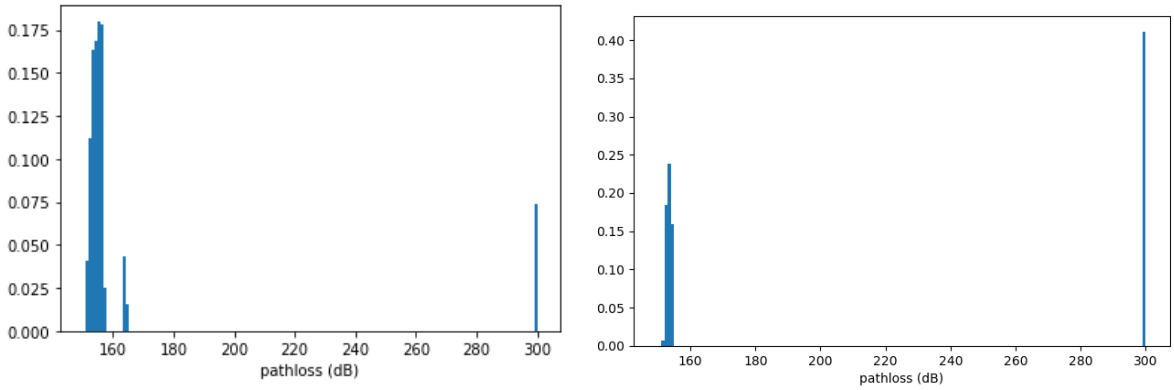

FIGURE 3-29 PATHLOSS HISTOGRAM OF SINGLE-UE DATASET. THE Y AXIS IS THE DENSITY. ALL PATHLOSS OF NS SAMPLES ARE SET TO 300DB. LEFT: STARLINK CONSTELLATION. RIGHT: WP3 CONSTELLATION.

We formulate satellite position (in meter), velocity (meter per millisecond), and pathloss (in dB) associated with satellite n in the single-UE dataset below:

$$P_n[t] = \left[Px_n[t], Py_n[t], Pz_n[t]\right]_{1x3} \quad (3.4.22)$$

$$V_n[t] = \left[Vx_n[t], Vy_n[t], Vz_n[t]\right]_{1x3} \quad (3.4.23)$$

where $0 \leq t \leq T_n - 1$. Px_n, Py_n, Pz_n, Vx_n, Vy_n, and Vz_n are all row vectors with length T_n. UE position (in meter) in the dataset is represented as

$$p = [px, py, pz]_{1x3} \quad (3.4.24)$$

where px, py, and pz are all real numbers.

Signal status of a sample from satellite n at time t as $SS_n[t] \in \{LOS, NLOS, NS\}$.

3.4.2.2 Pathloss Prediction on Single-UE Dataset

We formulate the pathloss prediction task like the FSPL prediction – given satellite n and UE data at time t, predict the pathloss (of the strongest signal path) from satellite n at time $t + \Delta t$, and $1 \leq \Delta t \leq 120$ (seconds). We set the Δt upper bound to 120 seconds which is about half of the largest satellite visible time in Starlink and WP3 constellation.
As shown in Figure 6, the pathlosses are separated into three groups in Starlink constellation and two groups in WP3 constellation. From previous analysis, we know these groups are

samples classified as either LOS, NLOS, or NS. Therefore, it might be beneficial for the pathloss prediction model to first perform a classification and then a regression task. We experimented with two model architectures, as shown in **Figure 3-30**, to perform such tasks.

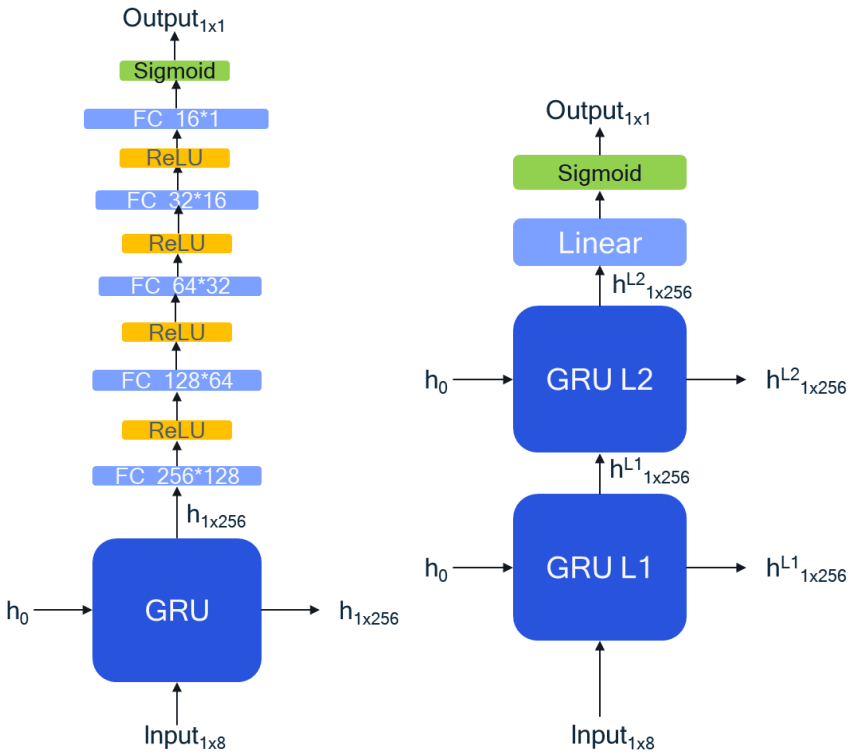

FIGURE 3-30 LEFT: MODEL#1, RIGHT: MODEL#2.

The first model is a single layer Gated Recurrent Unit (GRU) with hidden size 256 followed by five layers of DNN with ReLU activation in between and a sigmoid output layer. The second model is a two-layer GRU with hidden size 256 followed by a liner layer and a sigmoid output layer. h_0 is the initial state of the GRUs and is a zero vector. h, h^{L1}, and h^{L2} are hidden states of the GRUs. Input data for both models include the followings extracted from satellite n and UE data at time t: relative positions between satellite n and UE, velocities of satellite n, and pathloss. In addition, Δt is also included in input. Specifically:

$$input(n,t,\Delta t) = \{P_n[t] - p, V_n[t], PL_n[t], \Delta t\} \quad (3.4.24)$$

$$= \left[Px_{rel,n}[t], Py_{rel,n}[t], Pz_{rel,n}[t], Vx_n[t], Vy_n[t], Vz_n[t], PL_n[t], \Delta t\right]_{1\times 8}$$

where relative positions $Px_{rel,n}[t], Py_{rel,n}[t]$, and $Pz_{rel,n}[t]$ are $Px_n[t] - px, Py_n[t] - py$, and $Pz_n[t] - pz$, respectively.

Both models predict the pathloss at $t + \Delta t$, so the expected output is the true pathloss at time $t + \Delta t$, or $PL_n[t + \Delta t]$.
The prediction task can be considered as a function of input data that depends on n, t, and Δt. The output of the function is an estimated $PL_n[t + \Delta t]$, or $\widetilde{PL}(n,t,\Delta t)$:

$$f\big(input(n,t,\Delta t)\big) = \widetilde{PL}(n,t,\Delta t) \quad (3.4.25)$$

Therefore, the prediction error can be written as $\widetilde{PL}(n,t,\Delta t) - PL_n[t+\Delta t]$.
For preprocessing, all positions and velocities in a constellation are scaled separately by their axis with standard scalar, i.e. all x axis positions (px and all elements of Px_n, $n \in \{1,2,\dots,N\}$) are scaled together with one standard scalar. For pathloss, we first reclassify all NLOS samples with pathloss 180dB or larger to NS and then set pathloss of all NS samples to 180dB. This separates the pathlosses of three signal status groups in Starlink constellation with similar distances. We then scale the pathlosses with a min-max scalar to preserve the relative relationships among pathlosses of the signal status groups.
The two model architectures are each trained using dataset generated from Starlink and WP3 constellation separately, resulting in two prediction models (one for Starlink and the other for WP3) each architecture. For each constellation, we would like to create a training dataset that includes every sample in input and expected output data. Therefore, we iterate through every sample in every satellite in the constellation and repeat the following to collect training dataset entries from a sample of satellite n at time t:

1. Generate Δt randomly, $0 < \Delta t \leq 120$
2. Collect input data from sample at $t - \Delta t$ and expected output data from sample at t

There are more than 210000 entries collected for each training dataset. The dataset is further split randomly so that 80% of the entries are used for training and 20% for testing.

For each trained model, another dataset is created to validate the model. We iterate through all satellites in a constellation and generate entries randomly. Specifically, the following procedure is repeated numerous times for collecting entries from satellite n:

1. Randomly pick a current time c within satellite visible time T_n
2. Randomly pick a Δt, $1 \leq \Delta t \leq min\,(120,\ T_n - c)$
3. If an entry created from c and Δt already exists in validation dataset, go to step 1
4. Collect input data from sample at c and expected output data from sample at $c + t$

More than 19000 entries were collected for each validation dataset with this method.
Training and testing procedures for models of the two architectures are identical. Mean square error is the loss criterion, and the Adam optimizer with learning rate 0.0001 is selected. The training is performed up to 200 epoch with early stop mechanism. In each epoch, training and test loss are monitored, and test loss is used to determine early stop and select the best trained model. Once the training is finished, the performance of the best model is evaluated against the validation dataset.

We use the percentage of absolute prediction errors smaller than 0.5dB as the prediction accuracy to evaluate the performance of the prediction models. The best model selected in training procedure is evaluated against both testing and validation dataset. In addition to evaluating all predictions together, we also grouped predictions by the signal status associated with the sample of the true pathloss and then evaluated the prediction accuracy of each group. For comparison, the prediction accuracy of a naïve prediction method that always predicts future pathloss as the current pathloss regardless of Δt is included. Following table shows the performance of model#1 and model#2 on Starlink and WP3 constellation:

TABLE 3-6 PATHLOSS PREDICTION ACCURACY (%) ON STARLINK CONSTELLATION.

	Model#1 Testing	Model#1 Validation	Model#2 Testing	Model#2 Validation	Naive
Overall	93.9	91.8	93.3	90.5	27.9
Truth LOS	97	94.6	97	94.4	30.8
Truth NLOS	79.7	83.3	77	75.3	4.1
Truth NS	76	75.6	69.5	80	52.5

TABLE 3-7 PATHLOSS PREDICTION ACCURACY (%) ON WP3 CONSTELLATION. THERE IS NO TRUTH NLOS ROW DUE TO NO SAMPLE MARKED AS NLOS AFTER PREPROCESSING.

	Model#1 Testing	Model#1 Validation	Model#2 Testing	Model#2 Validation	Naive
Overall	98.2	98.5	68.3	68.8	26.6
Truth LOS	97.9	98.1	54.9	49.2	24.8
Truth NS	98.6	99.2	82.8	95.4	29

As shown in **Table 3-6** Pathloss prediction accuracy (%) on Starlink constellation., the two models have comparable performance, and both outperform naïve prediction significantly on Starlink constellation. For WP3 constellation, both models outperform naïve prediction, however, model#1 performs much better than model#2, especially when the truth sample is marked as LOS.

We also evaluate prediction accuracy of both models and naïve prediction against Δt, where $\Delta t = 1, 25, 50, 75,$ or 100, on both Starlink and WP3 constellation.

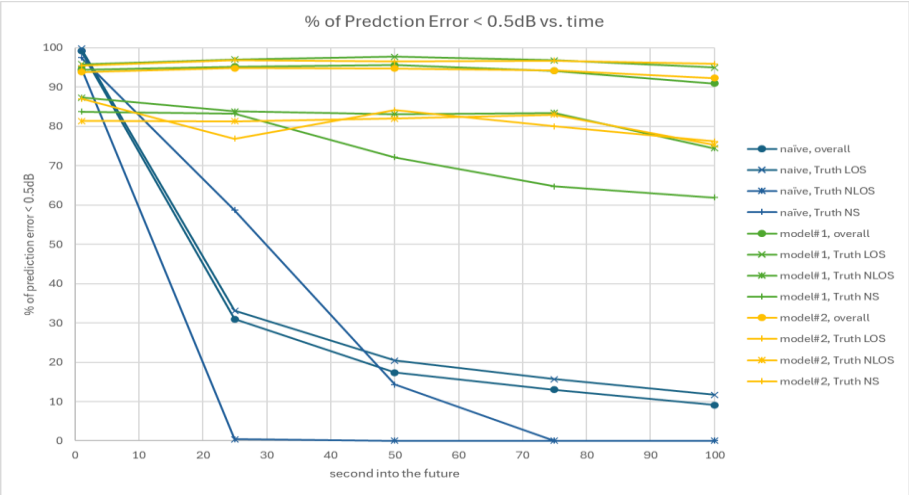

FIGURE 3-31 PREDICTION ACCURACY VS. ΔT (SECOND INTO THE FUTURE) ON STARLINK

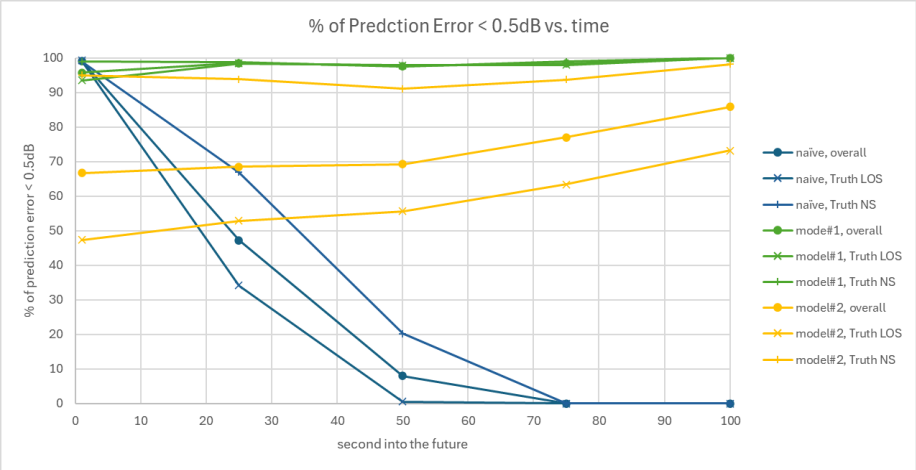

FIGURE 3-32 PREDICTION ACCURACY VS. ΔT (SECOND INTO THE FUTURE) ON WP3 CONSTELLATION.

For Starlink constellation dataset, both prediction models outperform naïve prediction method in prediction accuracy, as shown in **Figure 3-31**. For all three predictions methodologies, prediction accuracy drops as Δt increases. For WP3 constellation dataset, both prediction models outperform naïve prediction method in prediction accuracy, as shown in **Figure 3-32**. However, prediction accuracy of both models increases as Δt increases. This is likely caused by the unique signal status distribution of the WP3 constellation dataset. As shown in **Figure 3-28**, all satellites have LOS signal status (green) from the start of the visible duration, and most satellites have NS signal status (red) in the later part of its visible duration. Such signal status distribution may cause a large portion of entries in training dataset with large Δt have 180dB as their true pathloss, therefore, training both models to predict pathloss to 180dB with large Δt and turning the prediction into a classification task. If the classification is correct, the prediction error is 0.

Obviously, both prediction models outperform naïve prediction, and model#1 has more than overall 90% accuracy on both Starlink and WP3 constellation. As mentioned earlier, predicting neighbor NTN cell's link quality can reduce the number of measurements required. Predicting pathloss in Starlink or WP3 constellation using model#1 only requires one measurement at current time t regardless of Δt. For reference, we evaluate the number of measurements required until naïve prediction reaches 90% prediction accuracy. Specifically, for satellite n and a given Δt, we assume UE performs neighbor cell measurement every second from a randomly selected current time t and count the number of measurements required before the measured pathloss is 0.5dB or less away from the pathloss at $t + \Delta t$.

TABLE 3-8 NUMBER OF MEASUREMENTS REQUIRED FOR NAÏVE PREDICTION TO REACH 90% PREDICTION ACCURACY

Δt	1	25	50	75	100
number of measurements – Starlink constellation	1	19	42	66	96
number of measurements – WP3 constellation	1	17	42	65	90

3.4.2.3 NTN Ray-tracing Multi-UE Dataset

There are two multi-UE datasets, one created from Starlink and the other from WP3 constellation. The former has 10 UE positions and 15 satellites, and the latter has 10 UE positions and 30 satellites. For both datasets, satellite n ($n \in \{1,2,\dots,15\}$ for Starlink and $n \in \{1,2,\dots,10\}$ for WP3) has the same visible time for all the 10 UE. Therefore, for a given constellation dataset, a satellite n has a fixed visible time T_n (in seconds) regardless of UE position. For a combination of satellite n and UE position m, $m \in \{1,2,\dots,10\}$, in a constellation, the simulator samples the same set of data as described in the single-UE dataset section every second. There are 17310 and 29169 such samples in the Starlink and WP3 constellation dataset, respectively. The pathloss of a sample from satellite n and UE position m at time t can be presented as:

$$pathloss_{n,m}[t] = [pl_1, pl_2, \dots, pl_l]_{1xl} \tag{3.4.26}$$

where $0 \le t \le T_n - 1$, l is the multipath number at sample time t, and $pl_1, pl_2, \dots, pl_l$ are pathloss (in dB) associated with path 1 to l.

The pathloss of the strongest signal path of a sample from satellite n to UE position m at time t can be represented as

$$PL_{n,m}[t] = min\big(pathloss_{n,m}[t]\big) \tag{3.4.27}$$

where $0 \le t \le T_n - 1$, and $PL_{n,m}$ is a row vector with length T_n. As mentioned earlier, $PL_{n,m}[t]$ will be directly refered as pathloss (from satellite n to UE m at time t) in the rest of the paper. The same signal status classification mentioned before is applied to the two multi-UE datasets. For Starlink constellation dataset, there are 79.4% of the samples marked as LOS, 16.1% of NLOS, and 4.5% of NS. For WP3 constellation dataset, the signal status distribution of LOS, NLOS, and NS is 89.6%, 0.3%, and 10.1%, respectively. See the signal status distribution in ECEF frame per UE position in **Figure 3-33**.

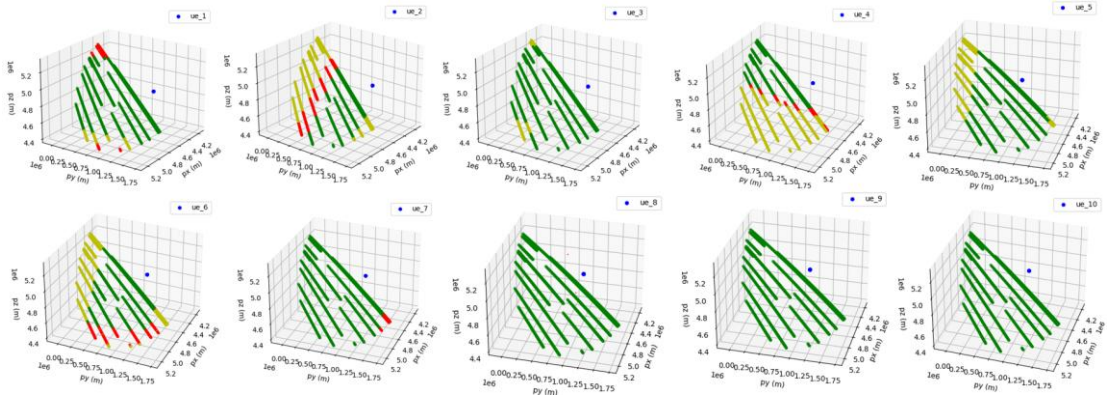

FIGURE 3-33. SIGNAL STATUS DISTRIBUTION IN ECEF FRAME PER UE POSITION: STARLINK CONSTELLATION DATASET.

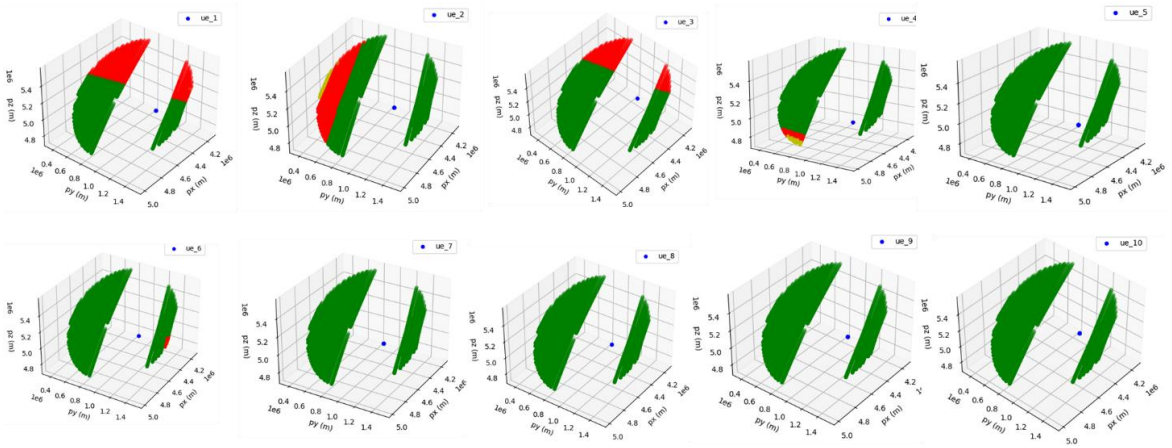

FIGURE 3-34 SIGNAL STATUS DISTRIBUTION IN ECEF FRAME PER UE POSITION: WP3 CONSTELLATION DATASET.

In Starlink constellation dataset, pathlosses of samples classified as LOS and NLOS range from 151dB to 157dB and 160 to 185dB, respectively. In WP3 constellation dataset, pathlosses of samples classified as LOS and NLOS range from 152dB to 155dB and 168 to 195dB, respectively. Pathlosses of NS samples are set to 300dB in both datasets. See the pathloss histogram in **Figure 3-35**

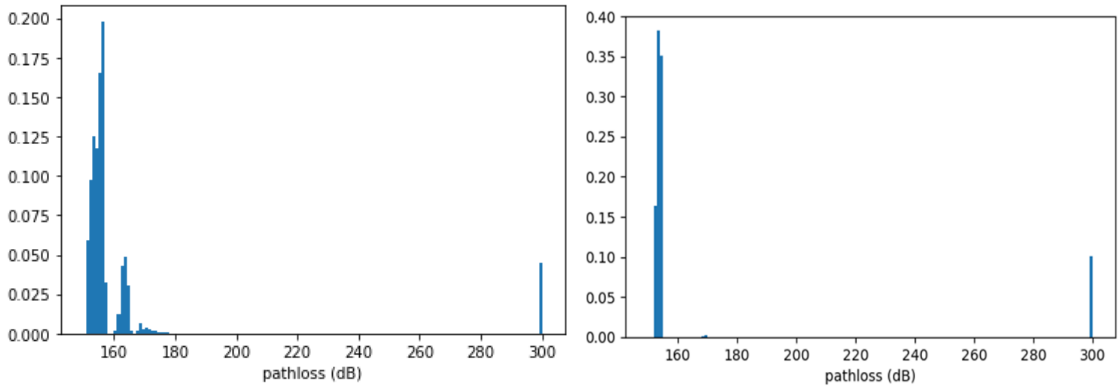

FIGURE 3-35 PATHLOSS HISTOGRAM. THE Y AXIS IS THE DENSITY. LEFT: STARLINK. RIGHT: WP3.

Since the satellite position, velocity, and visible time do not change from one UE position to another, we can reuse the expressions in single-UE database and formulate position (in meter) and velocity (meter per millisecond) for satellite n:

$$P_n[t] = \left[Px_n[t], Py_n[t], Pz_n[t]\right]_{1x3} \quad (3.4.28)$$

$$V_n[t] = \left[Vx_n[t], Vy_n[t], Vz_n[t]\right]_{1x3} \quad (3.4.29)$$

where $0 \leq t \leq T_n - 1$. Px_n, Py_n, Pz_n, Vx_n, Vy_n, and Vz_n are all row vectors with length T_n. Position (in meter) for UE position m in the dataset is represented as,

$$p_m = [px_m, py_m, pz_m]_{1x3} \quad (3.4.30)$$

where px_m, py_m, and pz_m are all real number.

3.4.2.4 Pathloss Prediction on Multi-UE Dataset

The pathloss prediction task is slightly different from the prediction task on single-UE dataset: given satellite n and UE position m data at time t, predict the pathloss (of the strongest signal path) from satellite n to UE position m at time $t + \Delta t$, and $1 \leq \Delta t \leq 120$ (seconds). The following prediction model architecture was experimented for the task.

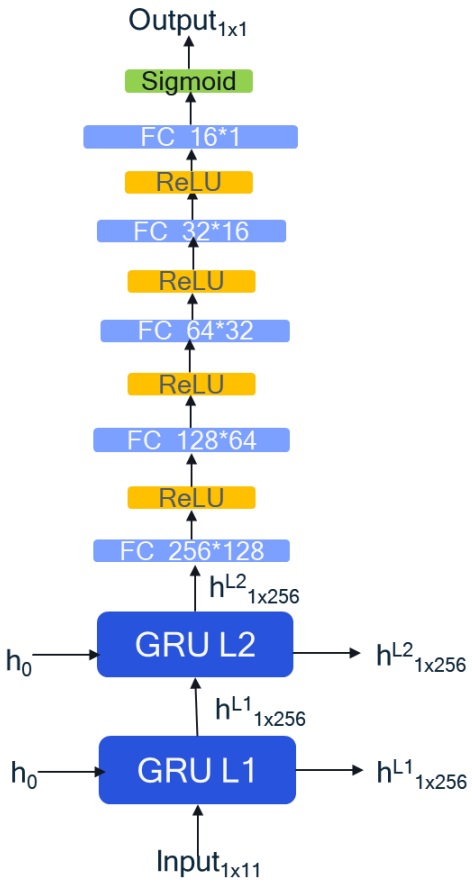

FIGURE 3-36 PREDICTION MODEL FOR MULTI-UE DATASET.

The model consists of two-layer GRU, each with hidden size 256, followed by five layers of DNN with ReLU activation in between and a Sigmoid output layer. h_0 is the initial state of the GRUs and is a zero vector. h^{L1}, and h^{L2} are hidden states of the GRUs. Input data of the model includes the following extracted from satellite n and UE position m data at time t: positions of satellite n and UE m, velocities of satellite n, and pathloss between satellite n and UE position m. In addition, Δt is also included in input. Specifically:

$$input(n, m, t, \Delta t) = \{P_n[t], p_m, V_n[t], PL_{n,m}[t], \Delta t\}$$
$$= \left[Px_n[t], Py_n[t], Pz_n[t], px_m, py_m, pz_m, Vx_n[t], Vy_n[t], Vz_n[t], PL_{n,m}[t], \Delta t\right]_{1\times 11} \quad (3.4,31)$$

The model predicts the pathloss at $t + \Delta t$, so the expected output is the true pathloss at time $t + \Delta t$, or $PL_{n,m}[t + \Delta t]$. The prediction task can be considered as a function of input data that depends on n, m, t, and Δt. The output of the function is an estimated $PL_{n,m}[t + \Delta t]$, or $\widetilde{PL}(n, m, t, \Delta t)$:

$$f\big(input(n, m, t, \Delta t)\big) = \widetilde{PL}(n, m, t, \Delta t) \quad (3.4.32)$$

Therefore, the prediction error can be written as $\widetilde{PL}(n,m,t,\Delta t) - PL_{n,m}[t+\Delta t]$.

For preprocessing, all satellite positions and velocities and all UE positions in a dataset are scaled separately by their axis with standard scalar, i.e. all UE positions on x axis (px_m, $m \in \{1,2,\dots,10\}$) are scaled together with one standard scalar. For pathloss in a dataset, we first reclass all NLOS samples with pathloss 200dB or larger to NS and set pathloss of all NS samples to 200. Then all pathlosses are scaled with a min-max scalar to preserve the relative relationships among pathlosses of the signal status groups.

For each constellation dataset, a dataset is created by iterating through every UE position and collecting data with the method specified in single-UE dataset prediction model section. There are more than 1150000 entries collected from each constellation dataset. They are split randomly so that 80% of the entries are used for training and 20% for testing the prediction model. This leads to two pathloss prediction models, one for the Starlink and another for the WP3 constellation dataset. We also create validation dataset from each constellation by iterating through all UE positions and collecting data as described in the single-UE prediction section. The validation dataset, with more than 200000 entries, is used to evaluate trained prediction model for each constellation dataset.

Training and testing procedures for the two prediction models are identical to the procedures described in the pathloss prediction for single-UE dataset section. Mean square error is the loss criterion, and the Adam optimizer with learning rate 0.0001 is selected. The same mechanisms for early stop and selecting best model are also utilized. The best model for a constellation dataset is evaluated against the validation dataset collected from the same constellation dataset.

The metric to evaluate prediction performance of the model is identical to the metric specified in the prediction for single-UE dataset section. Similarly, the evaluation results are also grouped by signal status for more insight:

TABLE 3-9 PATHLOSS PREDICTION ACCURACY (%) ON STARLINK CONSTELLATION

	Testing	Validation	Naive
Overall	91.9	92.3	30.1
Truth LOS	96.9	95.8	32.2
Truth NLOS	62.3	73.5	21.4
Truth NS	80.7	80.1	19

TABLE 3-10 PATHLOSS PREDICTION ACCURACY (%) ON WP3 CONSTELLATION

	Testing	Validation	Naive
Overall	97.4	97.6	56
Truth LOS	98	98.4	56.6
Truth NLOS	24.8	17.5	34.1

Truth NS	92.7	91.7	51.8

For Starlink constellation, the accuracy of the prediction model, trained by multi-UE dataset (multi-UE model), against validation dataset in **Table 3-9** is similar to that of the prediction model#1 trained by single-UE dataset (singled-UE model#1) in **Table 3-6**. Notice that single-UE model#1 outperforms multi-UE model by 10% if the predicting target is NLOS, and the multi-UE model is 7% more accurate in predicting NS target than single-UE model#1. Like the single-UE model, multi-UE model also significantly outperforms the naïve prediction method.
For WP3 constellation, the prediction accuracy of multi-UE model against the validation dataset in **Table 3-10** is about 1% less than that of the single-UE model in **Table 3-10**. Notice that the multi-UE model is 7% less accurate on predicting NS target, comparing with the single-UE model. Multi-UE model also performs poorly predicting NLOS target. This is likely due to the low portion (0.3%) of the signal classified as NLOS in the WP3 constellation dataset, causing insufficient data to train the model for predicting pathloss of NLOS signal.
The prediction accuracy of the multi-UE model and naïve prediction are evaluated against Δt, where Δt=1,25,50,75, or 100, on both Starlink and WP3 constellation.

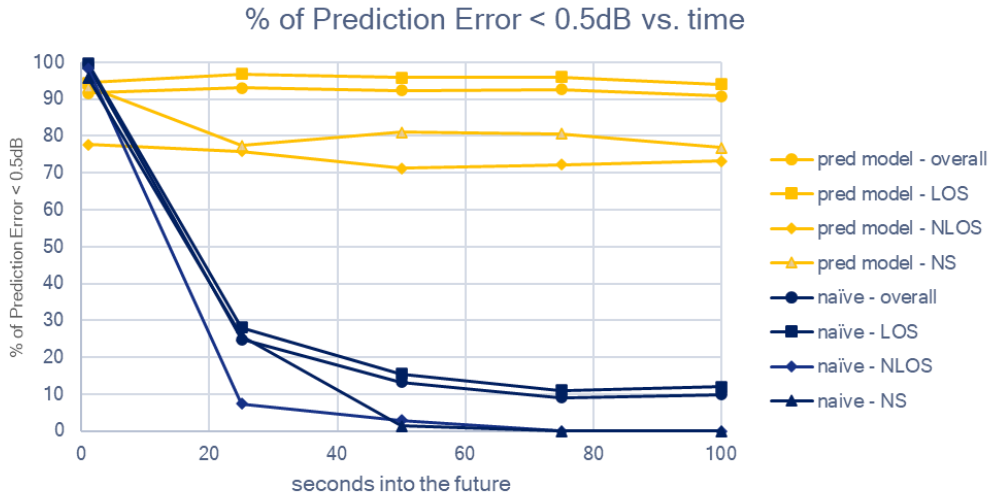

FIGURE 3-37 PREDICTION ACCURACY VS. ΔT (SECOND INTO THE FUTURE) ON STARLINK CONSTELLATION.

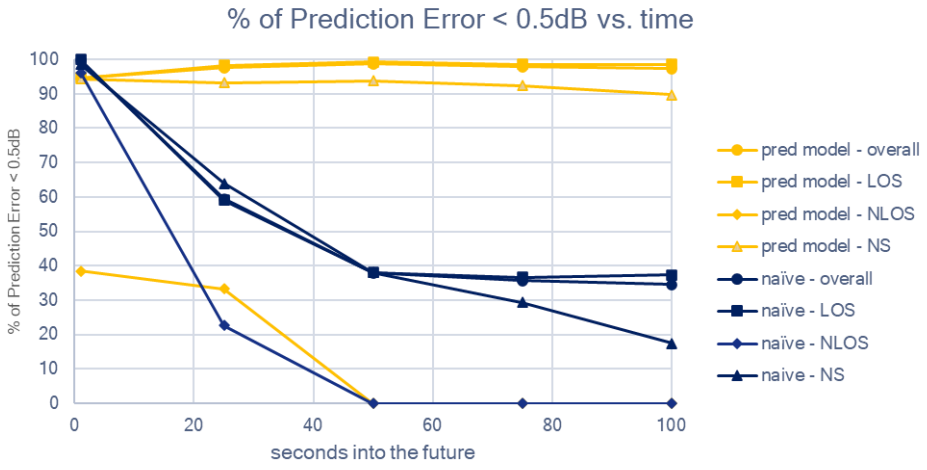

FIGURE 3-38 PREDICTION ACCURACY VS. ΔT (SECOND INTO THE FUTURE) ON WP3 CONSTELLATION.

As expected, multi-UE model outperforms the naïve prediction when Δt>1. For reference, we also include the number of measurements required until naïve prediction reaches 90% prediction accuracy in the **Table 3-11**.

TABLE 3-11 NUMBER OF MEASUREMENTS REQUIRED FOR NAÏVE PREDICTION TO REACH 90% PREDICTION ACCURACY

Δt	1	25	50	75	100
number of measurements – Starlink constellation	1	18	42	67	93
number of measurements – WP3 constellation	1	14	39	64	90

To further exam the if the prediction model can generalize UE location, cross validation (CV) is conducted on Starlink constellation dataset with the following step:

1. Select a UE location (CV UE location) to generate a cross-validation dataset
2. Use 9 other UE locations to generate training and testing dataset and train the model
3. Iterate through all 10 UE locations for cross validation

For each cross-validation iteration, we compare the LOS signal pathloss prediction accuracy between testing and cross validation dataset:

TABLE 3-12 PREDICTION ACCURACY AGAINST TESTING AND CROSS VALIDATION DATASET BASED ON UE LOCATION ON STARLINK CONSTELLATION.

CV UE Location	Testing	Validation
1	93.8	74.6
2	95.4	38.9
3	97.1	48.3
4	94.1	50.1
5	93.8	63.6
6	97.4	21.5
7	95.4	66.3
8	95.6	77.8
9	94.9	8.5
10	93.4	30.3

As shown in **Table 3-12**, the model predicts much poorly on cross-validation dataset, comparing with prediction accuracy on testing dataset. One possible explanation is that the model may first classify CV UE location to its nearest neighbor and then predict the pathloss based on the nearest neighbor's data in the testing dataset. For example, UE location 7 is the

closest neighbour of UE location 8, and they have relative similar signal status distribution, as shown in **Figure 3-39**. If the model is trained with data from all UE locations other than UE location 8 and cross-validated with data from UE location 8, the trained model may classify UE location to 7 and predict the pathloss according to UE location 7's signal distribution. Therefore, some LOS data in UE location 8 may be predicted as NLOS or NS, decreasing the prediction accuracy.

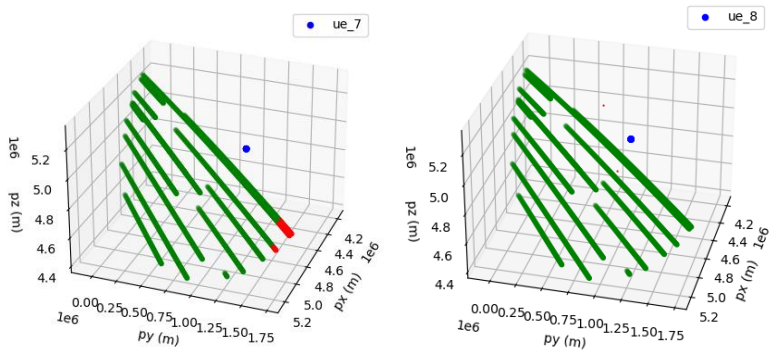

FIGURE 3-39 SIGNAL STATUS DISTRIBUTION OF UE LOCATION 7 AND 8 IN STARTLINK CONSTELLATION DATASET

Another example is the very different signal status distributions between UE location 9 and its closest neighbor UE location 6, as shown in **Figure 3-40**. Classifying UE location 9 to 6 will lead to many LOS signal to be predicted as NLOS or NS, resulting in the wors prediction accuracy in **Table 3-12**.

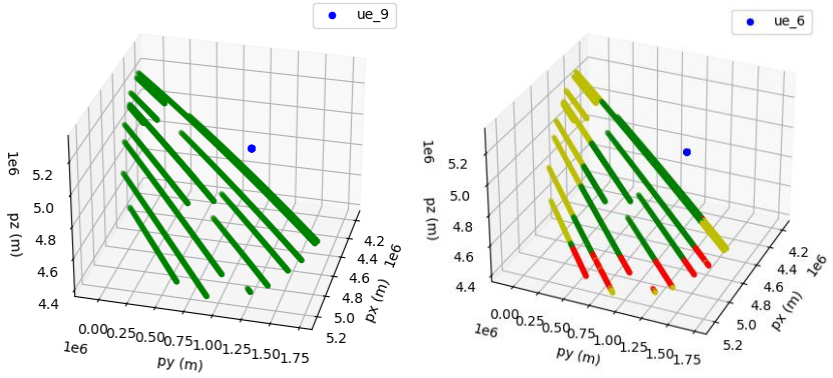

FIGURE 3-40 SIGNAL STATUS DISTRIBUTION OF UE LOCATION 6 AND 9 IN STARTLINK CONSTELLATION DATASET

Conclusion

This network function focuses on enabling UE-side AI-based NTN link quality prediction, reducing UE's reliance on frequent measurements and improving mobility efficiency. In D4.2, we introduced an AI/ML architecture aligned with the relevant 3GPP work, supporting modular deployment of data collection, model training, model transfer, inference, and lifecycle management across UE, ground and satellite MEC servers. In D4.6, we refined this concept for distributed NTN architectures, leveraging feeder satellites for data collection and training while maintaining UE-side inference to minimize latency. Our design incorporates dynamic model transfer, lifecycle management, and OTT user-plane interactions to simplify the integration of different network components without major protocol changes. Collectively, these efforts establish a robust AI-enabled framework for NTN mobility optimization.

Numerically, we show that by stacking a GRU and a neural network, it is possible to obtain a model to predict the pathloss of the strongest signal from a NTN LEO satellite to a UE location on the ground in the near future. Such model can have very high prediction accuracy, comparing with naïve prediction method. Although the model has decent prediction accuracy on the satellite and UE location whose data is included in the training dataset, its prediction accuracy drops significantly when predicting pathloss for an UE location not included in the training data. This implies an inadequate capability to generalize well across different UE locations or environment. Therefore, we suggest considering to use multiple prediction models, i.e., one model for one UE location/environment.

Note:

D4.5 also provides input relevant to link quality prediction analysis. However, compared to D4.5, D4.6 places link quality prediction inference on the UE and formalizes training/model transfer on the feeder-satellite MEC via OTT user-plane interactions, so predictions can guide satellite switching without continuous neighbor measurements.
Moreover, D4.6 extends the predictor beyond geometric FSPL by introducing a custom NTN ray-tracing multipath channel and evaluating GRU+DNN models on single-UE and multi-UE datasets (Starlink and WP3 constellation), reporting >90% accuracy and highlighting generalization limits across unseen UE locations, capabilities not covered previously.

4. CONCLUSIONS

This deliverable, D4.6, presents a comprehensive final report on the design and implementation aspects of AI-enabled RAN intelligent controller (RIC). The overall objective was to address the dynamic allocation of radio resources, particularly focusing on the Radio Resource Management (RRM), for integrated Terrestrial and Non-Terrestrial Networks (TN-NTN). This problem domain is central to achieving Quality of Service (QoS) assurance and overall system optimization across non-Real time (non-RT) and near-Real time (n-RT) operation layers. Such objectives become increasingly critical under conditions characterized by non-uniform user traffic demand and user density, and heterogenous and unpredictable channel conditions.

To this end, this report primarily focused on providing the in depth analysis on the deployment of AI/ML functions over the *Conventional Architecture*, and *Distributed Architecture* proposed in D3.5 of 6G-NTN project. As initially outlined in D4.2, where several RRM-related network functions were identified as potential candidates for AI integration and since each resource network function operates under different objectives, time scales, data dependencies, therefore a unified one-size-fits-all deployment framework was not feasible to meet the service requirements. Thereby, the adopted AI/ML architecture leverages the flexibility of 3GPP compliant O-RAN and AI-RAN frameworks, incorporating hierarchical intelligence distribution and split learning paradigms to adapt to local requirements and computational demands of each network function.

Beyond the AI/ML deployment aspects, D4.6 highlights the necessity of adaptive learning optimization within each network function. The non-static nature of TN-NTN environment implies that the static or heuristic models are insufficient to adopt the dynamic nature of network requirements. Therefore, continual adaptation and self-evolving AI network are required to ensure performance sustainability under evolving traffic and channel conditions. Accordingly, each network function presented in this report formulated its RRM problem, utilizing the KPMs from the network or UE, employing advanced and efficient deep learning algorithms incorporating the coordinated mechanism during training and inference, resulting in significant overall efficiency gains.

Future Work on AI-Enabled RIC

The continued evolution of AI-native RIC architecture opens several promising development directions and are worth to study further:

- Deploying the Agentic AI capable of autonomous policy refinement to enhance RRM robustness and adaptability in a *scalable manner*.
- Advancing the TRL of AI-driven RIC implementations in terms of disaggregated and edge-deployed AI would allow more realistic framework to meet different use case requirements simultaneously.

- Investigating foundation models and cross-domain transfer learning for RAN control may provide a unified intelligence layer capable of handling diverse TN-NTN use cases with minimal retraining.

REFERENCES

[1] 6G-NTN Deliverable 2.1, “Use Case Definition”, v1.0.

[2] 6G-NTN Deliverable 3.3, “Software Defined Payload and its Scalability”, v1.0.

[3] 6G-NTN Deliverable 3.5, “Report on 3D/Multilayered NTN Architecture”, v1.0.

[4] 6G-NTN Deliverable 4.3, “Open Datasets for 6G-NTN Data Driven Radio Access Networks”, v1.0.

[5] 6G-NTN Deliverable 4.2, “Report on 6G-NTN Radio Controller”, v1.0.

[6] 3GPP, “Study on New Radio Access Technology: Radio Access Architecture and Interfaces,” v14.0.0, 3GPP, Tech. Report. TS 38.801, April,2017.

[7] O-RAN Working Group 1, “O-RAN Architecture Description 5.00,”5.0.0, OpenRAN, Tech. Spec, July 2021.

[8] AI-RAN Alliance, “Integrating AI/ML in Open-RAN: Overcoming Challenges and Seizing Opportunities,” AI-RAN Alliance, Tech. Report,March 2024.

[9] 6G-NTN Deliverable 3.6, “Report on 3D/Multilayered NTN Architecture”, v1.0.

[10] 3GPP, “Study on new radio access technology: NG-RAN; Architecture description,” v18.4.0, 3GPP, Technical Reports TS 38.401, 2024

[11] Novlan, T., Andrews, J. G., Sohn, I., Ganti, R. K., & Ghosh, A. (2010). Comparison of Fractional Frequency Reuse Approaches in the OFDMA Cellular Downlink. In 2010 IEEE Global Telecommunications Conference GLOBECOM 2010 (pp. 1-5). IEEE.

[12] Shahid, Syed Maaz, et al. "Load balancing for 5G integrated satellite-terrestrial networks." IEEE Access 8 (2020): 132144-132156.

[13] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, “Qtran:Learning to factorize with transformation for cooperative multi-agent reinforcement learning,” in International conference on machine learning. PMLR, 2019, pp. 5887–5896.

[14] 3GPP, “NR; Study on international mobile telecommunications (IMT) parameters 6.425-7.025GHz, 7.025-7.125GHz and 10.0-10.5GHz,” v18.0.1, 3GPP, Technical Reports TR 38.921, 2024.