D4.7: Final report on AI-based radio resource management, RAN softwarisation and onboard processing

Revision: v1.2F

Work package	WP 4
Task	Task 4.3, Task 4.4 and Task 4.5
Due date	31/03/2026
Submission date	30/03/2026
Deliverable lead	CTTC
Version	V1.0
Authors	Luis Blanco, Cristian J. Vaca-Rubio, Engin Zeydan, Marius Caus (CTTC) Oriol Font, Piotr Gawłowicz (SRS) Nathan Borios, Albekaye Traore (TAS-F) LI Zheng (ORANGE)
Reviewers	Benjamin Barth (DLR)
Abstract	This document reports the final outcomes of Tasks 4.3 “Data-enhanced Radio Intelligent Controller Design”, Task 4.4 “Onboard Processing and Communication Capabilities” and Task 4.5 “RAN Softwarisation”. Building upon the O-RAN architecture and interfaces, RIC placement strategies are proposed, and radio resource management algorithms are conceived for near- and non-real time timescales. On the basis of a fully regenerative payload, the document provides a high-level description of the on-board processing capabilities. Finally, to enable the proof-of-concept, the required modifications to the gNB radio protocol and the inter-block communication have been investigated.
Keywords	O-RAN, RRM, non-RT RIC, near-RT RIC, NTN, TN, OBP

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	30/09/2024	Document creation	CTTC
V0.2	04/02/2025	Functional split analysis	CTTC
V0.3	13/05/2025	Finalization of the chapter on on-board processing	TAS
V0.4	20/06/2025	RAN softwarisation	SRS
V0.5	03/09/2025	RIC placement and federated learning	CTTC
V0.6	15/09/2025	Near-RT RIC solutions	CNIT
V0.7	30/09/2025	Probabilistic forecasting	CTTC
V0.8	19/03/2026	Clustering algorithms	CNIT
V0.9	23/03/2026	Multivariate probabilistic forecasting	CTTC
V1.0	26/03/2026	QA review	DLR
V1.1	27/03/2026	Revision according to the received comments from the QA review	CTTC
V1.2.F	30/03/2026	Approved final version for EC portal submission	DLR

DISCLAIMER

Co-funded by the European Union 6GSNS

5G-STARDUST (*Satellite and Terrestrial Access for Distributed, Ubiquitous, and Smart Telecommunications*) 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 101096573.

COPYRIGHT NOTICE

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	**R**	
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

In this deliverable, 5G-STARDUST investigates aspects related to network architecture, radio resource management (RRM), on-board processing (OBP) and radio access network (RAN) softwarisation.

The first part of the document is devoted to analyzing and designing architectural and functional split options in integrated terrestrial and non-terrestrial networks. By leveraging the modularity of the open-RAN (O-RAN) architecture, multiple configurations are evaluated for distributing RAN functions (radio unit (RU), distributed unit (DU), centralized unit (CU)) and user plane function (UPF) components across space and ground domains. The analysis has identified key trade-offs related to latency, onboard complexity, feeder link dependency, and autonomy, culminating in three principal design options: (i) DU and RU onboard, (ii) full gNB onboard, and (iii) full gNB with UPF and space edge computing (SEC) functions onboard. For each option, sub-variants were considered depending on whether the functions were co-located on the same satellite or distributed across multiple nodes connected via inter-satellite-links (ISLs).

RAN intelligent controller (RIC) placement strategies are proposed to complement these split designs, outlining both ground-based and hybrid control architectures. The mapping between split configurations and RIC distribution showed that control plane responsiveness and optimization capabilities depend heavily on the persistence and latency of inter-domain links. While each architectural option offers different benefits, increasing autonomy and reducing latency generally require migrating more intelligence onboard. However, this comes with higher processing demands, greater implementation complexity, and power constraints. Therefore, architectural decisions must be tailored to specific mission objectives, satellite class, and required levels of ground independence. Beyond performance metrics, function placement strategies must also account for operational feasibility, spectrum regulations, and interoperability with existing 5G core network (CN). Among the three principal architectural options studied, the full gNB stack (RU, DU, CU) integrated onboard a single satellite- emerges as the most practical and balanced choice in terms of protocols and hardware standpoint under current technology constraints. It avoids fronthaul and midhaul timing bottlenecks, reduces signalling latency, and enables low-complexity RAN management while preserving compatibility with 5G CN interfaces. Its self-contained structure simplifies orchestration, enhances scalability, and lowers dependency on persistent feeder links or high-throughput ISLs. These qualities make the full gNB onboard particularly suited for low-Earth orbit (LEO) constellations targeting broadband non-terrestrial network (NTN) deployments in the near term.

In alignment with the O-RAN specification, this deliverable classifies the problems according to the timescale. The entity that supports large time scale RAN optimization (>1s) is the non-RT RIC. In this case, the software application that is designed to run on the non-real time (non-RT) RIC is referred to as rApp. For smaller timescales (between 10ms and 1 s), the RAN optimization relies on the near-RT RIC. In this context, the xApps are applications that are designed to run on the near-RT RIC.

Focusing attention on large timescales (>1s), the deliverable proposes innovative solutions to enhance traffic predictions, which are exploited to perform resource allocations in NTN. The designs described in this deliverable leverage on previous research on Kolmogorov-Arnold networks (KANs), which is detailed in [1]. A key contribution of this deliverable is the introduction of probabilistic forecasting models for radio resource management. By moving beyond deterministic point predictions, these models explicitly quantify uncertainty, allowing operators to balance spectral efficiency against service reliability. In particular, KANs with

probabilistic outputs, Gaussian and T-Student variants, demonstrate superior trade-offs compared to both point-forecasting and MLP-based baselines. While T-Student KAN achieves higher efficiency at the expense of underprovisioning risk, the Gaussian KAN offers more conservative and calibrated forecasts, reducing QoS risks. This probabilistic approach is shown to be indispensable for adaptive PRB allocation in realistic NTN environments. The second enhancement applies federated learning (FL) to KAN. The motivation for the FL approach with KAN for the traffic prediction problem considered here stems from the unique challenges presented by LEO satellites and their dynamic environment. Due to the constantly changing positions of LEO satellites, maintaining consistent communication and synchronization between satellites and ground station is critical when making accurate artificial intelligence (AI) decisions. However, traditional centralized learning methods have difficulty adapting to these dynamic schedules. Therefore, FL with the KAN approach is particularly advantageous for these scenarios, especially when decentralized model training is required. FL with the KAN provides a robust alternative by enabling local model training and periodic updates of models, reducing the dependency on always on real-time connections with central servers. Numerical results obtained with the satellite network data show a significant reduction in training and test loss when compared to the traditional solutions.

Within the non-real time RIC context, it has been analyzed the benefit of considering state-of-the-art multivariate probabilistic techniques in multi-beam satellites. AI-based multivariate probabilistic models are able to learn the complex dependencies of multi-beam traffic demands. As shown in this deliverable, the proposed approaches are able to provide significant bandwidth saving and reduced underprovisioning rate in real satellite operation data.

With respect to near-RT RIC solutions, the deliverable explores two different, and potentially synergic, applications of and machine learning (ML) algorithms to the 5G-STARDUST system:

- The first one is explicitly designed to support the user-centric beamforming solutions designed in [2] and it is aimed at reducing the computational complexity of the designed scheduling algorithms while improving the performance in terms of the achieved capacity. To this aim, clustering algorithms are implemented to cluster the on-ground users based on the available ancillary information at the gNB (either the channel state information, CSI, or the location); the clusters group users with similar channel or location and this information is exploited by the scheduler to automatically avoid ill-conditioned matrices in the beamforming algorithms (something that called for computationally extensive iterations in the non-AI based solutions in [2]). The designed algorithm indeed reduces the computational complexity and also improves the achieved capacity, in particular in low-medium traffic scenarios. In high traffic scenarios, the designed solution is not performing significantly better, and this calls for more advanced solutions, such as a Hierarchical Scheduling algorithm proposed in this report as a potential enhancement step for future activities.

- The second AI-based solution exploits neural networks (NNs) to estimate the signal-to-interference plus noise ratio (SINR) on-board the gNB. This tool can be clearly included in the user-centric beamforming algorithms (from both this document or [2]), as it is able to estimate the SINR experienced by the user for a given scheduling and time slot without needing to compute the beamforming coefficients for a tentative scheduling allocation. In fact, without SINR prediction, potentially complex (i.e., computationally demanding) iterative approaches to evaluate the potential performance of beamforming for a given scheduling in the time slot would be needed. While this is the main potential application of the designed solution, notably a NN capable of estimating the SINR has a potential application also for other algorithms or steps in the overall system management. As discussed in this report, the designed NN algorithm achieves a reduction of a factor 5 in

the big-o complexity with CSI-based beamforming and of 2 orders of magnitude with location-based beamforming. The SINR estimation accuracy achieved by the NNs, evaluated in terms of the root mean square error (RMSE), remains under 1 dB in most of the cases.

To assess the feasibility of hosting the RIC and network functions on board satellites, this document presents a high-level study on the development of regenerative payloads with OBP capabilities. The study classifies satellites into three categories: traditional telecommunication satellites, hybrid satellites that support both telecommunications and edge computing, and satellites primarily designed for edge computing. For each category, the current state-of-the-art in CPU, RAM, and storage configurations is examined. In addition, tools and methodologies for profiling application-level resource usage are reviewed to better understand the limitations of OBP.

To enable the PoC, general stack modifications have been introduced, with the most significant features detailed in this deliverable, with respect to the preliminary report, architectures and features of the CU/DU have been extended. The splits supported include option 2, 6 and 7.2x. Furthermore, radio functions have been modified to enable the NTN functionality within an O-RAN compatible architecture. For instance, an external SIB19 generator has been implemented, with a rapid updating mechanism in the gNB. The HARQ management has been improved to disable retransmissions or to extend the number of processes. Additional improvements include the feeder link support, by signalling the corresponding propagation delay and by compensating the Doppler shift. To further optimize the RAN, the E2 interface development, which enables the interconnection of the CU and the DU with near-RT RICs, has been documented. Finally, a baseline implementation of the O1 interface has also been described to enable the remote and automated configuration of the gNB.

TABLE OF CONTENTS

Disclaimer 2
Copyright notice 3
1 INTRODUCTION 18
2 ARCHITECTURAL SPLIT AND RADIO INTELLIGENCE CONTROLLER PLACEMENT 20
2.1 O-RAN Architecture Overview 21
2.2 Option 1: DU and RU Onboard (Split 2) 26
2.2.1 Option 1a: DU in single satellite 27
2.2.2 Option 1b: DU and RU in different satellites 27
2.3 Option 2: Full gNB Onboard 27
2.3.1 Option 2a: Monolithic gNB 29
2.3.2 Option 2b: Distributed gNB Onboard 30
2.4 Option 3: Full gNB Onboard with UPF Functions 31
2.4.1 Option 3a: The same satellite 33
2.4.2 Option 3b: Different satellites 33
2.5 RIC placement options 34
2.5.1 Architecture Extension I: Splitting near-RT RIC Across Earth and Space 35
2.5.2 Architecture Extension II: Network of near-RT RIC deployment in Space with dynamic E2 node assignment 36
2.5.3 Architecture Extension III: Splitting Non-RT RIC Across Earth and Space Distributed Clusters 37
2.6 Discussions and Future Work 40
3 AI-TECHNIQUES FOR NON-REAL-TIME RRM 44
3.1 Probabilistic forecasting-enabled Kolmogorov Arnold Networks 44
3.1.1 Problem Statement 44
3.1.2 Probabilistic Kolmogorov-Arnold Networks (P-KANs) 45
3.1.3 Performance evaluation 46
3.2 Federated KANs 53
3.2.1 FL and KANs for NTN 54
3.2.2 General Architecture 54
3.2.3 Federated KAN 55
3.2.4 Evaluation Results 57
3.2.5 Fed-KAN Applications in NTNs 60
3.3 Multivariate probabilistic forecasting for multibeam satellites 61
3.3.1 Shifting from single-point predictions to multi-beam probability forecasts of demands 61

3.3.2 Multivariate Deep Time Series Forecasting techniques 63
3.3.3 Evaluation results 68
4 AI TECHIQUES FOR NEAR-REAL TIME RRM 74
4.1 System Model 74
4.2 Clustering-aided Scheduler for user-centric digital beamforming 80
4.2.1 Rationale 80
4.2.2 Clustering basics 82
4.2.3 Clustering-Aided PQS (CA-PQS) 83
4.2.4 Computational complexity 91
4.2.5 Performance assessment 99
4.2.6 Conclusions and future enhancements 105
4.3 AI-aided SINR estimation 106
4.3.1 Rationale 106
4.3.2 Self-attention basics 107
4.3.3 DMHSA model for SINR estimation in user-centric beamforming systems 108
4.3.4 Computational complexity 111
4.3.5 Model training 113
4.3.6 Performance evaluation 115
5 ONBOARD PROCESSING 119
5.1 Processing and storage capabilities of satellites 121
5.1.1 Traditional telecommunication satellites 122
5.1.2 Hybrid telecommunication and edge computing satellites 122
5.1.3 Edge computing-first satellites 123
5.2 On-board architectures 124
5.2.1 CU/DU splitting 124
5.2.2 Full gNB on-board 126
5.2.3 Full gNB on-board + UPF 127
5.3 Resource consumption measurement tools and methodologies 128
5.3.1 Tools for measuring resource consumption 129
5.3.2 Methodology for resource profiling 129
5.3.3 Application to the 5G NTN testbed 130
6 RAN SOFTWARIZATION 131
6.1 O-RAN Architecture and functional splits 131
6.1.1 Disaggregation status and supported splits 131
6.1.2 Discussion on gNB deployment options in NTN scenarios 132
6.2 NTN-specific enhancements in the gnb 133
6.3 RIC and O&M service integration 135

6.3.1 O1 interface introduction 135
7 CONCLUSIONS 136
8 REFERENCES 141

LIST OF FIGURES

FIGURE 1: GENERIC O-RAN ARCHITECTURE WITH INTERFACES BETWEEN FUNCTIONAL AND INTELLIGENCE COMPONENTS [4]. 22

FIGURE 2: DEPLOYMENT OPTIONS FOR DU ONBOARD. LEFT: DU AND RU IN SINGLE SATELLITE. RIGHT: DU AND RU INTERFACED OVER ISL. 26

FIGURE 3: DEPLOYMENT OPTIONS FOR FULL GNB ONBOARD. LEFT: OPTION 2A (MONOLITHIC GNB). RIGHT-TOP: OPTION 2B WITH CU AND DU TOGETHER AND RU SEPARATED OVER ISL. RIGHT-BOTTOM: OPTION 2B WITH DU AND RU TOGETHER AND CU SEPARATED. ISLS SUPPORT FRONTHAUL OR F1-C/U, DEPENDING ON SPLIT. 29

FIGURE 4: OPTION 3: FULL GNB ONBOARD WITH UPF FUNCTIONS 32

FIGURE 5: ARCHITECTURAL EXTENSION I: NEAR-RT RIC SPLIT BETWEEN EARTH AND SPACE 35

FIGURE 6: ARCHITECTURAL EXTENSION II: NETWORK OF NEAR-RT RIC WITH DYNAMIC ASSIGNMENT 37

FIGURE 7: ARCHITECTURAL EXTENSION III: SPLITTING NON-RT RIC ACROSS EARTH AND SPACE DISTRIBUTED CLUSTERS. 38

FIGURE 8. ILLUSTRATION OF P-KAN ARCHITECTURE. 47

FIGURE 9. MEAN ABSOLUTE ERROR OF PROBABILISTIC METHODS VS ITS POINT FORECAST COUNTERPARTS. IN PROBABILISTIC METHODS, THE MEDIAN IS CONSIDERED FOR THE COMPUTATION OF THE MAE. 48

FIGURE 10. PERCENTAGES OF OVER-/UNDER-PROVISIONING WITH THE DIFFERENT PROBABILISTIC FORECASTING TECHNIQUES WITH 99-TH PERCENTILE, AND POINT-FORECASTING ALGORITHMS. BANDWIDTH SAVING IS ALSO PROVIDED IN COMPARISON WITH STATIC BANDWIDTH ALLOCATION. 49

FIGURE 11. BANDWIDTH SAVING VS UNDERPROVISIONING PERCENTAGE 49

FIGURE 12. CONTINUOUS RANKED PROBABILITY SCORE METRIC FOR THE P-KAN AND MLP WITH DIFFERENT CONSIDERED DISTRIBUTION (GAUSSIAN AND T-STUDENT). 50

FIGURE 13. FIC METRIC FOR P-KAN AND MLP WITH DIFFERENT UNDERLYING DISTRIBUTIONS (GAUSSIAN AND T-STUDENT) 51

FIGURE 14. NUMBER OF PARAMETERS FOR THE DIFFERENT POINT-FORECAST AND PROBABILISTIC FORECASTING TECHNIQUES PRESENTED. 51

FIGURE 15. UPPER FIGURE: GAUSSIAN P-KAN 24H FORECAST FOR DIFFERENT CONFIDENCE INTERVALS. LOWER FIGURE: PERCENTILE 99^{TH} VS STATIC PRB ALLOCATION OVER TIME AND ACTUAL PRB DEMANDS. 52

FIGURE 16. T-STUDENT P-KAN PREDICTION OF 24H PRB DEMANDS. UPPER FIGURE: 50% AND 90% CONFIDENCE INTERVALS. LOWER FIGURE: COMPARISON BETWEEN 99% FORECAST AND STATIC ALLOCATION. 53

FIGURE 17: GENERAL ARCHITECTURE OF NTN AND THE EXISTENCE OF DIVERSE APPLICATION TRAFFIC ON EACH SEPARATE BEAM OF SATELLITES. 55

FIGURE 18: FEDERATED KAN METHODOLOGY APPLIED WITHIN NTN SCENARIO. 57

FIGURE 19: FED-KAN CLASSIFIER LAYERS WITH INPUT (UPLINK AND DOWNLINK TRAFFIC OF LAST W HOURS), HIDDEN AND OUTPUT LAYERS (NEXT HOUR PERCENTAGE PREDICTION OF TRAFFIC CATEGORIES). 58

FIGURE 20: AVERAGE TRAINING MSE LOSS OVER FED-KAN ROUNDS. 59

FIGURE 21. ILLUSTRATION OF THE MULTIVARIATE PROBABILISTIC CONCEPT IN MULTIBEAM SATELLITES. 63

FIGURE 22. BASIC ARCHITECTURE OF N-BEATS. EXTRACTED FROM [43]. 64

FIGURE 23. N-HITS ARCHITECTURE (EXTRACTED FROM [44]). 66

FIGURE 24. PARETO TRADE-OFF BETWEEN BANDWIDTH SAVING AND UNDER-PROVISIONING (QOS RISK) 71

FIGURE 25. MEAN BANDWIDTH SAVING AND UNDERPROVISIONING RATIO FOR THE ASSESSED TECHNIQUES. 72

FIGURE 26. QUANTILE LOSS FOR 99-TH PERCENTILE 73

FIGURE 27: SERVICE AREA AND GROUND TRACK FOR STANDALONE USER-CENTRIC BEAMFORMING WITH $latS = 45°N, lonS = 5°E$, $\varepsilon S = 30°$, $hsat = 1000$ KM (5G-STARDUST POLAR CONSTELLATION, $i = 99°$). 74

FIGURE 28: EXAMPLE OF VISIBLE USERS ACCORDING TO THE NTN NODE POSITION ON ITS ORBIT WITH $latS = 45°N, lonS = 5°E$, $\varepsilon S = 30°$, $hsat = 1000$ KM (5G-STARDUST POLAR CONSTELLATION, $i = 99°$). 76

FIGURE 29: EXAMPLE OF CLUSTERING IN THE GEOGRAPHICAL OBSERVATION SPACE, $\mathcal{FGEOtR}$, WITH $NC = 5$ CLUSTERS. 83

FIGURE 30: EXAMPLE OF CLUSTERING IN THE CSI OBSERVATION SPACE, $\mathcal{FCSItR}$, WITH $NC = 5$ CLUSTERS. 83

FIGURE 31: HIGH-LEVEL DIAGRAM OF CLUSTERING AIDED SCHEDULING. 84

FIGURE 32: HIGH-LEVEL DIAGRAMS OF PQS (LEFT) AND PRS (RIGHT) FOR CLUSTER AIDED SCHEDULING. 84

FIGURE 33: COMPARISON OF THE COMPUTATIONAL COMPLEXITY OF THE BENCHMARK PQS ALGORITHM (D4.5) AND THE CA-PQS WITH THE CONSIDERED CLUSTERING ALGORITHMS. NOTE: HAC-3 IS OVERLAPPED WITH DBSCAN. 98

FIGURE 34: ACHIEVED AND UNMET CAPACITY WITH CA-PQS FOR (LB-)MMSE BEAMFORMING. 101

FIGURE 35: ACHIEVED AND UNMET CAPACITY WITH CA-PQS FOR CBF BEAMFORMING. 102

FIGURE 36: COMPARISON OF THE ACHIEVED AND UNMET CAPACITY WITH CA-PQS (KMEANS++ GEO) OPTIMAL PQS WITH (LB-)MMSE BEAMFORMING. 104

FIGURE 37: COMPARISON OF THE ACHIEVED AND UNMET CAPACITY WITH CA-PQS (KMEANS++ GEO) OPTIMAL PQS WITH CBF BEAMFORMING. 104

FIGURE 38: COMPARISON OF THE CDF OF THE NUMBER OF ALLOCATIONS PER USER WITH CA-PQS (HAC GEO) OPTIMAL PQS WITH (LB-)MMSE AND CBF BEAMFORMING. TRAFFIC CONFIGURATION: $5{,}100$ MBPS (SOLID LINE) AND $20{,}500$ MBPS (DASHED LINE). 105

FIGURE 39: MULTI-HEADED SELF-ATTENTION DIAGRAM. 108

FIGURE 40: DMHSA SINR ESTIMATION MODEL. 109

FIGURE 41: EXAMPLE OF LEARNT PE VECTORS ($NC = 8$) 110

FIGURE 42: COMPUTATIONAL COMPLEXITY AS A FUNCTION OF NC. 113

FIGURE 43: SINR ESTIMATION ERROR DISTRIBUTION WITH RANDOM SCHEDULER. 116

FIGURE 44: SINR ESTIMATION ERROR DISTRIBUTION WITH RANDOM SCHEDULER FOR DIFFERENT $NUE, sched$ VALUES..... 116

FIGURE 45: CDF OF THE SINR ESTIMATION ABSOLUTE ERROR WITH PQS SCHEDULING. 118

FIGURE 46: PAYLOAD AND GROUND ARCHITECTURE - SPLIT 2 (DU/RU ON BOARD, CU ON GROUND) 125

FIGURE 47: PAYLOAD AND GROUND ARCHITECTURE - GNB ON BOARD AND UPF/CN ON GROUND 126

FIGURE 48: PAYLOAD AND GROUND ARCHITECTURE - GNB + UPF ON BOARD AND CORE NETWORK ON GROUND..... 127

FIGURE 49: FUNCTIONAL SPLIT OPTIONS SUPPORTED BY SRS CU/DU IMPLEMENTATION. 131

FIGURE 50: EXCERPT OF THE SRS GNB CONFIGURATION FILE, FOCUSING ON NTN PARAMETERS [69]. 133

LIST OF TABLES

TABLE 1: BIT RATE REQUIREMENTS FOR DIFFERENT TRANSMISSION BANDWIDTH..... 24
TABLE 2: COMPARISON BETWEEN TERRESTRIAL O-RAN AND NON-TERRESTRIAL O-RAN MODELS..... 24
TABLE 3: SUMMARY OF COMPARATIVE CAPEX FOR O-RAN NTN ARCHITECTURAL SPLITS 34
TABLE 4: SUMMARY OF O-RAN BASED SPLIT DESIGN OPTIONS FOR INTEGRATED NTN 39
TABLE 5: CATEGORIZATION OF APPLICATIONS BASED ON THEIR PRIMARY FUNCTION IN THE CONSIDERED DATASET. 58
TABLE 6: COMPARISON OF FEDERATED LEARNING MODELS: FED-KAN VS. FED-MLP IN THE CONSIDERED NTN SCENARIO. 60
TABLE 7: SUMMARY OF THE CONSIDERED BEAMFORMING ALGORITHMS AND POWER DISTRIBUTION STRATEGY. 77
TABLE 8: SIMULATION PARAMETERS FOR CA-PQS. 99
TABLE 9: CONFIGURATION PARAMETERS FOR THE CA-PQS ALGORITHM. 99
TABLE 10: CLUSTERING PARAMETERS FOR THE CA-PQS ALGORITHM. 100
TABLE 11: OPTIMAL THRESHOLD VALUE WITH HAC-3 CLUSTERING AND GEOGRAPHICAL SCHEDULING. 100
TABLE 12: OPTIMAL THRESHOLD VALUE WITH HAC-3 CLUSTERING AND CSI SCHEDULING. 100
TABLE 13: OPTIMAL THRESHOLD VALUE WITH SPECTRAL CLUSTERING AND CSI SCHEDULING. 100
TABLE 14: OPTIMAL THRESHOLD VALUE WITH DBSCAN CLUSTERING AND GEOGRAPHICAL SCHEDULING. 100
TABLE 15: OPTIMAL PERFORMANCE OF THE BENCHMARK PQS ALGORITHM WITH GEOGRAPHICAL SCHEDULING CONSTRAINT. 101
TABLE 16: OPTIMAL PERFORMANCE OF THE BENCHMARK PQS ALGORITHM WITH CSI SCHEDULING CONSTRAINT. 101
TABLE 17: COMPUTATIONAL COMPLEXITY FOR SINR ESTIMATION. 112
TABLE 18: SIMULATION PARAMETERS FOR DMHSA TRAINING..... 114
TABLE 19: TRAINING PARAMETERS FOR DMHSA. 115
TABLE 20: ADDITIONAL PARAMETERS FOR DMHSA TESTING WITH PQS SCHEDULING. 117
TABLE 21: SINR ESTIMATION RMSE WITH PQS SCHEDULING..... 118
TABLE 22: EARTH MOVING BEAMS VS EARTH FIXED BEAMS COMPARISON 120
TABLE 23: TYPICAL CPU, RAM, AND STORAGE SPECIFICATIONS 123
TABLE 24: PROJECTED CHARACTERISTICS IN TERMS OF CPU, RAM AND STORAGE OF NEXT GENERATION SATELLITES 124

ABBREVIATIONS

5GC	5G Core network
AI	Artificial Intelligence
AMF	Access Mobility Function
ARQ	Automatic Repeat Request
BTW	Bit-Width
CA-PQS	Clustering Aided PQS
CDN	Content Distribution Network
CN	Core Network
COTS	Commercial Off-The-Shelf
CPRI	Common Public Radio Interface
CPU	Central Processing Unit
CR	Control Signalling Rate
CRC	Cyclic Redundancy Check
CSI	Channel State Information
CU	Central Unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DL	Deep Learning
DRL	Deep Reinforcement Learning
DU	Distributed Unit
E2SM	E2 Service Model
eCPRI	Enhanced Common Public Radio Interface
FCAPS	Fault Configuration Accounting Performance Security
Fed-KAN	Federated Learning with Kolmogorov-Arnold Networks
Fed-MLP	Federated Learning with Multi-Layer Perceptrons
FFT	Fast Fourier Transform
FL	Federated Learning
FPGA	Field-Programmable Gate Array

FR1	Frequency Range 1
FR2	Frequency Range 2
GEO	Geostationary Earth Orbit
GPU	Graphics Processing Unit
GW	Gateway
HARQ	Hybrid Automatic Repeat Request
HO	Hand Over
IFFT	Inverse Fast Fourier Transform
IP	Internet Protocol
ISL	Inter-Satellite Link
KAN	Kolmogorov-Arnold Network
LDPC	Low Density Parity Check
LEO	Low Earth Orbit
MLP	Multi-Layer Perceptrons
MAC	Medium Access Control
MEO	Medium Earth Orbit
ML	Machine Learning
MSE	Mean Squared Error
Near-RT	Near-Real-Time
NG-RAN	Next Generation Radio Access Network
NFV	Network Functions Virtualization
Non-RT	Non-Real-Time
OBP	On-Board Processing
O-CU	O-RAN Central Unit
O-CU-CP	O-RAN Central Unit - Control Plane
O-CU-DP	O-RAN Central Unit - Data Plane
O-DU	O-RAN Distributed Unit
OFDM	Orthogonal Frequency Division Multiplexing

OFH	Open Fronthaul Interface
O-RAN	Open Radio Access Network
PAPR	Peak-to-Average-Power Ratio
PCA	Principal Component Analysis
PDCP	Packet Data Convergence Protocol
PDU	Protocol Data Unit
PHY	Physical Layer
PoC	Proof-of-Concept
PQS	Priority Queue Scheduler
PRB	Physical Resource Block
QoS	Quality of Service
QoE	Quality of Experience
RACH	Random Access Channel
RAN	Radio Access Network
RIC	RAN Intelligent Controller
RLC	Radio Link Control
RMSE	Root Mean Squared Error
RRC	Radio Resource Control
RRM	Radio Resource Management
RT	Real-Time
RTT	Round-Trip Time
RU	Radio Unit
SBA	Service Based Architecture
SCS	Subcarrier Spacing
SDAP	Service Data Adaptation Protocol
SEC	Space Edge Computing
SIB19	System Information Block 19

SNR	Signal-to-Noise Ratio
SMF	Session Management Function
SMO	Service Management and Orchestration
SNR	Signal-to-Noise Ratio
SNO	Satellite Network Operator
UE	User Equipment
UPF	User Plane Function

1 INTRODUCTION

This deliverable D4.7 reports the outcomes of Task 4.3 “Data-enhanced Radio Intelligent Controller Design”, Task 4.4 “Onboard Processing and Communication Capabilities” and Task 4.5 “RAN Softwarisation”. The activities undertaken within the scope of these tasks, pursue the objective to:

- Tailor O-RAN architecture to non-terrestrial network (NTN) scenarios;
- Define, design and analyse data-driven management system components;
- Study, design, and analyse a 5G-based satellite network, implementing onboard processing and storage capabilities;
- Define, design, and analyse a full softwarisation of the end-to-end network architecture.

Section 2 presents a taxonomy of architectural and functional split strategies for Open Radio Access Network (O-RAN)-enabled NTN systems. The section analyzes key trade-offs in performance, latency, and autonomy by evaluating configurations that distribute RAN and core functions between satellites and ground nodes, from onboard Distributed Unit (DU) deployments to full gNB and user plane function (UPF) integration. The placement of near-real time (RT) and non-RT RAN intelligent controllers (RICs) is also explored, with flexible strategies proposed to optimize control loop performance and scalability. A comprehensive mapping is provided between architectural splits and RIC placement options, highlighting implementation constraints and interoperability.

Section 3 presents artificial intelligence (AI) models to predict traffic loads in NTN environments. The traffic forecasts are used to perform radio resource allocations on a time scale from seconds to minutes. The solution proposed in Section 3.1 relies on probabilistic forecasting. This section highlights the benefits of the proposed approach with respect to a deterministic model, to reduce under-provisioning and reduce the over-provisioning of radio resources. In Section 3.2 federated learning (FL) is proposed as a promising alternative that enables decentralized training while maintaining data privacy. Existing FL models, such as federated learning with multi-layer perceptrons (Fed-MLP), can struggle with high computational complexity and poor adaptability to dynamic NTN environments. To enhance the traditional methods, this section provides a detailed analysis for federated learning with Kolmogorov-Arnold networks (Fed-KAN) for traffic forecasting. Section 3.3 focuses on exploiting high-dimensional interactions across beams. In this context, multivariate AI-based prediction techniques for RRM in multi-beam satellite systems are evaluated against traditional single-beam prediction methods. The results highlight the substantial advantages of multivariate AI models, demonstrating their potential to enhance RRM efficiency and reliability in next-generation satellite communication networks.

Section 4 discusses the AI/ML-aided solutions for scheduling in user centric beamforming applications and SINR estimations. Section 4.1 reports the main system and architecture assumptions that hold throughout this section, based on the technical work in WP3 and WP4. Section 4.2 is dedicated to user-centric beamforming, with the rationale for this application of AI and the design of clustering-based scheduling. The performance assessment provides the comparison with non-AI-aided solutions detailed in [2]. Section 4.3 reports the rationale for the application of NNs to SINR estimation, and the design and assessment of a neural network that can be exploited for this task. It shall be noticed that this tool can be exploited for the designed user-centric beamforming scheduling or other algorithms that would benefit from SINR estimates.

Section 5 analyses the OBP capabilities of regenerative payloads. The section categorizes satellites into three types: traditional telecommunication satellites, hybrid satellites supporting telecommunication and edge computing and satellites primarily designed for edge computing. For each, the section explores current state-of-the-art CPU, RAM, and storage configurations, supported with real-world examples. For the network architectures introduced in Section 2, the functions and the network elements that are allocated on-board of the satellite are analyzed from the resource consumption point of view.

Section 6 describes the general stack modifications that are needed to the PoC implementation. This includes SIB19 generator, support of two NTN HARQ management approaches and support of feeder link. New functionalities have been added to enable the interconnection of the CU and the DU with near-RT RICs and support the remote and automated configuration of the gNB.

Finally, Section 7 concludes this document.

2 ARCHITECTURAL SPLIT AND RADIO INTELLIGENCE CONTROLLER PLACEMENT

This section presents an extensive analysis of various architectural splits to enable the adoption of O-RAN-based architecture in NTNs, focusing on the specific challenges and constraints inherent to satellite platforms. It systematically evaluates the advantages and disadvantages of each split option in the context of NTNs, providing insights into their feasibility, performance, and implementation complexity and open challenges. Furthermore, this section conducts a thorough investigation into the placement of the RIC components within O-RAN-based NTNs, analyzing the trade-offs, challenges and network scalability. Through this analysis, it offers a comprehensive understanding of the design considerations for O-RAN integration in NTNs. The work is published in [3].

In particular, the key contributions of this section are the following:

- Taxonomy of architectural O-RAN splits for NTN: Proposes a detailed taxonomy of architectural and functional split options in line with O-RAN principles based on the available capabilities in the spaceborne segment.
- Systematic evaluation of split configurations: Analyzes a wide range of architectural split scenarios for O-RAN NTN, from ground-based processing to fully onboard deployments (e.g., gNB and UPF on satellites), including intra- and inter-satellite processing strategies.
- Trade-off analysis: Provides a comprehensive analysis of the trade-offs associated with different architectural splits in terms of performance, latency, autonomy, scalability, and deployment complexity.
- RIC placement strategies: Investigates optimal placement strategies for Near-RT and Non-RT RAN Intelligent Controllers in space-ground split environments, addressing control loop efficiency, computational requirements, robustness to intermittent feeder link conditions, persistent ground access, scalability and open challenges (e.g., the need for new interfaces, limitations, requirements, etc).
- Mapping of architectural splits and RIC placement: Delivers a novel mapping between architectural split options and RIC deployment alternatives, emphasizing implementation feasibility and interoperability within O-RAN-based NTNs.
- Design guidelines for O-RAN in NTNs: Offers practical insights and design considerations for implementing modular, efficient, and standardized O-RAN architectures in satellite networks.
- Identification of NTN-specific challenges and future research directions: Highlights the unique constraints of NTNs (e.g., dynamic topologies, limited onboard processing, heterogeneous links) that impact O-RAN integration and architectural design. Furthermore, we outline open challenges and future research areas necessary to advance TN-NTN convergence and ensure effective O-RAN adoption in non-terrestrial contexts.

2.1 O-RAN ARCHITECTURE OVERVIEW

5G has opened the door for a revolution in the architecture and management of mobile networks. With the introduction of the service-based architecture (SBA) concept at the mobile core, mobile networks are leaving behind closed architectures. Mobile network components are no longer monolithic blocks employing dedicated protocols for direct communications among entities. This trend is also followed in the RAN segment through the propositions of the Open RAN (O-RAN) alliance [4]. It promotes openness, intelligence, and interoperability by the disaggregation of traditional monolithic base station architecture into flexible, virtualized components connected by standardized interfaces and naturally including closed-loop operations enhancing network performance. In this way, the proposed model fosters vendor-neutral deployments, avoiding lock-in situations to operators, thus improving cost-efficiency and flexibility to adapt network deployments to meet the requirements of challenging use cases proposed by vertical industries. More specifically, the foundations of O-RAN are based on four pillars:

- Disaggregation: the gNB is decoupled into three logical units, namely RU, DU, and CU. This allows operators to upgrade or replace hardware and software independently.
- Open Interfaces: all functional splits between entities (e.g., between the RU, DU, and CU) are standardized, enabling multi-vendor deployments.
- Cloud-Native Virtualization: O-RAN enables RAN functions to run on COTS hardware, leveraging network function virtualization (NFV) and containers.
- Intelligence and Automation: O-RAN proposes the use of RICs to support AI/ML-driven optimization and dynamic control at different timescales.

Based on these benefits, it is reasonable that in future 6G networks, chasing interoperability between NTN and TN segments, the NTN components shift towards such O-RAN paradigm to provide the required flexibility in terms of network management and optimization. Next subsection presents the main entities and interfaces proposed by current O-RAN specifications, designed for the TN segment, as depicted in Figure 1. Later, we sketch the potential challenges are sketched, which motivates the architectural discussions for different functional split and RIC placement for the NTN segment in the following sections. As observed in Figure 1, with shaded rectangles, the adoption of O-RAN paradigm in NTN will have a great impact in the current O-RAN architecture.

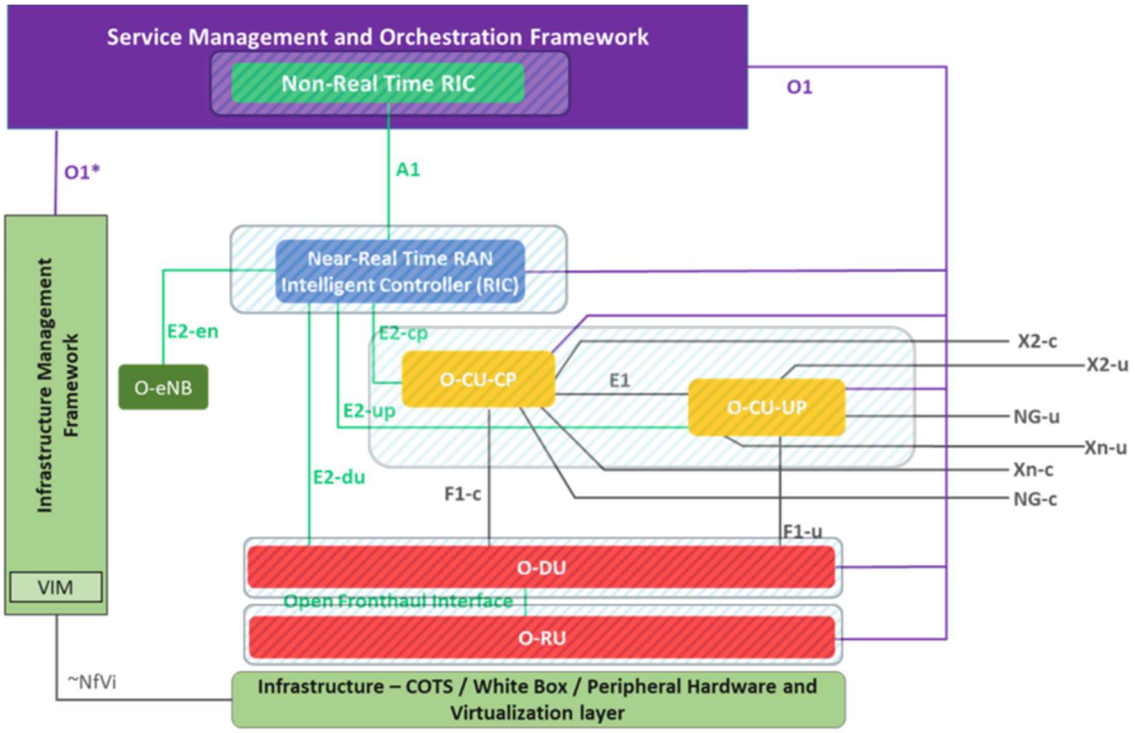


Figure 1: Generic O-RAN architecture with interfaces between functional and intelligence components [4].

O-RAN Functional Entities and Interfaces

Figure 1 presents the generic RAN architecture proposed by O-RAN Alliance specifications [4]. The first thing to notice is the disaggregation of the gNB into three different network functions, as previously mentioned, thus allowing network management multiplexing gains due to distributed deployments. These functions are:

- O-RU: manages the radio front-end and lower PHY layer operations (e.g., FFT/IFFT, beamforming) and transmits signals to the UEs.
- O-DU: handles real-time lower-layer protocol processing functions (e.g., Layer 1/2 MAC, RLC), such scrambling, modulation, and resource element mapping, close to the O-RU.
- O-CU: Executes non-real-time higher-layer protocol processing function (e.g., PDCP, SDAP) which are further separated into control and user plane functions, often deployed in centralized data centres.

At the Intelligence layer, the O-RAN architecture proposes the use of RICs, namely:

- Near-Real Time (RT) RIC: offers control functions such as load balancing, mobility management, and interference mitigation in near-real-time (10ms–1s). This is done through the use of xApps, which are custom logic applications responsible for collecting information, computing it, and triggering control actions to the O-RAN network functions.
- Non-RT RIC: policy-based guidance, analytics, ML model management collaborating with the Near-RT RIC in control-loops bigger than 1s. It hosts rApps application supporting RAN optimization and management tasks. It is placed within the Service Management and Orchestration (SMO) framework.

Finally, the SMO Framework enables the end-to-end lifecycle management (i.e., orchestration and FCAPS (Fault, Configuration, Accounting, Performance and Security) operations) of the different instances of gNB's O-RAN network functions and RICs.

Regarding the interfaces, O-RAN specifications include the following ones:

- Open Fronthaul (OFH) Interface: based on the functional split 7.2x proposed by 3GPP, it is designed to strike a balance between the simplicity of the O-RU and the data rates and latency requirements on the interface between the O-DU and O-RU.
- F1: defined by 3GPP, connects the O-CU and O-DU to exchange control and user plane traffic.
- E2 Interface: Enables the Near-RT RIC to collect data from the O-RAN network functions (i.e., O-CU and O-DU) and to provide actions.
- E1 Interface: defined by 3GPP, E1 enables independent scaling and management of the control and user plane functions within the O-CU.
- O1 interface: Enables the SMO to perform FCAPS of O-RAN network functions and the Near-RT-RIC.
- A1 interface: employed by the Non-RT RIC to provide policies, enrichment information and ML models to the Near-RT RIC and collect feedback.

It is important to remark that the RAN disaggregation concept cannot be straightforwardly applied to NTN. The limitation essentially comes from the bit rate and the latency requirements of fronthaul and midhaul networks, which connect RU-DU and DU-CU, respectively. Some details are provided hereinafter following the methodology proposed in [5]. The downlink and uplink fronthaul bit rate is defined by

$$BR = M \times N \times L \times BTW \times \frac{2}{T_s} + CR.$$

The bit rate is based on the bit-width (BTW) which is the number of I and Q bits, the number of subcarriers M, the number of symbols N, the number of layers L, the slot duration T_s and the control signaling rate (CR). The CR is determined by the overhead regarding beamforming, resource element mapping and scheduling. For 20 MHz bandwidth, 64-QAM and 2 layers, the resulting rate is CR=1.856 Mbps. The rate linearly scales with the bandwidth, the modulation order and the number of layers. Accordingly, the rate for different system parameters can be easily obtained from the reference value. The maximum latency for a fronthaul network is limited to 500 μs.

The downlink and uplink midhaul bit rate is given by

$$BR = \mathrm{PR} + \mathrm{CR},$$

where PR denotes the peak rate. The control and signaling rate CR is calculated as 24 Mbps and 16 Mbps in downlink and uplink, respectively. For 20 MHz bandwidth, 16-QAM and 1 layer, the peak rate in uplink is PR=50 Mbps. In the downlink, for 20 MHz bandwidth, 64-QAM and 2 layers, the peak rate is PR=150 Mbps. The parameter CR is assumed to be constant, while the peak rate PR linearly scales with the bandwidth, the modulation order and the number of layers. Accordingly, the rate for different system parameters can be easily obtained from the

reference value. The maximum latency for a midhaul network is between 1.5 and 10 ms. Since the latency is not limited by the HARQ procedure, thus the midhaul segment tolerates high latency. The bit rate requirements are computed in Table 1. As for the air interface, the following assumptions have been made: i) the cell is fully loaded, ii) the bit rate is computed in the downlink, iii) there are 14 OFDM symbols per slot, iv) I/Q samples are represented with 14 bits, v) there is one layer and one antenna port and vi) modulation order up to 64-QAM.

Table 1: Bit rate requirements for different transmission bandwidth

Bandwidth	100MHz	200MHz	400MHz
Subcarrier spacing	60 KHz	60 KHz	120 KHz
Number of PRBs	132	264	264
Fronthaul bit rate (Gbps)	2.48	4.97	9.96
Midhaul bit rate (Gbps)	0.399	0.774	1.524

As expected, the fronthaul network demands more resources than the midhaul network. The reason lies in the fact that the interface shall transport samples. Conversely, the midhaul network calls for more relax bit rate requirements.

Considerations and Challenges for extending O-RAN to NTNs

Unlike terrestrial O-RAN deployments, where stable fibre-based fronthaul, dense edge infrastructure, and powered nodes are assumed, NTN scenarios introduce fundamentally different constraints. NTNs, focusing on LEO/MEO/GEO satellites are integral to 5G Advanced and 6G networks for providing global and ubiquitous coverage, but require architectural adaptations from the generic architecture considered for the TN segment to address their unique operating environments. These include integrating distributed applications (dApps) or space-native applications (spaceApps) [6] and redefining functional splits between satellites and ground stations.

Table 2: Comparison Between Terrestrial O-RAN and Non-Terrestrial O-RAN Models

Aspect	Terrestrial O-RAN Models	Non-Terrestrial O-RAN Models (NTNs)
Topology and Coverage	Dense deployment of fixed sites with overlapping coverage, stable cells, and limited topology changes.	Highly dynamic topology due to satellite movement (especially in LEO/MEO), global coverage including remote and maritime areas.
Latency and Timing	Low latency (<5 ms) enabled by fiber-based fronthaul and midhaul; deterministic synchronization.	High propagation delays: ~5–20 ms for LEO, ~50-100 ms for MEO, >250 ms for GEO; difficult synchronization across interfaces (e.g., F1, E2).
Link Stability	Fronthaul/Midhaul connections via stable terrestrial fibre or microwave links.	Frequent feeder link outages and Doppler-induced channel variability; requires adaptive link management and buffering mechanisms.
Functional Split Feasibility	Fine-grained splits (e.g., Option 7.x) feasible due to low-latency fronthaul links; standardized deployment widely adopted.	Coarser splits preferred (e.g., Option 2, Option 5) to minimize fronthaul timing constraints; requires adapting O-RAN architecture for satellite links.
RIC Placement	Near-RT RICs typically co-located in edge clouds; non-RT RICs centralized within data centres.	Distributed or hybrid RIC architectures required to handle intermittent connectivity and topology changes; may

Aspect	Terrestrial O-RAN Models	Non-Terrestrial O-RAN Models (NTNs)
		co-locate near-RT RICs onboarded in satellites.
Processing and Energy Constraints	Terrestrial nodes are powered and can host high-capacity compute for AI/ML workloads.	Satellites have limited onboard compute, storage, and energy; algorithms must be lightweight and power-aware.
Interoperability	Seamless integration with terrestrial 5G Core networks via standardized interfaces.	Requires tighter integration between NTN and TN domains to ensure session continuity, dynamic routing, and unified resource management.
Security and Reliability	Mature security mechanisms supported by stable infrastructure and private backhaul networks.	Increased attack surface due to open O-RAN interfaces, regenerative payloads, and distributed space-ground functions; anti-jamming and robust encryption are critical.

Table 2 summarizes the key differences between terrestrial and non-terrestrial O-RAN models, highlighting the architectural adaptations required for NTN environments. While O-RAN architectures have been extensively studied in terrestrial settings, their direct application to non-terrestrial environments is non-trivial. NTN deployments face inherently different constraints, such as long propagation delays (especially in GEO), limited onboard energy and processing capabilities, dynamically varying topologies (as in LEO constellations), and dependency on feeder link availability. These differences critically impact the feasibility and performance of various functional split options. For example, Option 7.x - widely adopted in terrestrial O-RAN - relies on tight fronthaul timing synchronization, making it unsuitable for high-latency satellite links. Hence, functional split strategies and RIC placement decisions must be revisited to accommodate NTN-specific requirements.

We discuss these splits in the following sections, considering the following key challenges:

- Latency and Timing Constraints: NTN links introduce propagation delays (e.g., 5-20 ms for LEOs, larger than 250 ms for GEOs), which have impact on control-plane signalling and real-time interfaces like E2 and F1. Buffering, synchronization, and jitter handling mechanisms must be adapted.

- Doppler and Link Variability: Satellite movement leads to Doppler shifts and dynamic link changes (e.g., unstable performance of feeder link), which are not considered in current O-RAN PHY/MAC-level standards. This demands for enhanced PHY designs and scheduler algorithms.

- Interface and Protocol adaption: O-RAN interfaces assume consistent, low-latency terrestrial links. In NTN, protocols must be enhanced with mechanisms for delay tolerance, retransmission, and session persistence. In addition to this, for non-GEO, visibility with the gateway (GW) on ground can change, thus introducing additional routing considerations to exchange data between satellites running O-DUs and O-CUs and near-RT RICs. Indeed, for non-GEO, the continuous movement of the gNB introduces an additional dimension to the handover problem and its connections towards entities running on ground. It also advocates for distributed and synchronized architectures of near-RT RICs and non-RT RICs.

- Interoperability with TNs: TNs and NTNs must share spectrum, mobility anchors, and user session management. This requires tight integration between 3GPP Core networks, SMO, and both RICs to support dynamic routing and resource orchestration.

- Security and Reliability Risks: the openness of O-RAN interfaces, software-based deployments and distribution of the network functions in regenerative satellite payload expands the attack surface raising concerns around encryption, jamming, and system integrity. This can be extremely critical for defence-communications.

- Hardware, computational constraints and CAPEX: Although advancements in the expenses associated with the deployment of satellite constellations and satellite manufacturing, such deployments are still expensive and require an accurate preparation. Moreover, satellite operates under strict power, processing capacity and thermal management conditions, thus introducing fluctuations in the operations they can handle based on the onboarded hardware equipment (e.g., CPUs, RUs, feeder link and ISL modems), including possible reconfiguration of network functions, and algorithms they can host and run.

When integrating NTNs into 5G architectures, the deployment of gNB components in the space and terrestrial segments is crucial. In the following, three key options for functional split are discussed, with increasing levels of satellite autonomy and complexity. It is worth mentioning that this increase in complexity will impact in the CAPEX associated to the satellite deployment, where the cost is directly associated to the available computational load and the communications needs. In that sense, these costs will be affected by the cost of onboarded processing units like CPUs, GPUs, FPGAs or other specialized hardware to execute RAN, CN functions or even vertical applications, RUs elements, feeder link modems and ISL hardware, which will shape the possible capabilities of the satellite constellation.

2.2 OPTION 1: DU AND RU ONBOARD (SPLIT 2)

In this approach, the DU of the gNB is deployed onboard the satellite, while the CU remains on the ground. This corresponds to the Split 2 architecture in the 5G RAN, in which the satellite handles physical layer functions such as modulation, channel coding and real-time scheduling, as well as procedures such as the random access, the hybrid automatic repeat request HARQ retransmission scheme and the system information block 19 (SIB19) generation. The schematic view of the architecture is depicted in Figure 2, where a monolithic DU-RU setup within a single satellite, and a distributed setup with the DU and RU network functions in different satellites are depicted.

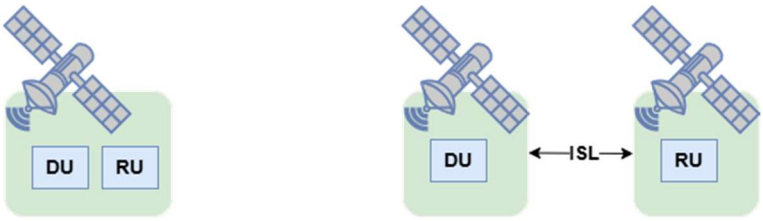


Figure 2: Deployment options for DU onboard. Left: DU and RU in single satellite. Right: DU and RU interfaced over ISL.

The main advantage of this setup is the reduction of latency for time-sensitive PHY/MAC functions. However, there are also limitations with this architecture:

- The CU remains on Earth, limiting responsiveness and dependence on terrestrial links.

- Application-layer services and edge computing capabilities are not feasible onboard.

Regarding inter-satellite mobility, when a UE moves from the source to the target satellite, two types of procedures can be differentiated. More precisely, mobility is handled as either intra- or inter-CU handover, depending on whether the source and target satellites are connected to the same CU or to different CUs, respectively. A specific characteristic of the intra-CU handover is that it does not require involvement of the core network, as shown in [7]. The two topology options of Figure 2 are described hereinafter.

2.2.1 Option 1a: DU in single satellite

In this configuration the satellites of the constellation are identical, each hosting the DU and the RU network functions. Moreover, each satellite must establish feeder links to connect the onboard DU with the CU, which is co-located to the satellite gateway. Therefore, the F1 interface shall be implemented over the feeder link. It is important to remark that the F1 interface requires a persistent connection. For non-GEO constellations, this means that satellites must be able to maintain two feeder links, while the connection is transferred from the source to the target gateway. The support of variable latency and seamless handovers becomes the primary challenges of the feeder link.

From the signalling point of view, it is worth highlighting that the radio resource control (RRC) would reside in the CU. Hence, CU-DU interactions over F1 interface will include the procedures detailed in [8]. Consequently, signalling requirements must be carefully considered to ensure proper dimensioning of the feeder link. During a feeder link switchover, the DU should handover all the UEs under the satellite coverage. Remarkably, this procedure requires a very high signalling load.

2.2.2 Option 1b: DU and RU in different satellites

The option 1b, offers a more distributed architecture in which satellites are not identical. In this regard, there are two classes of satellites to enable RAN disaggregation. Class 1 satellites are equipped with a DU and include feeder links. The RUs are implemented in class 2 satellites. The fronthaul interface that connects the DU and the RU is implemented over an ISL. In this deployment, class 1 satellites include four permanent ISLs, two connecting to adjacent satellites in the same orbital plane and two connecting to adjacent orbital planes. Therefore, the DU is connected to four RUs, which are located in different satellites.

The primary technical challenge of this deployment arises from the round-trip time (RTT) delay between adjacent satellites, which is significantly larger than typical RU-DU delays encountered in terrestrial networks. Remarkably, adjacent satellite could be separated hundreds of kilometres, or even up to a thousand kilometres. The dominant technology in terrestrial deployments to interface the DU and the RU is the common public radio interface (CPRI). However, the specification reported in [9] highlights that this technology is not suitable to transport digitized signals over ISLs. The limitation is that the maximum one-way delay supported by eCPRI is $500\mu s$. The RTT delay in most ISLs exceeds the maximum value specified by eCPRI. This observation highlights the necessity of using a technology specifically tailored to NTN.

2.3 OPTION 2: FULL GNB ONBOARD

In this configuration, the complete gNB stack comprising the RU, DU, and CU is deployed onboard the satellite. This setup eliminates the need for fronthaul and midhaul interfaces between terrestrial entities and reduces the dependence on Earth-based control and scheduling. It corresponds to a fully autonomous gNB entity [10, 11] according to 3GPP NR architecture, with N2/N3 interfaces directly connecting the satellite to the 5G CN through feeder

links [12]. By co-locating the entire RAN protocol stack onboard [13], the system executes the complete sequence of 5G RAN functions:

- The physical layer performs FFT/IFFT operations, LDPC/BCH encoding, cyclic prefix addition and beamforming.
- The MAC and RLC layers execute time-critical functions such as resource scheduling, HARQ retransmissions and radio bearer reconfiguration.
- The RRC and SDAP layers handle control signalling, mobility procedures, bearer establishment, and QoS mapping.

The onboard integration removes the stringent timing requirements of midhaul and fronthaul interfaces, which typically cannot be met in NTN environments due to propagation delays.

A core benefit of this option lies in the simplification of RAN-5G CN integration. As the full gNB stack is onboard, the satellite only requires the N2 and N3 interfaces to connect with the 5G CN via feeder links, avoiding complex control synchronization with Earth-based CUs. This reduces signalling overhead and allows for fast UE context management [11, 14], especially relevant for intra-satellite mobility and session continuity. Moreover, by embedding all scheduling and RRC procedures locally, this approach enhances autonomy and responsiveness, particularly in cases where gateway visibility is intermittent or disrupted. Additionally, mobility anchoring and bearer management are performed on orbit, which allows for seamless UE handovers at the gNB level, reducing the signalling burden on the CN. These advantages make Option 2 especially attractive for NTN systems requiring low-latency access and mobility robustness.

However, the absence of an onboard user plane function (UPF) introduces important limitations. As the N6 interface is not terminated in space, no localized breakout or protocol data unit (PDU) level processing is feasible. All user-plane traffic must be routed to the ground-based UPF, increasing the load on feeder links and introducing additional latency for application-level services. Consequently, Option 2 does not support caching, content delivery network (CDN) integration, or space edge computing (SEC) capabilities. This lack of edge functionality renders the architecture suboptimal for data-intensive or real-time applications that would benefit from local service exposure. Additionally, the integration of the full gNB stack onboard significantly increases power consumption and processing requirements. Satellites must be equipped with high-performance, onboard processing units capable of handling real-time PHY/MAC tasks, ciphering and control signalling, which may be infeasible for small satellite platforms with limited power budgets.

While this configuration reduces timing constraints on the feeder link, it still demands persistent, high-throughput N2/N3 connectivity to the core network. For non-GEO constellations, the dynamic motion of satellites requires robust mechanisms for gateway reassignment and session re-anchoring to ensure continuity. A potential solution for managing these feeder link handovers is the NG-Flex configuration specified in 3GPP TS 38.410 [15]. In this setup, each Next Generation-RAN (NG-RAN) node can be connected to multiple access and mobility function (AMF) sets within an AMF Region, as long as the corresponding slice is supported. This architecture, complemented by definitions in TS 23.501 [16], allows the onboard gNB to switch among ground-based AMFs without requiring session reestablishment, thus simplifying mobility anchoring across gateways. This makes NG-Flex particularly suitable for Option 2 deployments with full gNB onboard, where feeder link transitions occur frequently due to satellite motion. Moreover, managing IP mobility, synchronizing bearer contexts and maintaining QoS guarantees under variable link conditions remain open research challenges. Furthermore, Option 2 provides an effective compromise between latency performance and

architectural complexity, offering a high degree of autonomy and fast control-plane responsiveness while remaining compatible with current gNB/CN interface standards. Nevertheless, its lack of edge-service support and the elevated onboard processing requirements necessitate careful design trade-offs. It is best suited for medium-class LEO or MEO satellites capable of hosting the complete RAN stack but not yet supporting full edge computing. This configuration is particularly applicable to scenarios prioritizing mobile broadband connectivity, efficient mobility handling and simplified terrestrial integration without the additional complexity of local user-plane functions. To accommodate varying architectural constraints and mission objectives, two distinct deployment variants of the full onboard gNB can be considered, as illustrated in Figure 3. Option 2a, shown on the left of the figure, implements a monolithic integration of the CU, DU, and RU within a single satellite platform. In contrast, Option 2b, which is depicted on the right, distributes these functional blocks across multiple satellites interconnected via ISL links, thus supporting fronthaul or F1-C/U traffic depending on the split.

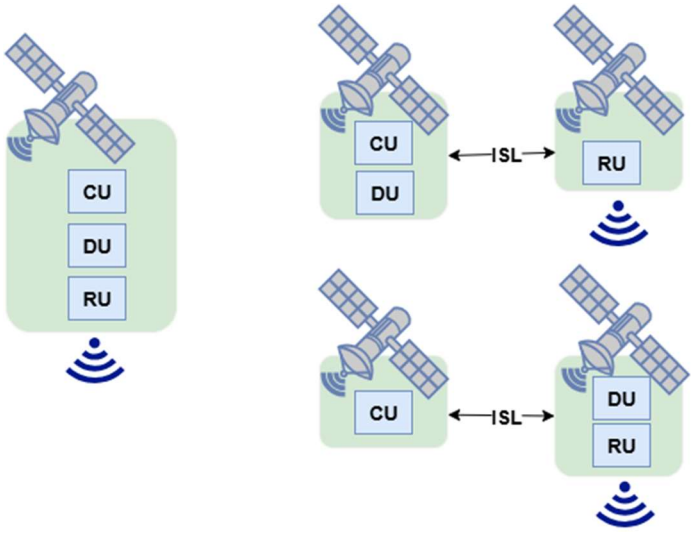


Figure 3: Deployment options for full gNB onboard. Left: Option 2a (monolithic gNB). Right-top: Option 2b with CU and DU together and RU separated over ISL. Right-bottom: Option 2b with DU and RU together and CU separated. ISLs support fronthaul or F1-C/U, depending on split.

2.3.1 Option 2a: Monolithic gNB

In this variant, the entire gNB stack is fully embedded within a single satellite platform. This monolithic integration simplifies internal communication and removes the need for any inter-satellite synchronization or high-speed link coordination. All intra-gNB interfaces (e.g., fronthaul, midhaul) become local, effectively eliminating latency variability caused by long-distance physical separation between components. This is particularly beneficial for timing-sensitive processes like HARQ feedback, MAC scheduling, and random access channel (RACH) response handling. Since all functions share the same physical platform, coordination is inherently tight, and software integration can leverage shared memory or fast bus-level interconnects, achieving minimal processing delay.

This approach is the most straightforward to implement from both a protocol and hardware standpoint. It reduces system complexity, limits the risk of synchronization errors, and facilitates thermal co-design of processing units. Furthermore, it simplifies RAN management and orchestration as the entire node can be managed as a single logical and physical entity. Deployment, provisioning, and fault recovery operations are also more contained and deterministic. Because intra-satellite delays are negligible (on the order of microseconds), real-time control loops such as channel state information (CSI) reporting, scheduling decisions and

HARQ processes can function with latencies comparable to those in terrestrial networks. However, this configuration imposes a non-trivial burden on the satellite OBP, which must support the entire gNB protocol stack, including high-rate baseband signal processing and real-time scheduling decisions. Depending on the supported bandwidth, number of antenna ports and expected user load, the satellite must be provisioned with high-performance CPUs, possibly accompanied by FPGAs or GPUs for acceleration of PHY layer operations. Further details are provided in Section 5. The resulting power, mass, and thermal requirements may exceed the capabilities of small LEO platforms, suggesting this architecture is more suitable for medium or large LEO/MEO satellites with adequate payload capacity.

Another operational consideration is that the satellite must maintain continuous or quasi-continuous feeder link visibility with at least one gateway, as both the N2 and N3 interfaces require persistent connectivity to the 5G CN. Gateway handovers must be managed carefully to prevent session interruption, requiring dual gateway tracking or fast reattachment procedures. Nonetheless, the full co-location of the gNB stack allows these handovers to be more manageable compared to distributed setups, as there is no dependency on external RAN functions that could become unreachable during a feeder link handover. Consequently, Option 2a represents a baseline configuration for autonomous NTN RAN nodes and serves as a feasible intermediate step toward fully edge-enabled satellites in Option 3. Its simplicity and robustness make it an attractive choice for mission profiles where processing integration is preferred over architectural distribution and where multi-satellite coordination introduces unacceptable risk or complexity.

2.3.2 Option 2b: Distributed gNB Onboard

In contrast, Option 2b explores a distributed approach in which the components of the gNB stack, specifically the RU, DU, and CU, are spread across different satellites. For example, one satellite might host the CU and DU, while another, physically separated satellite hosts the RU. These components would then be connected via ISLs, forming a logical gNB node across a spatially distributed platform. The primary motivation for this approach is architectural scalability: by decoupling the RU and DU/CU roles across satellites, one can imagine a constellation where certain satellites specialize in radio front-end operations (analog/RF + lower PHY), while others aggregate digital baseband processing and control functions.

This modular separation allows for potential reusability of satellite types and dynamic reassignment of RAN functions across the constellation. For example, a group of RU satellites could beamform toward different ground coverage zones while a more powerful DU/CU satellite manages their scheduling and mobility procedures in a centralized manner. This supports flexible coverage shaping and resource pooling, especially in large constellations.

However, this distribution brings significant challenges. The most critical is the added delay and synchronization overhead introduced by the ISLs. Even with advanced ISLs operating at Gbps speeds and microsecond latencies under ideal conditions, the physical separation between satellites of typically hundreds of kilometres, results in non-negligible one-way delays. Thus, the standard fronthaul protocol stack is not directly applicable in this scenario, requiring custom transport protocols or enhanced synchronization techniques capable of compensating for larger propagation delays and jitter.

In addition, the physical mobility of satellites in LEO results in constantly changing inter-satellite topologies. Maintaining reliable ISL connectivity requires active routing and dynamic link-state awareness. When multiple RU satellites feed into a shared DU/CU satellite, load balancing, link failures, and variable traffic patterns must be handled in real time, necessitating advanced inter-satellite control-plane signalling and possibly distributed RIC instances. The need for synchronization of HARQ timers and CSI reporting becomes even more pronounced in this

context, potentially requiring predictive scheduling mechanisms or modified HARQ cycles that account for variable propagation delays.

Furthermore, placing the RU on one satellite and DU/CU on another breaks the symmetry of deployment and complicates resource management. While Option 2a allows a one-to-one mapping of resource blocks and UE sessions to a single platform, Option 2b requires distributed session context management and reassembly of transport blocks over ISLs, which introduces complexity and increases error propagation risk.

Despite these challenges, Option 2b holds strategic promise in advanced NTN systems with extensive satellite constellations and sufficient processing redundancy. It aligns with the principles of disaggregated RAN and may allow for better resilience and resource optimization when paired with intelligent routing and synchronization frameworks. However, realizing such an architecture will require significant innovation in both protocol stack design and satellite networking mechanisms, potentially involving a space-adapted fronthaul standard or custom MAC/RLC implementations.

A practical variant of Option 2b involves placing both the RU and DU on one satellite, while hosting the CU on another satellite with which it communicates via an inter-satellite F1 interface. This configuration avoids the stringent latency and synchronization constraints of fronthaul protocols and instead leverages the more relaxed tolerances of the F1-C interface. According to 3GPP guidelines and industry experience, F1-C, the control-plane interface between the DU and CU-CP, can tolerate one-way propagation delays in the range of approximately 2 to 10 milliseconds [17, 5], depending on vendor implementation and buffering strategy. Given that ISLs exhibit a typical one-way propagation delay of about 1.8 to 3.6 ms per hop, this allows for a maximum of 2 to 3 satellite hops between the DU and CU while remaining within acceptable latency budgets. This implies a proper ISL mesh planning, where a centralized CU satellite could serve a cluster of multiple RU+DU nodes, each placed within a low-hop-radius region of the constellation.

This architectural design enables the pooling of CU functions, such as RRC signalling, bearer management, QoS enforcement, and mobility coordination, across several RU+DU access satellites, improving control-plane scalability and resource utilization. Moreover, centralized CU deployment simplifies handover coordination and potentially allows for tighter integration with onboard or edge-deployed RIC instances. However, the inherent mobility and variable topology of LEO constellations still present challenges for session anchoring, F1 association maintenance, and scheduling cycles. These may require adaptive transport-layer mechanisms (e.g., reliable routing overlays), dynamic F1 rebinding protocols, and robust session context management across satellite transitions. Nevertheless, this RU+DU - CU split offers a practical and scalable compromise, particularly for NTN deployments targeting mid-density access scenarios with centralized control logic and moderate beam steering capabilities.

2.4 OPTION 3: FULL GNB ONBOARD WITH UPF FUNCTIONS

This option envisions a highly autonomous non-terrestrial node where the entire gNB (RU + DU + CU) and the UPF are fully integrated onboard the satellite payload. This architectural design, which is depicted in Figure 4, means the entire 5G NR base station and a 5G core user-plane node reside in space.

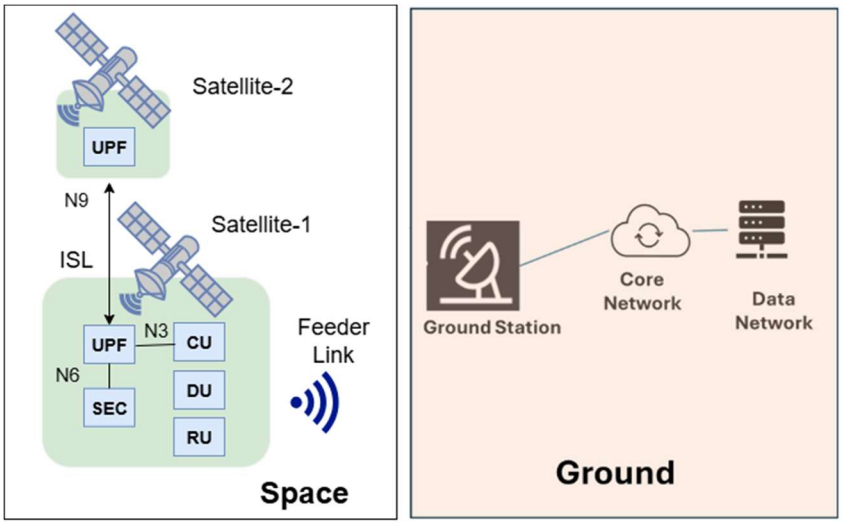


Figure 4: Option 3: Full gNB Onboard with UPF Functions

A recent white paper from the O-RAN Alliance also notes that future versions (3GPP Rel-19) will consider scenarios that include "a gNB on the satellite [...] with the UPF onboard" [17]. This means the entire 5G NR base station and a 5G core user-plane node reside in space. An ESA-funded 5G study likewise assumed "a complete gNB onboard the satellite" to meet NTN latency constraints [18]. In practice, demonstration projects have already proposed putting full gNB functionality on satellites, with ESA explicitly showing "Full gNB onboard with UPF might be also included [19]. This design supports local data breakout and reduced latency paths, particularly beneficial for edge-enabled services and delay-sensitive applications.

Placing a UPF in the satellite means standard 5G CN interfaces would traverse the satellite link. Notably, the N3 interface (gNB-CU user plane to UPF) becomes an internal link on the satellite, and the N4 interface (session management (SMF) to UPF) must operate over the satcom backhaul for control of the remote UPF. 3GPP's ongoing work in Release 18 explicitly addresses this: the study on Support of Satellite Edge Computing via UPF onboard was presented in the TSG SA WG2#152E Electronic meeting [20]. In essence, the network can deploy an onboarded UPF as a local data anchor, so that traffic can break out directly at the satellite via its N6 interface to local applications or caches (enabling "*satellite edge computing*"). The 3GPP study solutions in [21] allow an onboarded UPF to act as a local PDU session anchor, performing uplink classifier and branching for local routes. The authors in [22] examine the UPF (and by extension the N3/N4 signalling) allocation on satellites, using the N9 interface between the satellite-based UPFs. The paper in [23] demonstrates that satellite UPF deployment can reduce the energy consumption by 85.2% while maintaining comparable latency performance. At the same time, the paper in [24] find that cumulative conditional handover delay but demands 55%–70% more computational resources than the Split 7.2× architecture. Therefore, the full gNB onboard architecture was able to fulfil the strict requirements of ~50 ms delay for applications such as cloud gaming, while partial regenerative splits struggled to stay below 100 ms. This shows that low latency real-time services directly benefited from option 3. In addition, having the RAN and a portion of core co-located at the edge (or on the satellite) can handle local mobility management and local data path offload aligning with "ultra-flat, highly-distributed" architectures [18]. To summarise, standard 5G core interfaces (N2/N3/N4) can be extended via satellite links to manage an onboard UPF, and that the UPF's N6 interface can enable local breakout or edge computing applications in the satellite (e.g. content delivery or processing of data in the satellite itself).

As shown in the Figure 4, the onboard gNB consists of the full stack: the RU handles analog-digital conversion and low PHY (i.e., ADC/DAC, FFT/IFFT), while the DU processes MAC,

RLC, and high PHY functions such as modulation and coding. The CU manages both the control and user planes, interfacing with the onboard UPF over N3/N4 interfaces. Together, this enables complete gNB + UPF data path termination in space, decoupling the user plane from the terrestrial CN for most of the traffic. A key advantage of this architecture is its compatibility with N9-based mesh routing via satellites with ISLs. This allows packets to be forwarded between the satellites, reducing dependency on feeder links and enabling resilient and scalable routing across a satellite constellation. In addition, the presence of a fully operational UPF unlocks the N6 interface for integration with SEC platforms. These SEC platforms can host applications at the data plane (e.g. CDN caching, content filtering, analytics), enabling PDU-level processing in the satellite and service exposure. However, to realise this capability, significant onboard computing resources are required, e.g., dedicated CPUs and GPUs for packet routing, user processing and application execution.

Despite its advantages, this configuration also has significant disadvantages: (i) High computational demand necessitating powerful onboard hardware (e.g., CPUs, GPUs), payload complexity, and power consumption. These onboard processing units must be energy-efficient, radiation-tolerant and capable of handling significant data storage and a throughput of several Gbps. (ii) Increased system complexity and power requirements. Therefore, resource requirements and system complexity increase significantly, so thermal management, orchestration and fault tolerance are critical aspects of the design.

2.4.1 Option 3a: The same satellite

With regenerative satellites hosting UPFs, it becomes possible to form a mesh network in space. The satellites can route data traffic between each other at the user level, using ISLs as backhaul. In 5G terms, an N9 interface (the interface between two UPFs in the core network architecture) can be implemented via the ISL to route data between the UPFs onboard the satellites [22]. With option 3a, both the complete gNB stack (RU, DU, CU) and the UPF, including optional SEC functions, are hosted on the same satellite platform. This co-location ensures that all interfaces - including N2, N3, N4 and N6 - remain within the same satellite, enabling extremely low latency between functions and simplified synchronization between layers. The router, which is also onboard, supports ISL-based N9 interfaces for routing user traffic across the satellites when required, but no signalling between the satellites is required for local gNB-UPF interaction. This design is advantageous for latency-sensitive applications, real-time analytics and mission-specific services that can run completely isolated from ground control (e.g. tactical IoT clusters or content delivery in remote areas).

From an implementation perspective, Option 3a simplifies interface management, eliminates feeder-link dependence for most user traffic, and provides a self-contained edge computing node in space. However, it requires a powerful satellite payload, as the integration of gNB, UPF, routing logic and SEC functions on a single platform requires significant processing overhead (CPU/GPU), memory and I/O throughput, which can increase payload weight and thermal complexity. This option is suitable for multifunctional satellites with high payload integration, especially in LEO constellations where short revisit times and proximity to the user terminal increase onboard autonomy.

2.4.2 Option 3b: Different satellites

Option 3b considers the gNB and UPF functions distributed across different satellites - for example, a satellite hosting the complete gNB stack offloads user plane traffic to another satellite equipped with UPF and optionally SEC capabilities. Communication between these satellites takes place via ISL-based N9 interfaces, which enable dynamic user-plane routing in space. This setup supports functional disaggregation in the satellite domain and allows for specialisation of resources - some satellites focus on access (radio and RAN stack), while others are optimized for data processing, storage or application hosting.

This option allows for greater flexibility in the design of satellite constellations, as functions can be scaled independently and allocated dynamically based on mission requirements, geographic location of services or traffic load. For example, UPF-equipped satellites could serve as regional data anchors that aggregate and process traffic from multiple access satellites. However, this design introduces new challenges: it increases latency on the N3/N9 path between gNB and UPF due to inter-satellite forwarding, and it requires strict coordination and QoS control between the nodes. In addition, interruptions on the N9 path or satellite handovers could affect the continuity of the user plane unless mitigated by robust mobility anchoring and buffering. Option 3b is particularly relevant for mesh-enabled multi-orbit NTN architectures, where service chaining and distributed function placement are strategic enablers. It offers a modular and scalable path to support advanced satellite services, albeit with increased requirements for inter-satellite routing intelligence, network state propagation, and control-plane resilience.

Finally, Table 3 presents a summary of comparative CAPEX considerations for the presented architectural splits.

Table 3: Summary of comparative CAPEX for O-RAN NTN architectural splits

Architectural Option	Onboard Processing	Feeder Link / Ground Station	Additional ISL Hardware	Total Satellite CAPEX
Option 1 (DU/RU Onboard)	Medium (Only DU/RU)	High (Requires terrestrial CU/5G CN)	Varies (N/A for 1a, Medium for 1b)	Medium
Option 2 (Full gNB Onboard)	High (Full gNB stack)	Medium (Requires persistent N2/N3)	Varies (N/A for 2a, Medium for 2b)	High
Option 3 (Full gNB+UPF Onboard)	Very High (gNB + UPF/SEC)	Low (Local breakout)	High (Mesh routing)	Very High

2.5 RIC PLACEMENT OPTIONS

Deploying RICs in O-RAN-based NTNs requires addressing critical trade-offs between centralized intelligence and latency sensitivity. The placement of these components must consider several critical aspects, such as, the functional gNB splits described in previous section, the nature and origin of information exchanged by E2 nodes (i.e., CU and DU) and the capabilities offered by ISLs in terms of visibility and routing. While ground-based deployment simplifies management and AI/ML coordination, it introduces feeder-link delays that impair time-sensitive functions like beam management and handovers. Onboard deployment reduces latency but faces scalability and resource constraints in large constellations. With the above said, a ground-onboard trade-off architecture is essential to effectively balance these requirements through strategic functional distribution. Accordingly, we propose different potential architectural extensions, one on top of the other, to the O-RAN framework in this section.

Regarding the information reported by E2 nodes, a DU provides near-real time metrics related with physical resource block (PRB) usage, MAC scheduler statistics, cell and UE-level KPIs, beam management info. In contrast, the CU mainly delivers control-plane events, including RRC setup, handovers, mobility-related events and user-plane statistics like throughput per QoS flow, flow-level delay/loss, session setup/teardown events. While both E2 nodes play a role, in practice, most near-RT data consumed by xApps on the near-RT RIC comes from the DU, leaving CU as a supplementary element. In terms of communication infrastructure, current

LEO satellite system commonly utilize up to four ISLs per satellite: two intra-plane links and two inter-plane links. Satellites within the same orbital plane maintain permanent visibility of one another, whereas inter-plane visibility varies depending on multiple factors such as orbital altitude, angular separation between orbital planes, and the field of view and targeting capability of the ISL terminals (laser or RF). In modern constellations like Starlink (550 km orbit height) or OneWeb (1,200 km orbit height), inter-plane links typically last between 5 and 15 minutes per satellite pair, and the link quality remains stable throughout the visibility window. Regarding routing capabilities of LEO constellation, current routing solutions employ predictive static routing tables. Based on known orbits, routes are generated to/from each region of the planet, anticipating how the network will move over time, so forwarding decisions are made on each satellite, like a router. Based on that, current satellite networks, like the one deployed by Starlink, can reach intercontinental distances in 6-10 hops, where the typical latency per hop is about 2 to 4 ms.

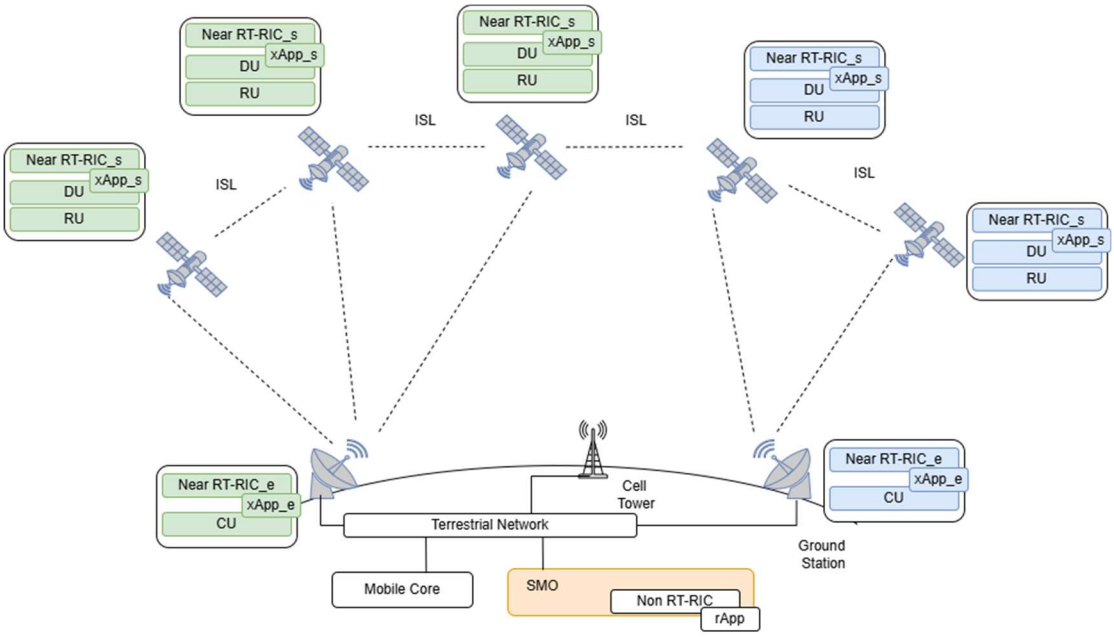


Figure 5: Architectural extension I: near-RT RIC split between Earth and Space

2.5.1 Architecture Extension I: Splitting near-RT RIC Across Earth and Space

According to O-RAN specifications, among the requirements of the Non-RT RIC we found that it "*shall support relevant AI/ML model training ... for generating/optimizing policies and intents to guide the behaviour of applications in Near-RT RIC or RAN*". These operations demand significant computational resources and power consumption, making it practical to deploy the Non-RT RIC exclusively on the ground. Furthermore, its control-loop operates at a time scale exceeding one second, thus not imposing additional burdens and making feasible this location on ground to propagate policies through O1 and A1 interfaces. However, the propagation of these policies need to take into account the visibility of the implied elements (e.g., Near-RT RIC, DU, CU) based on their location and the availability of feeder links.

In contrast, the Near-RT RIC operates with a much tighter control-loop ranging from 10 ms to 1 s. This implies low-latency communication with E2 nodes, thus imposing great burden in their deployment location. LEO satellites introduce a 5 to 20 ms round trip delay, making the near-RT RIC placement on Earth, while the DU is in space, infeasible for strict near-RT use cases (e.g., scheduling optimization, HARQ, radio resource management). Thus, this advocates for

a near-RT RIC in space, co-located with the DU, since most of near-RT data consumed by xApps comes from the DU, as stated before. According to the considered split in Section 2.2, where CU is on Earth, we propose a similar logical split for the near-RT RIC, that is, splitting the near-RT RIC between Earth and space, as depicted in Figure 5. In this split, we consider one component part of the near-RT RIC handling DU loops (on orbit) and one component handling CU loops (on Earth). Further digging into this logical separation, lightweight inference xApps are deployed on satellites (e.g., for fast beam selection or HARQ prediction), while heavier models are hosted on earth. Such an approach could provide several architectural benefits, such as: i) keeping E2 interface local; ii) avoids long-latency cross-E2 links; iii) logical separation aligns with RAN function splits (DU vs. CU) and iv) allows hierarchical xApp design (i.e., local fast xApps (space) and global coordination or policy xApps (Earth)). This approach, however, introduces complexity and several new challenges. It necessitates the definition of a new dedicated interface to enable coordination between the space and Earth segments of the RIC. The current A1 interface would also require extension to facilitate communication between non-RT RIC and the different near-RT RIC to perform policy sharing. Additionally, consistent state management becomes more complex, particularly during mobility events where the DU movement triggers an inter-CU handover. Ensuring reliable and low-latency coordination in such cases requires robust state-consistency mechanisms and efficient context handover procedures. Finally, inter-segment communication must be secured against space-ground link vulnerabilities, such as jamming or spoofing, adding further complexity to the system design.

2.5.2 Architecture Extension II: Network of near-RT RIC deployment in Space with dynamic E2 node assignment

When both the DU and CU are placed on orbiting satellites, as explored in Section 2.3 and Section 2.4, the whole near-RT RIC instance must be placed in space to meet tight control-loop timing requirements. However, as not highlighted in previous section, the addition of a new entity on the satellite is posing more burden to satellite capabilities in terms of processing and power consumption, thus requiring a very careful planning of the distribution of the different entities among satellites. In that sense, near-RT RICs instances will be deployed across the satellite constellation forming a network, where DU and CU attach to them dynamically based on their reachability, given the varying position, latency constraints and ISL availability, as depicted in Figure 6.

Within this scenario, the SMO and/or the non-RT RIC, via O1 interface, could play a central role in orchestrating E2 node reassignment to maintain low-latency control loops and computational balance. By monitoring satellite trajectories and using orbital knowledge (ephemeris) and real-time ISL status performance metrics, the system can predict when a given E2 node will exit the optimal control region of its current near-RT RIC instance. This foresight enables pre-emptive migration of the E2 interface connection to a better-positioned near-RT RIC instance. The reassignment is guided by multiple factors, including location, link quality, and load of satellite computational resources.

This predictive handover strategy ensures that latency constraints for near-RT control loops are consistently met, while also distributing load computational tasks more evenly among satellites. Moreover, it enhances network resilience by reducing the risk of performance degradation due to congested nodes, deteriorating ISLs, or localized RIC failures. The approach also supports scalability, as new satellites equipped with RIC capabilities can seamlessly integrate into the distributed control plane.

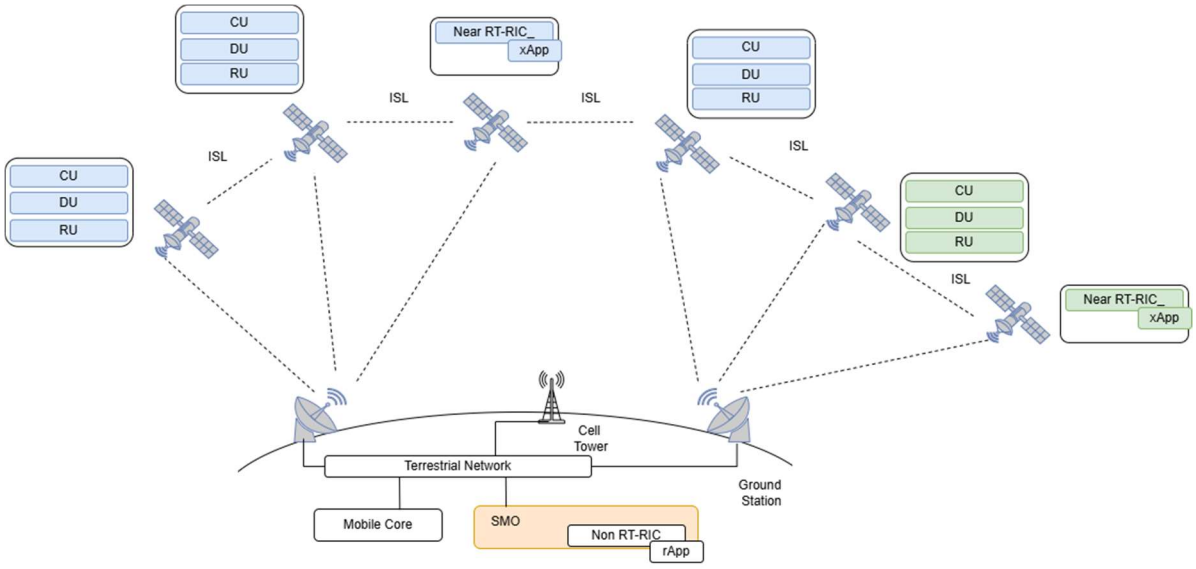


Figure 6: Architectural extension II: Network of near-RT RIC with dynamic assignment

Despite its potential, this architecture poses several challenges opening new research directions. A key limitation is the absence of standardized inter-RIC protocols in the current O-RAN framework, which hampers seamless coordination and state transfer between RIC instances. Realizing this architecture also requires predictive control mechanisms that can accurately forecast satellite movement and ISL status to initiate proactive and stateful handovers without service interruption. Such handovers can be even more complex if also considering the proposed extension in Section 2.5.1, which requires proper associations between DU and CU instances and corresponding components of the near-RT RIC (i.e., earth vs space) if there are CU instances on Earth, as mentioned for Split 2. Managing consistent control state -including UE contexts, policy timers, and RAN-specific variables- across RICs under such tight time constraints introduces substantial synchronization overhead. Addressing these challenges is essential to unlock the full potential of near-RT RIC deployment in NTN environments.

2.5.3 Architecture Extension III: Splitting Non-RT RIC Across Earth and Space Distributed Clusters

In architecture extension I, we proposed partitioning the near-RT RIC between terrestrial and space segments, while retaining the non-RT RIC exclusively on the ground. As an alternative extension, we can consider a configuration where the non-RT RIC is distributed across both terrestrial and satellite clusters, with near-RT RIC functions migrated onboard satellites. This may also be interpreted as a variation of extension architecture II. Let us consider a hierarchical policy control framework as shown in Figure 7. In this architecture, a ground-based non-RT RIC orchestrates network-wide policies, while distributed space-based RIC clusters enable localized optimization. Specifically, the satellite constellation is partitioned into multiple clusters, each managed by a designated cluster leader satellite equipped with a secondary non-RT RIC. The remaining satellites within each cluster, referred to as cluster followers, are responsible for executing near-RT RIC operations. Based on this idea, we establish the following hierarchical levels:

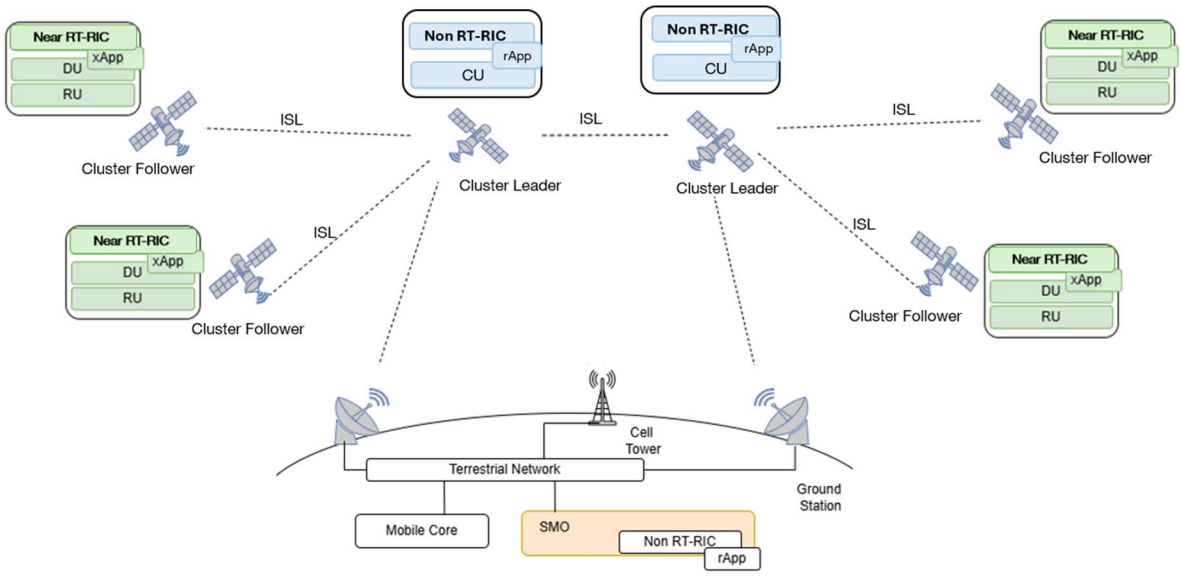


Figure 7: Architectural extension III: Splitting Non-RT RIC Across Earth and Space Distributed Clusters.

Level 1 (Ground Non-RT RIC): This entity is responsible for long-term, network-wide orchestration, including AI/ML model training and global policy definition. By centralizing intelligence, the level 1 controller ensures coordinated resource allocation across satellite clusters and minimizes redundant computations. This ground non-RT RIC can be envisaged to be located at SMO for example.

Level 2 (Cluster Leader Non-RT RIC): Each satellite cluster is managed by a leader satellite hosting a secondary non-RT RIC instance, effectively serving as an edge cloud node within the constellation. This layer refines and contextualizes global policies received from level 1, adapting them to cluster-specific operational conditions. Hierarchical ML frameworks are deployed at this level to decompose high-level objectives into actionable thresholds for follower satellites. Level 2 also manages the allocation and scaling of near-RT RIC instances across follower satellites, dynamically responding to real-time cluster demands. This edge autonomy mitigates dependence on ground connectivity, thereby ensuring resilience to temporary backhaul disruptions. The primary function of the cluster-level non-RT RIC is to perform localized optimization and distributed coordination within the cluster. The precise definition and formation of satellite clusters remains an open research question. This second non-RT RIC can be envisaged co-located with CU for the benefits explained previously.

Level 3 (Follower Satellite Near-RT RIC): At the lowest tier, co-located with DU and RU, the near-RT RIC instances operate directly on follower satellites, executing mission-critical tasks with sub-100 ms latency requirements. These entities are responsible for running xApps for real-time operations and implementing distributed learning algorithms, leveraging meta-policies provided by level 2 to optimize individual satellite performance.

The hierarchical interaction among these layers enables multi-timescale optimization: ground-based systems perform strategic planning on hourly or daily timescales, while cluster-level RICs facilitate minute-level adaptation. This separation of concerns allows terrestrial controllers to focus on global network KPIs (e.g., aggregate energy consumption across constellations), whereas onboard controllers optimize local metrics (e.g., per-beam throughput or ISL latency). An additional advantage of this architecture is the facilitation of inter-satellite coordination, managed by the cluster-level non-RT RIC, which is particularly advantageous for scenarios involving heterogeneous satellite roles. Furthermore, the dynamic allocation of near-

RT RIC instances by level 2 edge clouds supports seamless scalability as the constellation expands, thereby avoiding bottlenecks associated with centralized architectures.

Despite its potential, the proposed extension presents several challenges that merit further investigation. The optimal formation and definition of satellite clusters are complex tasks that require careful design to fully exploit the benefits of the level 2 non-RT RIC. In contrast to the current O-RAN architecture, the proposed system necessitates new interfaces between ground-based and cluster-level non-RT RICs. Additionally, the dynamic association and dissociation of satellite clusters may pose significant challenges in mobility management, synchronization, and overall system robustness. For instance, the F1 interface may need to be extended to support dynamic reassignment as satellites move, enabling fast context transfer. As in Extension I, enhancements to the A1 interface are also required to facilitate communication between the non-RT RIC and various near-RT RIC instances. Furthermore, the F1 interface may require protocol improvements to address variable ISL delay and jitter, as well as to support seamless handovers between moving satellites.

Finally, Table 4 summarizes the mapping with architectural splits and RIC placements.

Table 4: Summary of O-RAN Based Split Design Options for Integrated NTN

Option	O-RAN functions in Space	O-RAN functions on Ground	Pros	Cons
1a/1b	• Option 1a: RU + DU in the same satellite (Split 2) • Option 1b: RU + DU in different satellites (class 1 + class 2)	*Extension 1, 2 and 3:* CU + Near-RT RIC + Non-RT RIC + Core	• Lightweight satellite • Full reuse of ground infrastructure • Lower CAPEX	• F1 interface needs to be implemented in feeder link • Satellite must be able to maintain 2 feeder links (for handover) • Need to support variable latency in the feeder link and seamless handovers • High signalling load for CU handover • In option 1b, new fronthaul technology is required • High latency • No local control
2a/2b	• Option 2a: RU + DU + CU in the same satellite *-Extension 2:* Near-RT with dynamic E2 *-Extension 3:* Near-RT RIC in space • Option 2b: DU, CU and RU in different satellites	• *Extension 1:* Near-RT RIC splitting in earth and space • *Extension 2:* Non-RT RIC on earth • *Extension 3:* Non-RT RIC splitting on earth and space	• Reduce the dependence on earth -based controller and scheduler • Remove the stringent timing requirements • Reduce signalling burden • Seamless UE handovers at gNB level	-Higher burden on the satellite onboard processing unit -Quasi-continuous feeder link visibility -No caching capabilities (increased latency and higher load feeder links)

	-*Extension 2:* Near-RT with dynamic E2 -*Extension 3:* Near-RT RIC in space		• Good compromise between latency and arch. complexity • Easy to implement from protocol and hardware stand point	
3a/3b	• Option 3a: RU + DU + CU + UPF in the same satellite -*Extension 2:* Near-RT with dynamic E2 -*Extension 3:* Near-RT RIC in space • Option 3b: DU, CU, RU and UPF in different satellites -*Extension 2:* Near-RT with dynamic E2 -*Extension 3:* Near-RT RIC in space	• *Extension 1:* Near-RT RIC splitting in earth and space • *Extension 2:* Non-RT RIC on earth • *Extension 3:* Non-RT RIC splitting on earth and space	• Reduced application latency • Simplified interface management • Elimination of feeder-link dependence • Self-contained edge computing node in space • Compatible with mesh connectivity through the N9 interface at the ISLs	• High computational depends on onboard hardware • Higher payload complexity and power consumption • Interruptions on the N9 path or satellite handovers affects the continuity of the user place

2.6 DISCUSSIONS AND FUTURE WORK

This section outlines future research directions for architecture design, with a particular focus on the functional split and RIC placement. It is worth noting that aspects of the functional split related to the broader network softwarisation paradigm are addressed in WP5 ([25]), while the reference baseline was established in WP3 and documented in [26].

WP5 complements the work reported in this deliverable by further investigating network function placement. More precisely, the deployment of core network functions in space, as well as the placement of UPF and MEC components, as detailed in [25].

Open Challenges and Gaps Identified

Disaggregating gNB components in NTNs implies novel architectural and operational challenges due to the mobility of hosting platforms like LEO or MEO satellites and the strict latency and synchronization constraints of terrestrial RAN interfaces. While functional split architectures (presented Options 1 to 3 in Sections 2.2, 2.3 and 2.4) offer deployment flexibility, they require reevaluating all core interfaces and control functions in the 5G stack.

In Option 1, the DU and RU are co-located onboard the satellite, while the CU remains on the ground. This reflects a Split 2 architecture, where lower-layer functions (e.g., PHY, MAC, HARQ) are processed in space, reducing latency for time-sensitive operations. However, the F1 interface must traverse the feeder link between satellite and Earth. This link is sensitive to

propagation delay, intermittency, and jitter, which may compromise control-plane responsiveness and degrade user-plane throughput under variable link quality. In contrast, N2 and N3 interfaces (connecting the CU to the 5G CN) and the E2 interface (between CU and near-RT RIC) remain fully terrestrial and are unaffected by the satellite hop. Still, the end-to-end responsiveness of the RAN is limited by the F1 bottleneck, particularly during handovers or real-time reconfigurations initiated from the CU.

Option 2a places the RU, DU, and CU on a single satellite. This removes the need for internal F1 and E1 links and avoids fronthaul timing issues, but introduces feeder link constraints for N2/N3. If the RIC stays grounded, the E2 interface suffers from propagation delay, impacting time-sensitive functions like beamforming or mobility control. Deploying near-RT RICs onboard may mitigate these delays but brings added complexity for lifecycle management and reliability under satellite resource constraints. Option 2b spreads RU, DU, and CU across different satellites connected by ISLs. This modular approach supports scalability and redundancy, but stresses protocol limits, for instance in F1-C interface. This limits CU placement and requires strict control over constellation topology. E1 separation between CU-CP and CU-UP faces similar issues with added complexity for session state consistency across moving platforms. The Xn interface, essential for inter-gNB coordination, is poorly suited for dynamic satellite environments and needs redesign for beam-aware handovers and fast UE context forwarding. E2 becomes critical in this distributed setup. Near-RT control suffers under multi-hop signalling delays and losses. Solutions include embedding near-RT RICs with each DU or CU or clustering RICs across local satellites. These strategies require efficient synchronization and orchestration. xApps must be space-aware, using telemetry, visibility maps, and link conditions to manage HARQ, beamforming, and load balancing with limited compute resources. As satellites move or become unavailable, DU and CU nodes may need to migrate. Existing context transfer protocols are insufficient for maintaining continuity across F1 and E1 under dynamic topologies. Synchronization, vital for RAN performance, is also challenged by variable ISL delays. New timing methods or predictive mechanisms are needed to avoid HARQ and scheduling issues from skew and misalignment.

Option 3 adds onboard UPF, turning satellites into edge data nodes with local breakout over N6. This benefits edge computing use cases and eases backhaul load but creates new problems for routing, security, and state persistence. First, hosting gNB and UPF functions on a satellite imposes significant processing demands on the spacecraft. The satellite must perform real-time baseband processing, packet routing, and possibly edge computations within strict size, weight, power, and radiation constraints. The ESA 5G Space-Based Infrastructure paper [18] estimates the power needed: in a mid-term LEO scenario, just the NR baseband processing (full gNB) plus feeder link modem could draw on the order of 130+ watts onboard where the user link (onboard NR baseband processing/full gNB) consumes approximately 78.6 W and the feeder link modem (DVB-S2X) draws about 55.9 W. The computing load is significantly higher -for a fully orbital gNB compared to split architectures (55-70 % more demands) [24]. This means that more powerful onboard CPUs/DSPs or dedicated accelerators are required. In fact, researchers are also actively researching specialised hardware for satellite edge processing. The article in [27], for example, presents “GPU@SAT”, a GPU-like high-performance accelerator implemented on an FPGA and designed for onboard processing of intensive tasks in space. These efforts emphasise the need for space-grade computing hardware (multi-core CPUs, FPGAs/GPUs, AI accelerators) for the 5G RAN and the CN functions in orbit. At the same time, UPF failure or satellite loss also demand fast recovery and session relocation. For this reason, RICs must now account for N6 optimization and service placement, increasing their role in system orchestration.

To summarise, there are still unresolved gaps in all options. No current interface protocol supports the mobility of hosting nodes. RIC placement and xApp migration are not standardised for space environments. RICs lack awareness of ISL conditions and there is no

scalable satellite-native synchronization scheme. For gateways and session anchors, ground contact is still critical, making the systems vulnerable to link outages. Finally, onboard processing must operate under tight power limits, requiring lightweight and resilient designs. For this reason, overcoming these challenges requires more than the extension of terrestrial standards. They require new protocols, timing architectures and control strategies tailored to dynamic, delay-sensitive and distributed satellite networks.

Future Directions

As NTN integration progresses toward commercialization in 5G-Advanced and 6G ecosystems, several future directions must be pursued to mature the architectural and functional split models proposed in this work.

1. AI-Native Orchestration and Autonomy

Beyond static placements of functions, future systems should support advanced reconfigurable splits that can adapt to factors like link quality, mobility, and computational load. This level of dynamic control requires highly sophisticated orchestration platforms capable of real-time telemetry ingestion and predictive resource management across space and ground domains. This intelligent orchestration will optimize the interplay between Near-RT and Non-RT RICs. For instance, federated learning and semantic policy distillation can be used to manage these disaggregated control loops, enabling efficient global policy enforcement and local adaptation. Intent-based networking will also play a central role in this process. By moving beyond rigid configurations and toward a more autonomous, AI-driven architecture, NTN systems can achieve greater resilience, performance, and efficiency.

2. Implications for Standardization: 3GPP and O-RAN Alliance

The architectural and RIC placement strategies proposed in the project have significant implications for the evolution of both 3GPP and O-RAN Alliance standards. While 3GPP Release 17 and 18 have introduced support for NTN, these specifications primarily address transparent and regenerative payloads, and do not fully account for the architectural disaggregation enabled by O-RAN. Our findings highlight key areas where current standards fall short and require enhancement to achieve efficient TN-NTN convergence.

3GPP Implications: The mobility of non-geostationary satellites poses challenges to the existing 3GPP protocols, which assume stable connectivity. The NG-Flex configuration in TS 38.410 is a good starting point for managing gateway reassignment and session re-anchoring in full gNB deployments onboard (Option 2). However, further work is needed to standardise robust mechanisms for IP mobility, bearer context synchronization and QoS management under variable connection conditions. The study presented in TSG SA WG2 to support satellite edge computing via UPF onboard is also highly relevant, and confirms the feasibility of Option 3. In this regard, 3GPP could strengthen these efforts to define standard procedures for UPF onboard deployment, N9-based inter-satellite routing, and fast recovery from satellite failures.

O-RAN Alliance Implications: The Open Fronthaul and F1/E1 interfaces are not designed for the long round-trip delays and Doppler shifts that occur in space links. A new working group could be established to develop space-adapted fronthaul and midhaul standards that can cope with these limitations. This includes defining new transport layers and modifying HARQ and scheduling cycles. In addition, the O-RAN Alliance needs to standardise inter-RIC protocols, such as suggested in Section 2.5.2, to enable seamless coordination and state transfer between Near-RT-RIC instances distributed across a constellation. These new standards should also address the dynamic assignment of E2 nodes and the migration of xApps and

rApps to support a scalable and resilient control plane in NTNs. Finally, O-RAN's RICs should be enhanced to take into account ISL conditions and satellite telemetry for smarter and predictive resource management.

Conclusions

This section presents a structured framework for analyzing and designing architectural and functional split options in integrated terrestrial and non-terrestrial networks. By leveraging the modularity of the O-RAN architecture, multiple configurations are evaluated for distributing RAN functions (RU, DU, CU) and UPF components across space and ground domains. The analysis has identified key trade-offs related to latency, onboard complexity, feeder link dependency, and autonomy, culminating in three principal design options: (i) DU and RU onboard, (ii) full gNB onboard, and (iii) full gNB with UPF and SEC functions onboard. For each option, sub-variants were considered depending on whether the functions were co-located on the same satellite or distributed across multiple nodes connected via ISLs.

RIC placement strategies are proposed to complement these split designs, outlining both ground-based and hybrid control architectures. The mapping between split configurations and RIC distribution showed that control plane responsiveness and optimization capabilities depend heavily on the persistence and latency of inter-domain links. While each architectural option offers different benefits, increasing autonomy and reducing latency generally require migrating more intelligence onboard. However, this comes with higher processing demands, greater implementation complexity, and power constraints. Therefore, architectural decisions must be tailored to specific mission objectives, satellite class, and required levels of ground independence. Beyond performance metrics, function placement strategies must also account for operational feasibility, spectrum regulations, and interoperability with existing 5G CN. Among the three principal architectural options studied, Option 2a-with the full gNB stack (RU, DU, CU) integrated onboard a single satellite- emerges as the most practical and balanced choice in terms of protocols and hardware standpoint under current technology constraints. It avoids fronthaul and midhaul timing bottlenecks, reduces signalling latency, and enables low-complexity RAN management while preserving compatibility with 5G CN interfaces. Its self-contained structure simplifies orchestration, enhances scalability, and lowers dependency on persistent feeder links or high-throughput ISLs. These qualities make Option 2a particularly suited for medium-class LEO/MEO constellations targeting broadband NTN deployments in the near term. Looking ahead, O-RAN will be instrumental in enabling intelligent, flexible, and modular NTN systems, serving as the architectural backbone for 6G networks that span terrestrial and non-terrestrial domains.

3 AI-TECHNIQUES FOR NON-REAL-TIME RRM

This section describes AI techniques to predict radio demand loads in NTN environments to enable proactive resource management. The section details how the traffic forecasts are exploited to perform radio resource allocations on a time scale from seconds to minutes. The proposed solutions include probabilistic forecasting, federated learning (FL) and, finally, multivariate AI-based prediction techniques for multi-beam satellites. Sections 3.1 and 3.2 provide innovative solutions that extend previous research on KAN detailed in [1]. Section 3.2 shows how AI-based state-of-the art multivariate probabilistic forecasting can be considered in multi-beam satellites. The key findings are presented in the following sections.

3.1 PROBABILISTIC FORECASTING-ENABLED KOLMOGOROV ARNOLD NETWORKS

When building a forecasting model, one can distinguish between deterministic and probabilistic approaches. A deterministic model produces a single point estimate of the future value, such as the predicted next observation of a time series. This is useful if the only goal is to approximate the central tendency of the process. However, in many applications it is equally important to quantify the uncertainty surrounding a forecast. Probabilistic forecasting addresses this by modelling the entire distribution of possible future values, rather than a single number. This approach is particularly useful to reduce under-provisioning and reduce the over-provisioning of radio resources.

Kolmogorov–Arnold Networks (KANs) have recently emerged as a promising alternative to conventional Multi-Layer Perceptron (MLP)-based architectures [28]. KANs are grounded in the Kolmogorov–Arnold representation theorem [29], which establishes that any continuous multivariate function can be represented as a finite composition of univariate functions. Recent studies have demonstrated the effectiveness of KANs in time-series forecasting tasks [30]-[31]. As outlined in Section 4.1.3.1 of Deliverable [1], KANs are particularly advantageous for NTN applications, as they provide higher or comparable forecasting accuracy with reduced complexity. This enables the use of simpler and potentially faster models. Such efficiency is particularly important in contexts where computational resources are constrained or where rapid model deployment is essential.

In this work, we extend the work on KANs presented in [1] enabling KAN-based architectures to predict the probabilities associated with the future traffic and radio resource demands (e.g. PRBs). To this end, we introduce Probabilistic KANs (P-KANs), which incorporate uncertainty modelling intrinsically within Kolmogorov-Arnold Networks. We directly modelled the parameters of the probability distributions for the future timesteps and we trained the KAN to learn the mean and the variance of the parametric models. This formulation streamlines the overall framework, enhances scalability, and allows the model to adapt more effectively to heterogeneous datasets. Moreover, by parameterizing probability distributions directly, P-KANs are better positioned to capture heteroscedasticity and to deliver accurate, data-driven uncertainty estimates.

3.1.1 Problem Statement

We begin by formalizing the objective of probabilistic time series forecasting. Consider a univariate time series given by $\{y_t\}_{t=1}^{t=t_0+T}$. Let us denote:

- by $y_{t_0:T} \coloneqq [y_{t_0}, y_{t_0+1}, \dots, y_{t_0+T}]$ the forecast horizon of horizon length T

- by $\boldsymbol{y_{1:t_0-1}} \coloneqq [y_1, \dots, y_{t_0-1}]$ the historical values, where t_0 denotes the time we assume to be unknown.
- by $\boldsymbol{x} \coloneqq [y_{t_0-L}, \dots, y_{t_0-1}]$ the context window of size L, which considers the most recent L samples up to time t_0.

Probabilistic forecasting requires to learn a model that characterizes the conditional probability distribution:

$$P\left(\boldsymbol{y_{1:t_0-1}} \mid \boldsymbol{x}\right)$$

3.1.2 Probabilistic Kolmogorov-Arnold Networks (P-KANs)

Herein, we propose P-KAN a neural network architecture, derived from the Kolmogorov-Arnold representation theorem, developed to estimate the probabilistic distribution of traffic and/radio resources for NTN. Rather than generating deterministic point forecasts, the model produces the parameters of a predefined probability distribution at each time step t. The architecture is composed of an input layer, where the number of neurons corresponds to the time steps within the conditioning interval, several functional (hidden) KAN layers, hence, composed by learnable activation function, and an output layer. This output layer contains a number of neurons equal to the number of timesteps in prediction range (i.e., the horizon length) multiplied by the number of parameters of the underlying (considered) probability distribution, denoted by p. In other words, the size of the output layer is $T \times p$.

The p outputs are employed to estimate the parameters of the probability distribution at each time step t (e.g., the mean and the standard deviation in case an underlying Gaussian distribution is assumed), thereby quantifying forecast uncertainty. For example, if the likelihood is specified as a Gaussian distribution, it is parameterized by a mean μ_t and a standard deviation σ_t, and is given by:

$$l(y_t|\mu, {\sigma_t}^2) = \frac{1}{\sqrt{2\pi{\sigma_t}^2}} \exp\left(-\frac{(y_t - \mu_t)^2}{2{\sigma_t}^2}\right)$$

Assuming than the mean and the standard deviation can vary across the different time steps t, the log-likelihood for multiple independent observations $\{y_t\}_{t=1}^{t=n}$ is provided by

$$log\ l(\ y \mid \{\mu_t, \sigma_t^2\}_{t=1}^{n}\ ) = \sum_{t=1}^{n}\left[-\frac{1}{2}\log(2\pi\sigma_t^2) - \frac{(y_t - \mu_t)^2}{2\sigma_t^2}\right]$$

Let us denote by $z_t = [\boldsymbol{z}_{t,\mu}, \boldsymbol{z}_{t,\sigma}]$ the output of P-KAN at timestep t. P-KAN is trained to optimize the log-likelihood and estimates the parameters of the Gaussian distribution (mean and standard deviation) as follows

$$\mu_{t(\boldsymbol{z}_{t,\mu})} = \boldsymbol{z}_{t,\mu}$$

$$\sigma_{t(\boldsymbol{z}_{t,\mu})} = log\left(1 + exp(\boldsymbol{z}_{t,\sigma})\right)$$

Where the expression of $\sigma_{t(\boldsymbol{z}_{t,\mu})}$ is just an affine transformation of the $\boldsymbol{z}_{t,\sigma}$ to guarantee only positive values. The objective of this procedure is to convert unconstrained network outputs

into strictly positive standard deviations for probabilistic modelling, in a smooth and differentiable way. $\boldsymbol{z}_t$ is obtained through successive summations and univariate transformations as defined by the Kolmogorov-Arnold decomposition [28]. For the sake of simplicity and readability, we omit the details herein.

In a similar way, we can assume the likelihood to be the t-student location-scale distribution given by:

$$l(y_t|\nu,\mu,\sigma^2) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sigma\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(\frac{\nu+(\frac{y_t-\mu}{\sigma})^2}{\nu}\right)^{-\frac{\nu+1}{2}},$$

In this case, the Kolmogorov-Arnold Network can be trained to optimize the t-student log-likelihood, which depends on $p = 3$ parameters, depending on the output layer $z_t = [\boldsymbol{z}_{t,\mu}, \boldsymbol{z}_{t,\sigma}, \boldsymbol{z}_{t,\nu}]$ as follows:

$$\mu_{t(\boldsymbol{z}_{t,\mu})} = \boldsymbol{z}_{t,\mu}$$

$$\sigma_{t(\boldsymbol{z}_{t,\mu})} = log\left(1 + exp(\boldsymbol{z}_{t,\sigma})\right)$$

$$\sigma_{t(\boldsymbol{z}_{t,\nu})} = log\left(1 + exp(\boldsymbol{z}_{t,\nu})\right) + 2$$

3.1.3 Performance evaluation

Next, we assess the performance of P-KAN algorithm by means of some simulation results. Having this aim in mind, we have considered the 'Satellite Network Dataset' provided by Hispasat presented in D1.1. Throughout this subsection, we have considered the following architectures for the MLP (Multilayer Perceptron) and the KANs:

- MLP with 3 hidden layers de 300 neurons, i.e., [168, 300, 300, 300, 24]
- KAN the following layers are considered, i.e., [168, 30, 30, 30, 24]

We considered a conditioning range of 168 hours (7 days) to forecast the next 24 hours (day) as is illustrated in the following Figure.

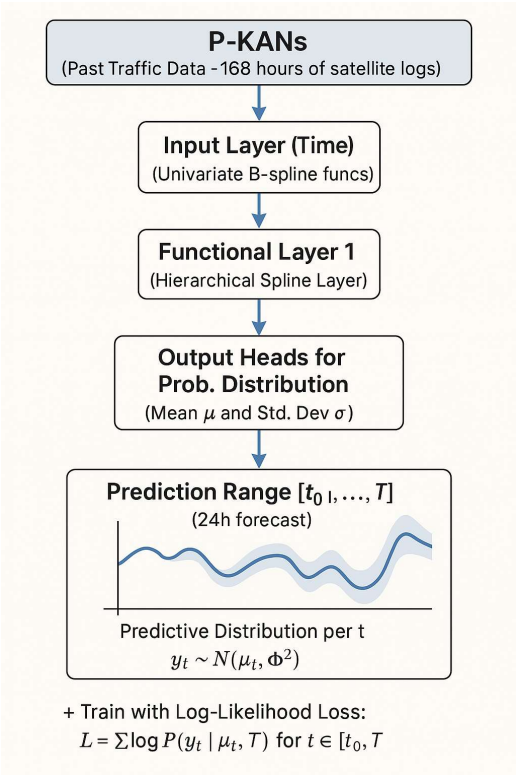


Figure 8. Illustration of P-KAN architecture.

The comparative evaluation of probabilistic and point forecast models for Physical Resource Block (PRB) allocation highlights the crucial trade-offs at the heart of radio resource management (RRM). While prediction error is an important metric, effective allocation requires balancing spectral efficiency, expressed as PRB saving, against quality-of-service (QoS) risks such as under-provisioning. The figures presented illustrate these dynamics from multiple perspectives, combining accuracy, provisioning trade-offs, model complexity, and integrated performance views to provide a holistic picture of model behaviour.

The first dimension of the analysis concerns raw predictive accuracy. As shown in the next Figure, the point forecast (PF) models achieve the lowest error values across all models. The KAN-PF (originally presented in [1]) achieves the best scores overall, significantly outperforming both Gaussian and T-Student probabilistic KANs and far exceeding the MLP probabilistic models, which surpass 3.0 in MAE. The MLP-P, an extension of conventional MLPs to predict the probabilistic distribution parameters [32], while less parameter efficient, still provides much stronger accuracy than its probabilistic counterparts. These results confirm that when viewed purely from a prediction standpoint, point forecasts appear as highly effective baselines. However, accuracy alone tells an incomplete story, and subsequent figures reveal why point forecasts are ill-suited as a standalone solution for PRB allocation.

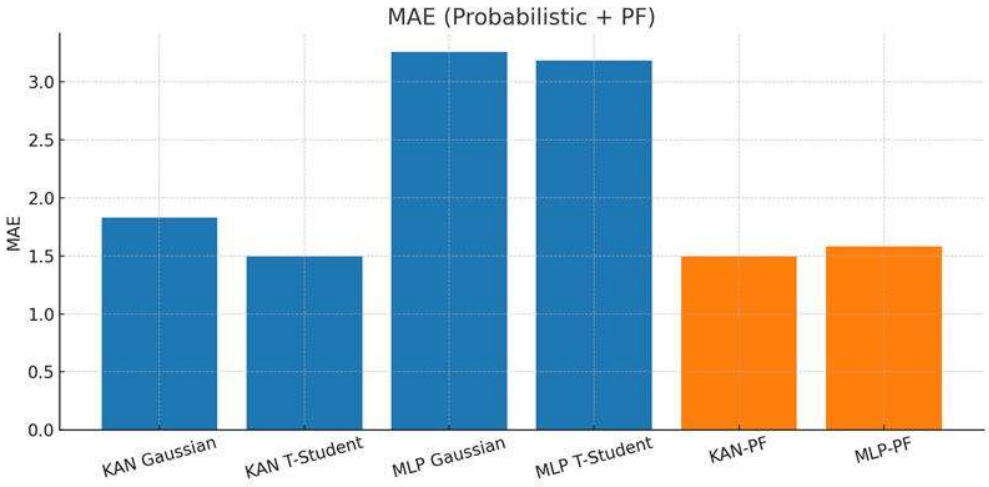


Figure 9. Mean Absolute Error of Probabilistic Methods vs its point forecast counterparts. In probabilistic methods, the median is considered for the computation of the MAE.

Figure 10 focuses on provisioning trade-offs and demonstrates why probabilistic forecasting is essential. For probabilistic forecasting algorithms, we considered the 99-th percentile, since this value is able to serve the traffic/resource demand with a very high probability (99%). The PRB saving is highest for point forecasts, with both KAN-PF and MLP-PF exceeding 62% savings compared to 30% for Gaussian KAN and 53% for T-Student KAN. On the surface, this seems highly efficient, but it also reveals the risk hidden behind these numbers. The same models that save the most PRBs also exhibit catastrophic underprovisioning rates: KAN-PF underprovisions in 55% of cases, compared to only 5.6% for Gaussian KAN. The Pareto scatter plot in Figure 11 makes this trade-off explicit: PF models sit on the frontier of maximum saving but at extreme risk, while Gaussian KAN anchors the safety frontier with minimal underprovisioning and lower savings. T-Student KAN occupies a middle ground, providing significant efficiency gains while accepting moderate increases in underprovisioning. Together these figures highlight the defining weakness of point forecasts: they are risk-blind, offering efficient but unreliable allocations that compromise QoS.

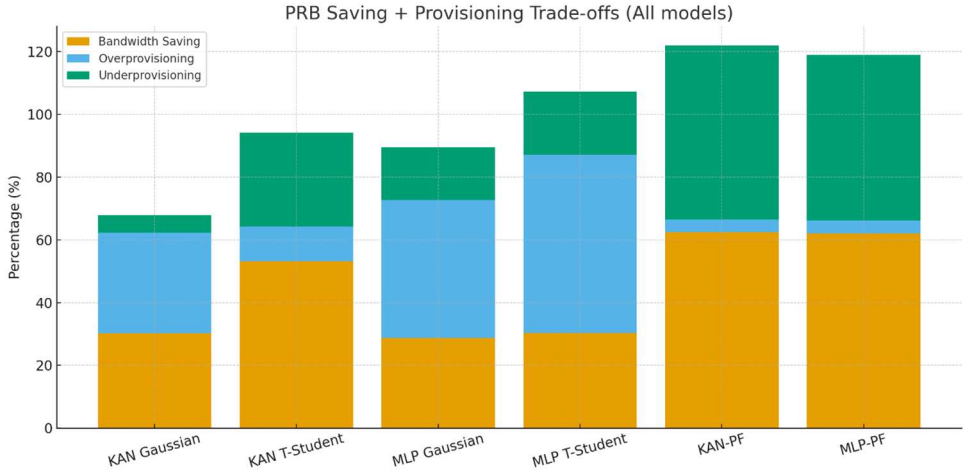


Figure 10. Percentages of over-/under-provisioning with the different Probabilistic Forecasting techniques with 99-th percentile, and Point-forecasting algorithms. Bandwidth saving is also provided in comparison with static bandwidth allocation.

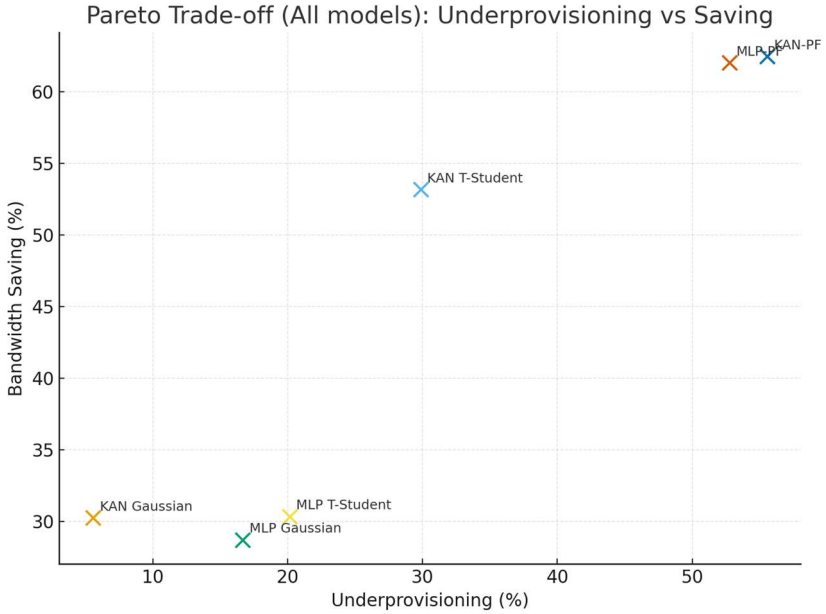


Figure 11. Bandwidth saving vs underprovisioning percentage.

The strength of probabilistic models is well illustrated by the CRPS and FIC metrics in Figure 12 and Figure 13. Let us first define both metrics:

- The Continuous Ranked Probability Score (CRPS) generalizes the MAE to the case of probabilistic forecasts. The CRPS is frequently used in order to assess the respective accuracy of two probabilistic forecasting models. The CRPS (Continuous Ranked Probability Score) is a proper scoring rule used to assess the quality of probabilistic forecasts of continuous variables. It measures the difference between the entire forecast cumulative distribution function (CDF) and the actual observed outcome, combining both calibration (how well predicted probabilities match observed frequencies) and sharpness (concentration of the predictive distribution).Formally, if the forecast is given as a cumulative distribution function $F(y)$ for a continuous variable, and the actual observation is given by x, then the CRPS is defined as:

$$\mathrm{CRPS}(F,x) = \int_{-\infty}^{\infty} (F(y) - \mathbf{1}\{y \geq x\})^2 \, dy.$$

- Where $F(y)$ is the predictive CDF, and 1{y≥x} is the Heaviside step function representing the observed outcome. CRPS has some key properties, lower CRPS is the better, a perfect forecast has CRPS = 0. Furthermore, it penalizes both bias (systematic deviation) and dispersion (uncertainty spread).

- The Forecast Interval Coverage (FIC) is a statistical measure that tells you how often the true values of a prediction fall inside the predicted forecast intervals (or prediction intervals).

The CRPS values highlight how the T-Student KAN generates sharper and more informative predictive distributions, achieving lower error in terms of probabilistic accuracy. However, the FIC results reveal that these distributions are often too narrow, with systematic undercoverage relative to the nominal levels. This explains why the T-Student KAN, while efficient, exposes the system to higher risk of underprovisioning. In contrast, the Gaussian KAN shows higher CRPS values, indicating less sharp distributions, but its FIC curves are better aligned with the desired coverage levels. This demonstrates a more conservative calibration that reduces risk and ensures greater reliability in PRB allocation. Together, these plots emphasize that probabilistic forecasting is not only about sharper distributions but about calibrating intervals correctly so that operators can rely on them for informed resource management decisions.

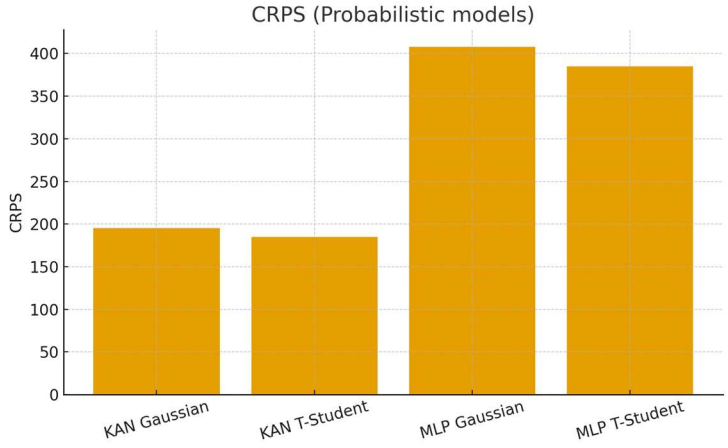


Figure 12. Continuous Ranked Probability Score metric for the P-KAN and MLP with different considered distribution (Gaussian and T-student).

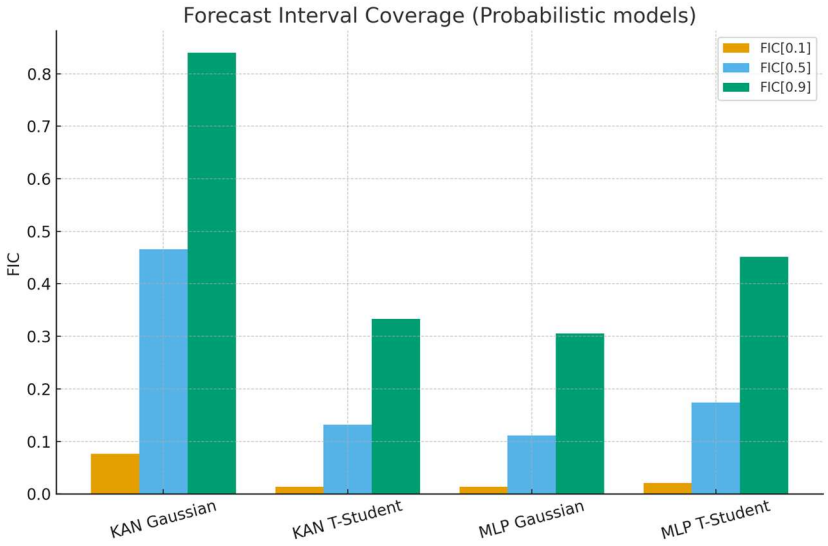


Figure 13. FIC metric for P-KAN and MLP with different underlying distributions (Gaussian and T-student).

Model complexity provides another crucial layer to the analysis. Next Figure compares the number of parameters across all models, showing that KAN architectures achieve their performance with around 75k-90k parameters, compared to 240k-250k for MLPs. This efficiency is particularly notable in the KAN-PF, which requires fewer parameters than MLP models yet delivers superior error and provisioning trade-offs. These results make clear that KANs are not only more accurate than MLPs but also significantly more resource efficient, a vital consideration for practical deployment in base stations or distributed RAN elements where computational budgets are constrained.

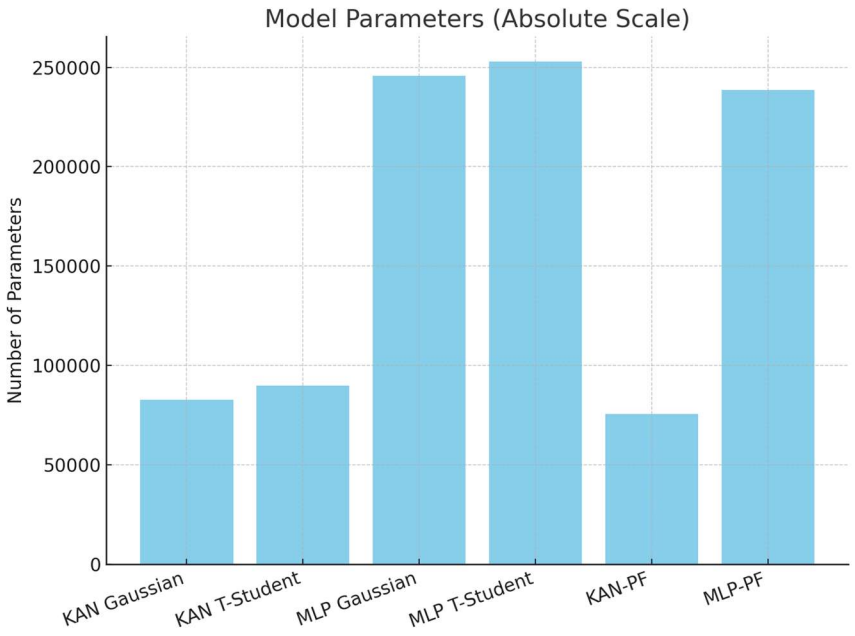


Figure 14. Number of parameters for the different point-forecast and probabilistic forecasting techniques presented.

The results show the overall dominance of KAN models across dimensions. The T-Student KAN maximizes efficiency, achieving low errors and high PRB saving, but at the cost of higher underprovisioning risk. Gaussian KAN adopts a more conservative strategy, delivering moderate PRB savings but much lower underprovisioning and better calibrated intervals. The PF models, while leading in error metrics, collapse on provisioning trade-offs, making them unsuitable for risk-sensitive RRM applications. Furthermore, probabilistic KANs, and particularly the Gaussian and T-Student variants, are the most balanced and flexible solutions for adaptive PRB allocation. Taken together, the figures demonstrate a clear narrative. Point forecasts are extremely accurate and efficient in saving PRBs, but they lack the ability to quantify uncertainty and therefore expose the system to unacceptable QoS risks. Probabilistic forecasts, in contrast, explicitly represent uncertainty, enabling operators to balance spectral efficiency against service reliability. Among these, the T-Student KAN is the most aggressive, achieving high efficiency but requiring acceptance of greater risk, while the Gaussian KAN provides a safer and more calibrated profile. MLPs, whether probabilistic or point forecast, are consistently outperformed by KANs both in accuracy and in parameter efficiency. In summary, probabilistic forecasting emerges as indispensable for PRB allocation in modern RRM. The choice between T-Student and Gaussian KAN depends on operator priorities: maximizing PRB efficiency versus safeguarding QoS. Point forecasts can serve as lightweight baselines for contexts where QoS risk is less critical, but in realistic wireless environments where demand fluctuates and reliability is paramount, probabilistic forecasts provide the only viable path forward. By quantifying uncertainty and enabling flexible provisioning strategies, they empower operators to achieve adaptive, risk-aware resource allocation that balances user experience with spectral efficiency.

Next, we illustrate the 24h prediction for different probabilistic methods. The Gaussian P-KAN (Figure 15) provides the most conservative probabilistic forecast. Its prediction intervals are deliberately wide, comfortably enclosing the true PRB demand trajectory. This design choice translates into low underprovisioning risk, which is critical for maintaining QoS guarantees in RRM. The cost, however, is evident: many intervals overshoot the actual demand, leading to systematic overprovisioning. Thus, Gaussian KAN exemplifies a risk-averse policy, favouring service reliability over spectral efficiency.

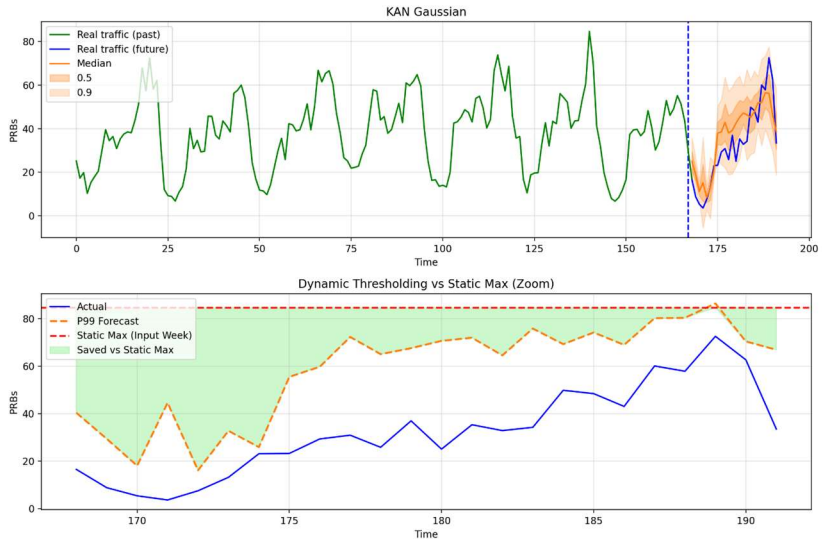


Figure 15. Upper Figure: Gaussian P-KAN 24h forecast for different confidence intervals. Lower Figure: Percentile 99th vs static PRB allocation over time and actual PRB demands.

In contrast, the T-Student KAN (Figure 16) demonstrates how sharper distributions can enhance efficiency. Its intervals are narrower and more adaptive, often hugging the demand curve closely. This yields greater PRB savings and shows how uncertainty modelling can be tuned for efficiency. Still, the trade-off is visible: whenever demand surges beyond the forecast, underprovisioning events appear. Hence, T-Student KAN embodies a more aggressive, efficiency-driven policy—appealing in scenarios where spectral utilization is paramount, but potentially riskier for QoS.

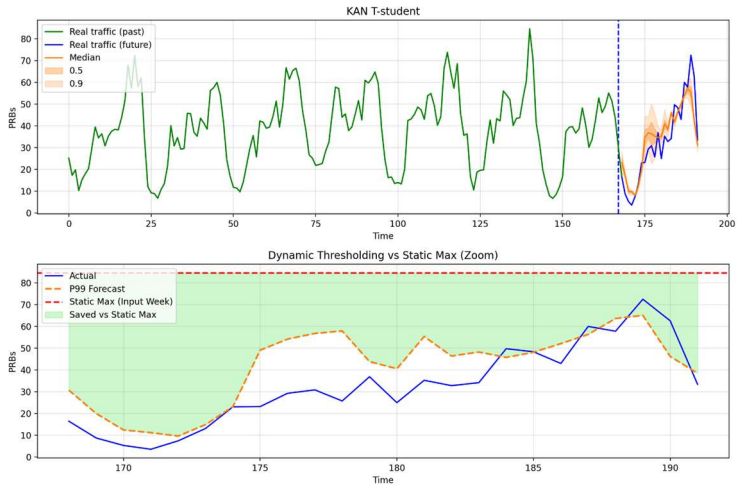


Figure 16. T-student P-KAN prediction of 24h PRB demands. Upper Figure: 50% and 90% confidence intervals. Lower Figure: comparison between 99% forecast and static allocation.

3.2 FEDERATED KANS

Traditional centralized learning approaches struggle to adapt to the dynamic nature of NTN due to intermittent connectivity, limited bandwidth and high latency. FL has emerged as a promising solution that enables decentralized model training across multiple distributed clients while preserving data privacy and reducing communication overhead. Despite the benefits of FL, challenges remain in selecting appropriate ML models that can effectively capture the complex and heterogeneous nature of NTN traffic. Conventional deep learning models, such as MLPs, often require extensive parameter tuning and have difficulty generalizing well under different network conditions. Moreover, the dynamic and time-varying characteristics of NTN traffic require models that can efficiently adapt to non-stationary patterns without incurring excessive computational overhead.

To address these challenges, Fed-KAN are proposed as a novel framework for optimizing network resource allocation in NTN environments. KANs offer superior functional approximation capabilities compared to conventional MLP [28], by learning more structured and interpretable representations of complex function, making them particularly suitable for capturing complex traffic patterns in satellite-based communication systems. By integrating KANs into an FL framework as in [33], predictive accuracy is enhanced while reliance on frequent satellite-ground communications is reduced, thereby improving the overall efficiency of NTN operations. The proposed architecture consists of multiple satellite (beams) acting as FL clients, each collecting uplink and downlink traffic data from users within their coverage area. The satellites use KAN-based local models to predict future traffic patterns and regularly exchange model updates with a central aggregator. The ground station, acting as an FL server for aggregation, integrates these updates to refine a global model, which is then distributed back to the to the satellite clients to improve local inference.

The proposed federated approach not only minimizes bandwidth consumption but also enhances adaptability to spatial and temporal variations in NTN traffic patterns. Together with this decentralized approach, adaptive traffic classification and efficient resource allocation for different application categories such as streaming, cloud services, communication and system updates can be enabled efficiently. It can also be used to support intelligent handover mechanisms to optimize the distribution of traffic in overlapping coverage areas of satellite beams. Overall, the integration of Fed-KAN mechanism into NTN can provide better generalization and interpretability when learning complex traffic behaviour and addresses the limitations of traditional FL models in NTN scenarios. Hereinafter, a comprehensive analysis of Fed-KAN and traditional FL-based methods, such as Federated MLP, is provided. The main findings are published in [34].

3.2.1 FL and KANs for NTN

FL has gained significant attention in recent years as a distributed machine learning paradigm designed to train models across decentralized clients while preserving data privacy. McMahan et al. [35] introduced the foundational concept of FL, demonstrating its potential for large-scale collaborative learning without centralizing sensitive data. Subsequent research has explored FL applications in various domains, including edge computing, mobile networks, and satellite communications [36]. In the context of NTNs, FL has been proposed as a solution to mitigate the challenges arising from intermittent connectivity and limited bandwidth in LEO satellite systems. Recent studies [37, 38] have explored FL-based architectures for satellite communication networks, focusing on efficient model aggregation and transmission optimization. These approaches aim to reduce latency and improve adaptability to network variations, which is crucial in dynamic NTN environments.

KANs, on the other hand, have emerged as an alternative to traditional neural networks by leveraging learnable activation functions instead of fixed ones [28]. This approach enables KANs to achieve superior function approximation, making them particularly suited for complex predictive modelling tasks. Our previous paper [33] introduces a novel FL approach that utilises KANs for classification tasks and aims to improve accuracy while preserving privacy. The study shows that Fed-KANs outperform conventional federated MLPs in metrics such as accuracy, precision, recall, F1-score, and stability, suggesting their potential for more efficient and privacy-preserving predictive analytics. Although KANs have been studied mainly in mathematical and scientific contexts, their integration in FL is still largely unexplored in satellite domain. Our work extends this research by proposing a Fed-KAN framework specifically tailored to the optimization of NTN resources. Furthermore, NTN-based resource allocation has been extensively studied in the literature, with techniques ranging from deep reinforcement learning [37] to graph neural networks [38]. However, most existing approaches are based on centralized learning architectures that suffer from high communication overhead and a lack of adaptability to the dynamics of real-time traffic. With the introduction of Fed-KAN, we close this gap and offer a decentralized and interpretable solution that increases network efficiency while preserving privacy.

3.2.2 General Architecture

The NTN-based RAN consists of LEO satellites that are responsible for managing the communication between the users within beams and the ground station connected to data/cloud network. Figure 17 shows the general architecture in an NTN-enabled system, in which several beams enable connectivity to various applications. Each beam of satellite represents LEO satellite beams, each covering a specific group of users within their area of operation. Each beam also has specific uplink/downlink traffic data from various user applications such as YouTube, Netflix, Instagram, and/or cloud communication services that use the satellite operator's services. The core network and the data network in the ground provide the backend infrastructure for data processing and Internet access. The core network

connects the aggregator to the wider data network, enabling large-scale traffic management and predictive analytics.

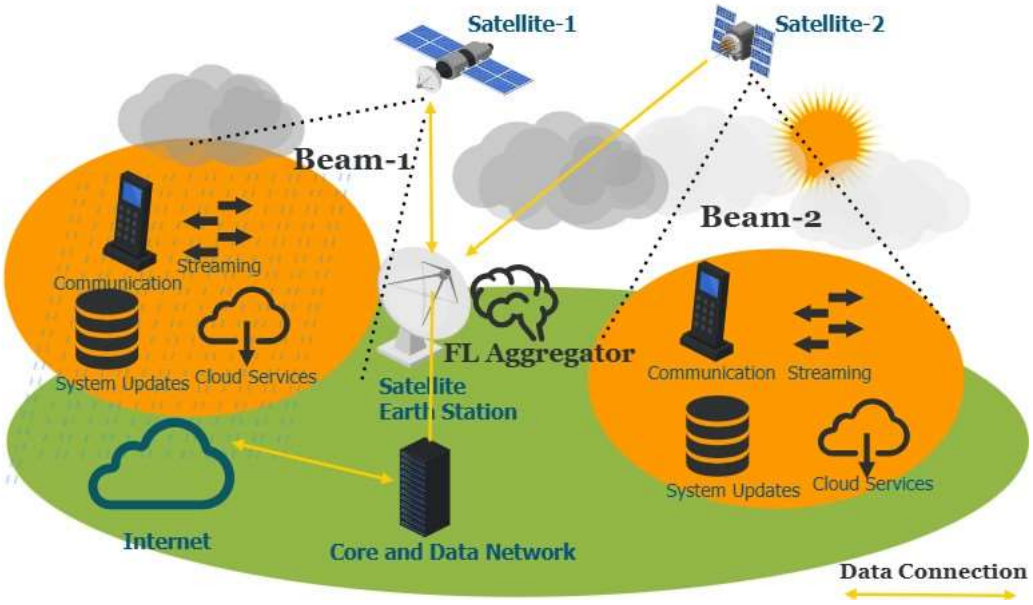


Figure 17: General architecture of NTN and the existence of diverse application traffic on each separate beam of satellites.

The architecture of Figure 17 can also be used to facilitate decentralized learning and efficient traffic prediction. Multiple beams on each satellite can act as localized FL clients that facilitate the transmission of local model updates from the individual beams to the central aggregator (e.g. the ground station), reducing the need for a direct and continuous connection to the core network. The aggregator serves as an FL server, aggregating local model updates received from multiple beams and plays a crucial role in classifying and predicting the distribution of traffic. Once the models are aggregated, the global update is sent back to the beams to refine their local predictions. A key advantage of the federated framework in this architecture is its ability to enable decentralized learning, ensuring that each beam learns optimally from local traffic patterns. FL in this NTN scenario can ensure that each beam trains its model independently based on local data, preserving privacy and reducing bandwidth consumption. This improves the scalability and personalization of the models, while minimizing the communication overhead between the satellites and the ground station and network. Improved prediction/classification capabilities of Fed-KAN can support intelligent resource allocation and optimized handover decisions for LEO satellites for better decision making. This is particularly useful for maintaining seamless connectivity in scenarios where satellite beams overlap and enables a smooth handover of traffic flows.

3.2.3 Federated KAN

KANs offer a powerful alternative to traditional neural networks (e.g. MLPs) by replacing fixed activation functions with learnable, spline-based transformations. The Kolmogorov-Arnold representation theorem states that any multivariate continuous function can be represented as a composition of simple univariate functions [39]. This property makes KANs particularly effective in capturing complex, nonlinear traffic patterns in a distributed environment. Proposed Fed-KANs extend this concept to FL, where multiple clients independently train local models while a central server aggregates the model updates without directly accessing raw data. Figure 18 shows these steps with nodes representing the main operations and arrows indicating the flow and iteration of the FL process for the particular NTN scenario where

multiple beams in satellites act as Fed-KANs clients and ground station as Fed-KANs server aggregator. The FL process follows the Federated Averaging (FedAvg) algorithm, where each Fed-KAN client trains an independent model on its local data before sending the updated model weights to the central server. The server on the ground aggregates these weights by averaging and iteratively updates the global Fed-KAN model over multiple federated rounds. This decentralized learning approach ensures that no raw data is shared between beams, preserving data privacy while taking advantage of collaborative learning across distributed datasets. By leveraging the KAN in a FL framework, Fed-KAN can enable decentralized, privacy-preserving learning for traffic prediction, making it well-suited for satellite network optimization and real-world forecasting applications.

Fed-KAN layers

The architecture of Fed-KANs consists of multiple layers where each Fed-KAN layer composed of univariate spline functions. These spline-based transformations allow the network to approximate nonlinear relationships in traffic patterns efficiently. The grid-based representation ensures smooth interpolation between control points, adapting dynamically to fluctuations e.g. in satellite beam traffic. The input layer receives two features: Downlink and Uplink traffic values for the last W hours. These inputs are then passed through a series of KAN layers, each responsible for incrementally refining the feature space. The first KAN layer performs feature transformations, while the second KAN layer applies non-linear approximations to capture complex dependencies in the data. The third KAN layer extends this by calculating higher order functions so that the model can learn complicated patterns in network traffic.

In the problem under study with prediction of next hour traffic, after the classical KAN transformations, the processed features go through the fully connected (FC) layers that further refine these features, with FC layer 1 containing 8 spline nodes and FC layer 2 containing 4 spline nodes, each incorporating ReLU activations and dropout regularization to improve learning stability at each client. Finally, the output layer predicts the percentage traffic values for the next hour for four main traffic categories, namely communication, streaming, cloud services and system updates.

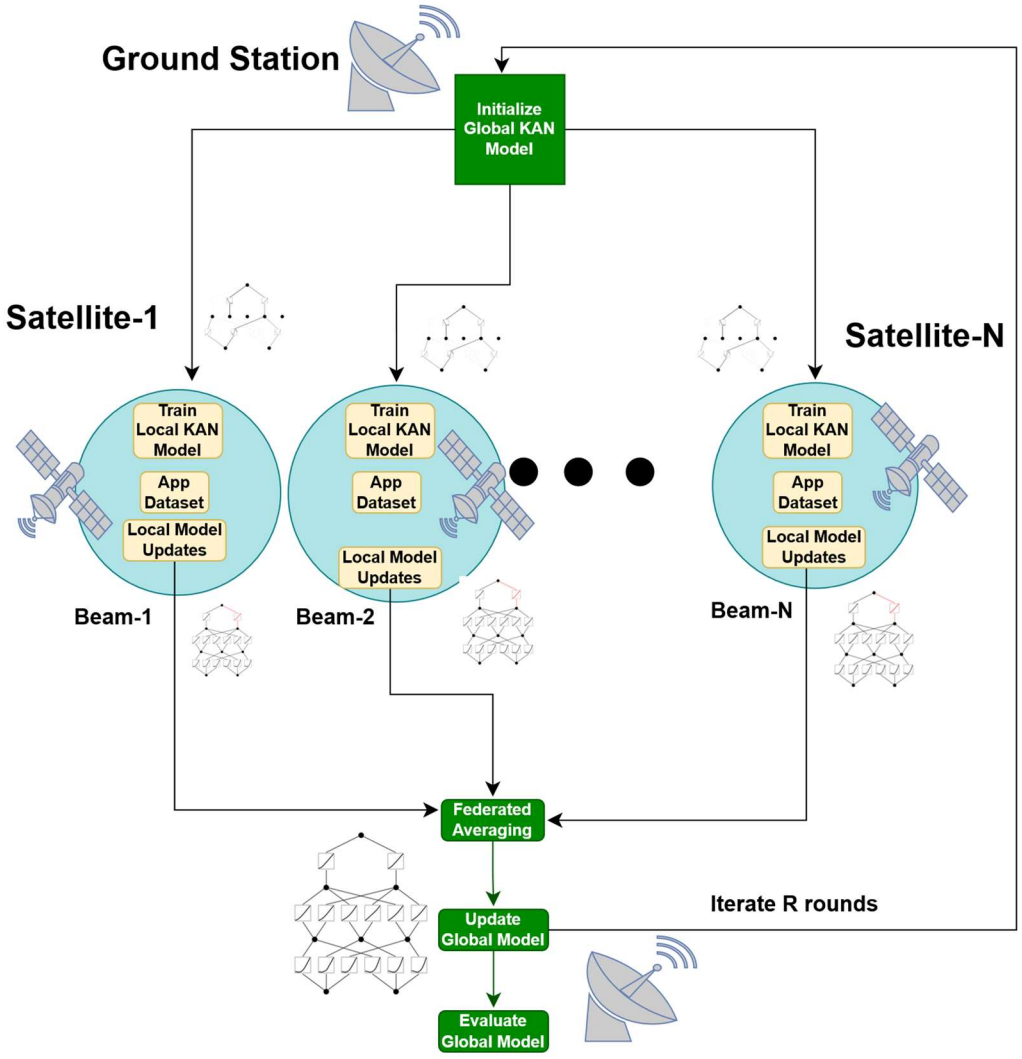


Figure 18: Federated KAN Methodology applied within NTN scenario.

3.2.4 Evaluation Results

The dataset consists of real satellite network operator network traffic data collected from four different satellite beams as detailed here [40]. Each dataset contains time-series traffic measurements, with each row representing a specific timestamp over 743 hours. The dataset is structured into two main categories: input features and target variables. The input features include Downlink and Uplink traffic values. These values indicate the amount of data being transmitted and received over the satellite beams at a given time. The target variables consist of four key categories: Communication, Streaming, Cloud Services, and System Updates. Table 5 shows various applications in each four main classes. The dataset is split in a structured way to ensure effective time series prediction while preserving the integrity of FL. For training purposes, first, input-output pairs are constructed using a lookback window of W hours. This means that for each prediction, the model uses the data of the last W hours as input to predict the values of the next hour. This creates overlapping sequences to capture temporal dependencies in the data. Next, an 80-20 split is applied between training and testing, separately for each beam (client), to ensure that 80% of the data is used for training, while

20% is reserved for testing. As this is a time series problem, the data is not shuffled. This maintains the sequential order, which is essential for accurate future predictions.

Table 5: Categorization of applications based on their primary function in the considered dataset.

Category	Applications
Communication	WhatsApp Calls, Telegram, TikTok, Facebook Video, Instagram, VKontakte Video, Google Docs, Office 365 Outlook, Facebook, Reddit, Twitter
Streaming	YouTube, YouTube Browsing, Netflix, Quic Obfuscated, Google Services, Amazon Video, Disney+, HBO MAX, Crunchyroll, Acorn TV, Pluto TV, Spotify, TikTok Live, HTTPS Streaming, EA Games, Sony PlayStation Store
Cloud Services	Google Cloud Storage, Amazon Cloud, iCloud, OneDrive, Huawei Cloud, MediaFire, Adobe Creative Cloud, Generic CDN, HTTP File Sharing, HTTP File Transfer, Google Photos, Windows Update, Apple Software Update, Steam, Amazon Alexa
System Updates	Windows Update, Apple Software Update, Akamai, Generic Web Browsing 3, Advertisements, STUN, BITS, Other UDP, NetEase Games, Epic Games Update

Model Architecture and Parameters

Figure 19 shows the architecture of the Fed-KAN classifier in a tree-like structure for a single client. The root node with the black circles is the input layer and each layer applies spline transformations (represented by integral symbols). The edges show how the transformations are connected to each other and form a hierarchical KAN. The model Fed-KAN, which builds on KANs, was developed to use piecewise polynomials and non-linear functional approximations to model complex relationships in the data. The input for our Fed-KAN and analyzed Fed-MLP models consists of the downlink and uplink traffic of the last $W = 5$ hours, which form a sliding window of historical data. The output is the predicted percentage of different application classes in the next hour. The Fed-KAN model features a width configuration of [2, 4, 8] spline nodes in hidden layers to progressively approximate nonlinear functions. The grid size was set to 5, defining the resolution of the spline transformations, while three Kolmogorov units (k=3) were used for higher-order function learning. A dropout layer with a probability of 0.5 was introduced for regularization before the final fully connected layer, which mapped the transformed features to the output predictions.

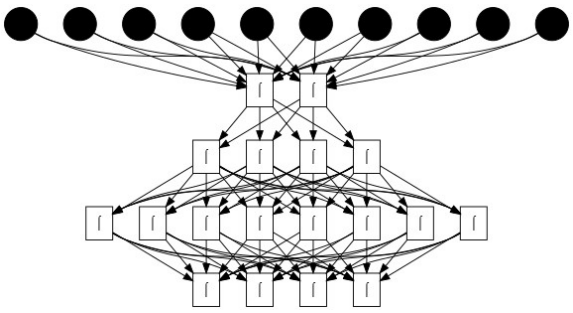


Figure 19: Fed-KAN classifier layers with input (uplink and downlink traffic of last W hours), hidden and output layers (next hour percentage prediction of traffic categories).

The Fed-MLP model followed a traditional three-layer architecture with the same hidden sizes to Fed-KAN is used but with [8, 16, 48] neurons to keep the same number of parameters (i.e. 1244). It employed ReLU activation functions in each layer, and dropout (p=0.5) was added to prevent overfitting. The fully connected output layer mapped the learned representations to the final 4-category regression task. Both models were trained using the Adam optimizer with a learning rate of 0.001 and weight decay of 1e-5, ensuring stable convergence. Additionally,

gradient clipping (max_norm=1.0) was applied to prevent gradient explosion. The models were trained over 20 rounds, with each client running 5 local epochs per round using a batch size of 16. For evaluation, MSE loss was used as the primary metric, given the regression nature of the task. The global model was validated on each client's test set after every Fed-KAN round, with the final performance measured by averaging the test losses across all clients.

Implementation details

The implementation of Fed-KAN is structured around four main stages: data preparation, model definition, federated training, and evaluation. The process begins with data preparation, where traffic datasets from multiple satellite beams are pre-processed. For training, input features, in particular uplink and downlink traffic values, are extracted together with target variables, namely communication, streaming, cloud services and system updates. The data is prepared for prediction by applying a time horizon window (e.g. $W = 5$ hours) to ensure that past traffic values can be used to predict traffic behaviour for the next 1 hour. After training, the global Fed-KAN model is evaluated with test datasets from all beams, calculating the MSE loss for performance evaluation. The model is also compared to a Fed-MLP baseline, which features a three-layer fully connected neural network with ReLU activations and dropout. Both models undergo the same federated training process, allowing a direct performance comparison. The trained models are further analyzed through prediction vs. actual value visualizations, providing insights into forecasting accuracy across different traffic categories.

Key Observations

LEO satellites regularly change satellites to maintain connectivity for users, and reliance on terrestrial gateways can lead to fluctuations depending on gateway utilization and weather conditions. Historical network traffic data are used in Fed-KAN and Fed-MLP models to anticipate future trends in application usage and optimize resource allocation in NTNs and LEO satellite communication systems. Both models analyze the past uplink and downlink traffic patterns to predict the distribution of application usage in the next hour. Figure 20 shows a comparison of the average training losses over 20 federated rounds between Fed-KAN and Fed-MLP models. The training loss for Fed-KAN decreases sharply in the first rounds and reaches a stable loss value around 0.175 after about 5 rounds, indicating that Fed-KAN converges faster and captures the patterns in the data more efficiently. In contrast, the loss for Fed-MLP gradually decreases and stabilizes at a higher loss value of around 0.22. This shows that Fed-KAN outperforms Fed-MLP in terms of learning efficiency and achieves a lower training loss across all rounds. The final training loss for Fed-KAN is about 20% lower than that of Fed-MLP, proving its superiority in FL scenarios, especially in NTN.

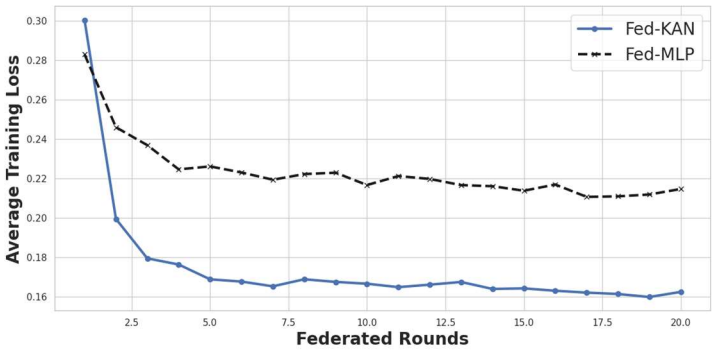


Figure 20: Average Training MSE Loss over Fed-KAN rounds.

Table 6 compares the performance of Fed-KAN and Fed-MLP based on the average test loss and the number of parameters used during training and testing. The results show that with the

same number of parameters, in the test dataset the Fed-KAN model outperforms Fed-MLP in predicting traffic utilisation by achieving a 77.39% reduction in average test loss for predicting the percentages of different application classes using the uplink and downlink data. This analysis suggests that Fed-KAN can be effective in predicting resource utilisation for satellite communication networks where dynamic topology changes and limited computational resources are the main constraints that need to be dynamically addressed.

Table 6: Comparison of federated learning models: Fed-KAN vs. Fed-MLP in the considered NTN scenario.

Model	Number of Parameters	Average Test Loss
Fed-KAN	1244	0.2583
Fed-MLP	1244	1.1433

3.2.5 Fed-KAN Applications in NTNs

Decision Engine and Actuator

Decision engines and actuators play a crucial role in optimizing the operation of satellite networks. In this phase, there could be three different deployment options that can be used to predict traffic patterns for the next hour over a sliding window period. These options include: (i) Full on-board deployment, where decision making takes place entirely on the satellite. (ii) Partial on-board deployment, where some DUs and CUs remain on the satellite while others operate remotely. (iv) No on-board deployment, where all functions are processed off-satellite. These flexible deployment strategies allow for optimal resource allocation and reduced latency for network adjustments. With overlapping satellite beams, an effective handover strategy is critical to maintaining seamless streaming traffic. If an imminent loss of coverage is predicted for a satellite traffic with Fed-KAN, the system can pre-emptively hand over the stream to another satellite or terrestrial station with overlapping coverage, thus ensuring an uninterrupted connection. This proactive approach improves the user experience and optimizes bandwidth usage. This makes Fed-KAN an important component of the next generation of satellite communications.

Discussions on Split Functions in O-RAN

Given the distributed and privacy-sensitive nature of NTN, Fed-KAN can be leveraged as an AI-driven intelligence model for optimizing NTN operations while preserving data privacy across different NTN nodes. The proposed Fed-KAN algorithm can be used within O-RAN RICs at various layers. Fed-KAN deployment can also be enabled as an xApp or rApp, ensuring dynamic, decentralized, and privacy-preserving AI/ML model training across the network. The placement of Fed-KAN within the O-RAN NTN systems depend on the architectural and functional split strategies, which are analysed in detail in Section 2, The most suitable architecture options are identified and further elaborated below.

- Option 1: Near-RT RIC and CU on Ground, DU+RU On-Board (Satellite)
- Option 2: Near-RT RIC and Full gNB On-Board (Satellite)

To summarize, Fed-KAN-enabled Non-RT-RIC applications deployed fully on ground can enable host the aggregator model, execute FL aggregation, and optimize policies based on long-term traffic patterns. However, it can suffer from high latency. In contrast, Fed-KAN-enabled near-RT-RIC applications deployed entirely on the satellite model can enable the FL

process within the satellite network itself by utilising the ISLs and on-board computing resources, rather than relying on ground-based server interaction for model aggregation. In this way, model aggregation can be done without frequent ground communication, allowing faster FL iterations which can be more suitable for LEO NTN systems. In the meantime, it should be noted that ISLs are still not as advanced as ground communication and some satellites still cannot communicate directly with each other and rely on ground stations communications. In such scenarios, hybrid FL approaches and configurations can be used, where the satellites can train local Fed-KAN models on board and perform partial updates. Later, the aggregation is partially performed on a lead satellite before the updates are sent to the ground-based non-RT RIC for final aggregation. The final global model is later distributed back to the satellites and ground networks.

3.3 MULTIVARIATE PROBABILISTIC FORECASTING FOR MULTIBEAM SATELLITES

The increasing complexity of modern multi-beam satellite systems demands advanced approaches for RRM that can effectively capture and exploit high-dimensional interactions across beams. Withing this context, AI/ML has emerged as a powerful tool for predictive modelling, enabling more accurate and adaptive resource allocation strategies. This work presents a quantitative assessment of multivariate AI-based prediction techniques for RRM in multi-beam satellites, systematically benchmarking their performance against traditional single-beam prediction approaches. Both probabilistic and single-point prediction methods are considered, allowing for a comprehensive evaluation of accuracy, robustness, and uncertainty representation. By comparing these methodologies, the study aims to highlight the potential gains of multivariate AI models in enhancing RRM efficiency and reliability in next-generation satellite communication systems.

3.3.1 Shifting from single-point predictions to multi-beam probability forecasts of demands

Classical data-driven, single-point AI-based models (e.g. LSTM [41] or GRU [42]) in resource management typically provide deterministic point forecasts, offering a single predicted value for the multi-step ahead time steps in the prediction window. While computationally efficient and straightforward to interpret, these approaches often induce a false sense of confidence by failing to capture the inherent uncertainty and variability present in complex, dynamic environments such as multi-beam satellite systems. In contrast, probabilistic forecasting techniques address these limitations by generating a range of possible outcomes rather than a single estimate. This enables explicit modelling and quantification of uncertainty, making predictions more robust to outliers, anomalies, and non-stationary behaviour in the underlying data. Moreover, probabilistic methods enhance the ability to anticipate rare or extreme events, thereby supporting risk.

Limitations of traditional single-point, univariate AI-based models

Limitation 1: They fail to capture uncertainty. Single-point estimators (like standard LSTMs) provide a "best guess" predicted value, based on some metric, such as for instance the MSE (Mean Square Error). Unfortunately, In SATCOM RRM, being wrong by even a small margin can lead to one side, an unmet capacity demand or degraded Quality of Service (QoS) or, on the order side, underutilized capacity.

Limitation 2: They ignore inter-dependencies amongst different time series. Univariate models analyze and learn from each resource stream (i.e., beam) in isolation, ignoring the complex inter-beam dependencies. As a consequence, deterministic single-point estimators are ineffective, due to their inability to capture multivariate dependencies and uncertainties effectively.

Multivariate probabilistic forecasting for multi-beam satellites

Instead of asking *"What will the demand be?"*, in multi-beam satellites we need to ask *"What is the probability distribution of future demands of resources for the multiple beams?".* This fundamental shift allows for more adaptive and reliable resource management. Here is the motivation for moving towards **Multivariate Probabilistic Forecasting**:

- **Capturing complex inter-beam demands dependencies**: Within the context of modern multi-beam satellite systems, the transition from traditional univariate models to state-of-the-art multivariate probabilistic forecasting is not just a technical upgrade, but also a necessity to capture complex inter-beam dependencies. Traditional univariate models analyse each beam's resource demand in isolation. This approach misses the complex correlations between multiple beams resource demands.

- **Quantification of uncertainty in the demands of resources of the multiple beams:** Provides joint confidence interval estimations (e.g., 50% and 95% prediction ranges) for every forecast in the multi-step ahead time steps in the horizon forecast.

- **Enable Risk-Aware Decisions:** Satellite operators can make informed trade-off decisions based on probabilistic guarantees, aligning resource allocation with specific risk tolerances and Service Level Agreements (SLAs).

- **Resource efficiency and robust decision making:** Probabilistic models allow for risk-aware resource management. Satellite operators can decide exactly how much risk they are willing to take (e.g., a 1% chance of congestion) and tune the satellite payload parameters to meet that specific service-level agreement (SLA).

Figure 21 illustrates the multivariate probabilistic forecasting concept in multi-beam satellites. The multi-beam radio resource allocation problem can be formulated as multivariate series. Let

$$\mathbf{D}_t = [D_{1,t}, D_{2,t}, \dots, D_{N,t}],$$

where $D_{b,t}$represents the demand in beam b at time t. The sequence forms a multivariate time series because multiple beams $\{b_b\}_{b=1}^{N}$ demands, which evolve jointly over time. This formulation captures not only the multi-beam dependencies (e.g., correlation between beams), but also the temporal dynamics (i.e., trends, seasonality, etc.). Within this context, instead of predicting a single value, we want to predict a joint probability demand distribution over future demand based on the dataset of historical observations $\mathcal{D}_t$:

$$P(\boldsymbol{D}_{t+1:t+H} \mid \mathcal{D}_t, \boldsymbol{x_{t+1:t+H}}),$$

where H denotes multiple steps ahead in the forecast horizon, $\boldsymbol{D}_{t+1:t+H} \in \mathbb{R}^{H\times N}$ is the multi-beam demand for the forecast horizon and $\boldsymbol{x_{t+1:t+H}}$ denotes a set of exogenous variables, the so-called covariates, such day, time, etc. This approach captures the demand evolution in the next *H* time steps in of the prediction window, dependencies across the beams and uncertainty for the different time steps in the horizon window.

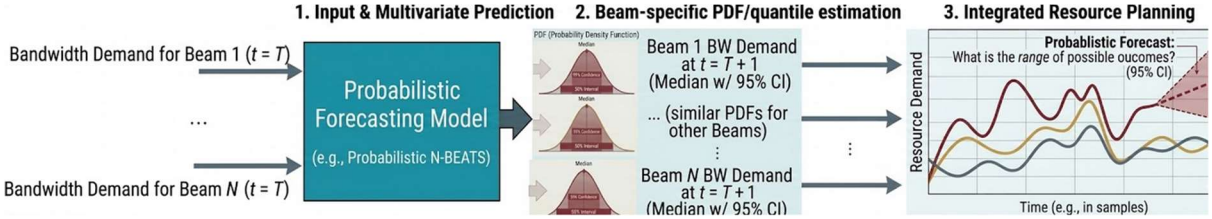


Figure 21. Illustration of the multivariate probabilistic concept in multibeam satellites.

3.3.2 Multivariate Deep Time Series Forecasting techniques

This section presents the state-of-the-art multivariate AI-based methods that will be utilized for the performance assessment in the Evaluation Results section.

3.3.2.1 N-BEATS

N-BEATS (Neural Basis Expansion Analysis for Interpretable Time Series Forecasting) is a state-of-the-art deep neural architecture designed specifically for time series forecasting. Introduced by B. N. Oreshkin in [43], it is notable for being a "pure" deep learning model relying entirely on Fully Connected (FC) layers rather than Recurrent Neural Networks (RNN) or convolutional (CNN) structures and clearly outperforms traditional non-AI-based statistical methods, like ARIMA and ETS. It relies on stacks of fully connected blocks combined through residual learning and basis expansion. Unlike recurrent models such as LSTMs, N-BEATS does not rely on sequential processing, making it highly parallelizable and highly scalable. The lack of recurrent connections allows for massive parallelization during training. N-BEATS is a hierarchical, purely dense neural network that decomposes signals through successive residual stages. Its primary innovation lies in its ability to provide mathematical interpretability (via trend and seasonality decomposition) without sacrificing the performance gains typically associated with deep "black-box" models.

Next Figure shows the N-BEATS architecture. The basic building block is typically a 4-layer Fully Connected (FC) network with ReLU (Rectified Linear Unit) activation functions. The architecture is composed of a sequence of blocks indexed by $c = 1, \dots, K$. Each block receives a residual signal $\mathbf{x}^{(c)}$and outputs two components: a backcast $\hat{\mathbf{x}}^{(c)}$, which reconstructs part of the input, and a forecast $\hat{\mathbf{y}}^{(c)}$, which contributes to the final prediction.

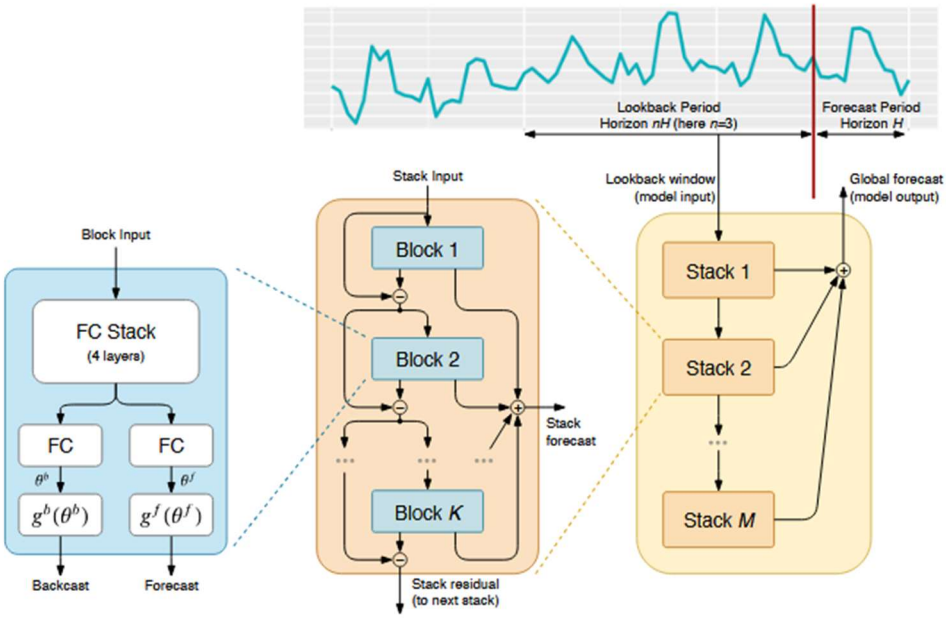


Figure 22. Basic architecture of N-BEATS. Extracted from [43].

The residual is updated iteratively as:

$$\mathbf{x}^{(c+1)} = \mathbf{x}^{(c)} - \hat{\mathbf{x}}^{(c)}, \hat{\mathbf{y}} = \sum_{c=1}^{K} \hat{\mathbf{y}}^{(c)}.$$

This residual stacking mechanism ensures that each block focuses on components of the signal not explained by previous ones. Each block is implemented as a 4-layer multilayer perceptron $f^{(c)}(\cdot)$that maps the input residual to a parameter vector:

$$\boldsymbol{\theta}^{(c)} = f^{(c)}(\mathbf{x}^{(c)}).$$

The outputs are then constructed through a basis expansion:

$$\hat{\mathbf{x}}^{(c)} = \mathbf{W}_{\text{back}}^{(c)} \boldsymbol{\theta}^{(c)}, \quad \hat{\mathbf{y}}^{(c)} = \mathbf{W}_{\text{fore}}^{(c)} \boldsymbol{\theta}^{(c)},$$

where $\mathbf{W}_{\text{back}}^{(c)}$and $\mathbf{W}_{\text{fore}}^{(c)}$are basis matrices that can be either learned (generic configuration) or fixed (interpretable configuration). In the interpretable variant, the basis functions are chosen to reflect classical time-series components, such as polynomial bases for trend and Fourier bases for seasonality, yielding an additive and interpretable decomposition of the forecast. In the generic variant, the basis is learned end-to-end without structural constraints prioritizing predictive performance. N-BEATS organizes blocks into stacks. The stacks use a hierarchical residual approach:

- Backcast Residual Path: Block receives the input minus the backcasts of all previous blocks. This forces each block to focus only on the "residual" signal that previous blocks could not explain.
- Forecast Aggregation: The final forecast is the sum of the partial forecasts produced by every block in every stack.

N-BEATS supports **multivariate forecasting by flattening the model inputs to a 1-D series and reshaping the outputs to a tensor of appropriate dimensions**.

Loss function: The single-point forecast version (i.e., the deterministic version) is trained to minimize the Mean Squared Error (MSE). In the probabilistic counterpart of N-BEATS, the training is performed by minimizing the Negative Log-Likelihood (NLL) loss, which allows the model to learn the full predictive distribution rather than producing only point forecasts. Herein, we have considered Laplacian distribution. It has heavier tails than Gaussian distribution and is more robust to outliers. For the sake of simplicity, the explicit mathematical formulation of the Laplacian NLL is omitted.

Remark on notation: Throughout this deliverable, the following notation is adopted. ***N-BEATS*** denotes the univariate deterministic version of the N-BEATS algorithm. ***N-BEATS-P*** refers to its univariate probabilistic extension, trained by minimizing the Negative Log-Likelihood (NLL) under a Laplacian assumption. ***N-BEATS-P-MV*** designates the multivariate probabilistic forecasting variant, optimized with respect to the multivariate Laplacian NLL.

3.3.2.2 N-HITS

The N-HiTS (Neural Hierarchical Interpolation for Time Series) model is a feedforward deep learning architecture, introduced in [44] designed for accurate and efficient time series forecasting. It extends the N-BEATS framework by incorporating hierarchical interpolation and multi-resolution analysis to better capture both short- and long-term **temporal patterns. It is designed to overcome limitations of additive and residual-based forecasting** models such as N-BEATS. While N-BEATS decomposes a time series into backcast and forecast components through residual stacking, N-HiTS replaces this paradigm with a fully modular hierarchical interpolation mechanism.

Figure 23, extracted from [44] presents the N-HITS architecture. This design reduces the memory and computational requirements, while promoting architectural parsimony and improving forecast accuracy. The model consists of multiple multilayer perceptrons (MLPs) with ReLU activations, organized into blocks connected through a doubly residual stacking scheme, where each block produces both backcast $\tilde{\boldsymbol{y}}_{t-L:t}^{(\ell)}$ and forecast $\hat{\boldsymbol{y}}_{t+1:t+H}^{(\ell)}$ outputs. The combination of multi-rate input pooling, hierarchical interpolation, and backcast residual connections encourages the specialization of predictions across different frequencies.

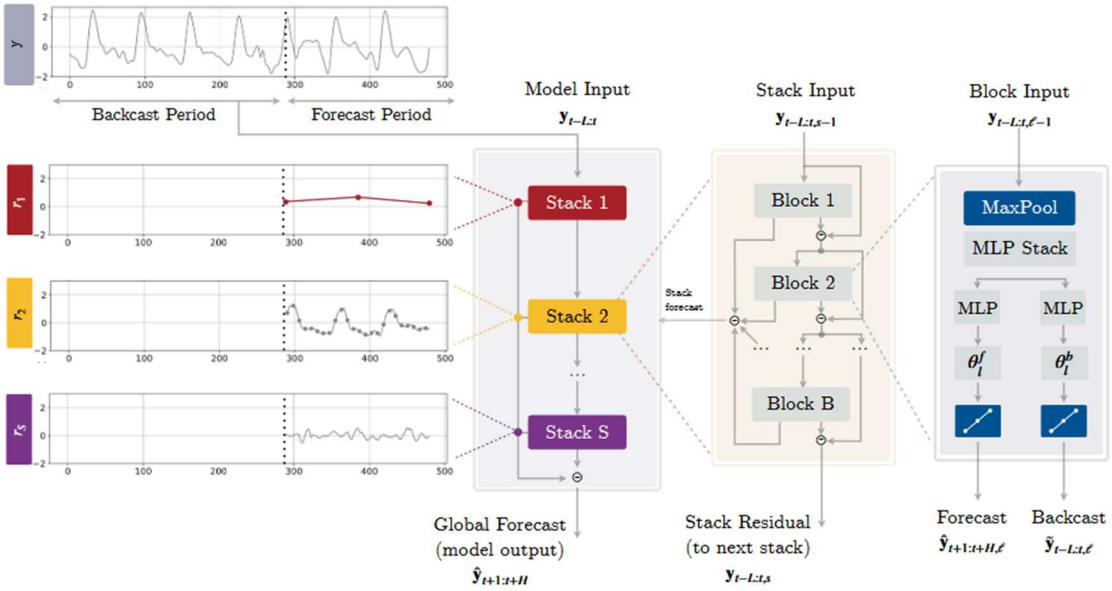


Figure 23. N-HITS architecture (extracted from [44]).

Given an input window $\mathbf{x} \in \mathbb{R}^L$, the goal is to predict a horizon $\hat{\mathbf{y}} \in \mathbb{R}^H$. Instead of iteratively refining residuals, N-HiTS partitions the forecast horizon into B disjoint subintervals:

$$\hat{\mathbf{y}} = \text{concat}\,(\mathbf{f}^{(1)}, \mathbf{f}^{(2)}, \ldots, \mathbf{f}^{(B)}),$$

where each block independently predicts a segment $\mathbf{f}^{(i)} \in \mathbb{R}^{H_i}$, with $\sum_{i=1}^{B} H_i = H$.

Each block operates on a down sampled representation of the input: $\tilde{\mathbf{x}}^{(i)} = P_{k_i}(\mathbf{x})$, where $P_{k_i}(\cdot)$ is a pooling operator controlling temporal resolution. A fully connected network produces coefficients $\boldsymbol{\theta}^{(i)}$, which are mapped to the forecast segment via a learnable interpolation function $\mathcal{I}(\cdot)$: $\mathbf{f}^{(i)} = \mathcal{I}_{H_i}(\boldsymbol{\theta}^{(i)})$. Unlike residual-based models, the absence of iterative error correction prevents error accumulation across blocks, reduces memory usage, and enables flexible control of temporal resolution. Furthermore, instead of relying on fixed basis expansions, N-HiTS employs learnable interpolation layers that map low-resolution representations to high-resolution forecasts, improving adaptability to complex temporal dynamics. This architecture makes N-HiTS particularly effective for long-horizon and high-resolution forecasting tasks, where it has demonstrated superior performance on multiple benchmark datasets.

Similar to NBEATS, **N-HITS handles multivariate forecasting by flattering inputs of the model to a 1-D series and reshaping the outputs to a tensor of appropriate dimensions**. However, it specifically utilizes Multi-Rate Sampling to reduce the input dimensionality of each stack, making it more computationally efficient when processing multiple variables over long windows.

Training loss functions

- **Deterministic Version:** The standard model is typically trained using Mean Absolute Error (MAE) or Mean Squared Error (MSE).

- **Probabilistic Version:** N-HITS minimizes the Negative Log-Likelihood to estimate parameters of probability distribution. Herein, we have considered Laplacian distribution. It has heavier tails than Gaussian distribution and is more robust to outliers. For the sake of simplicity, the explicit mathematical formulation of the Laplacian NLL is omitted.

Remark on notation: Throughout this deliverable, the following notation is adopted. ***N-HITS*** denotes the univariate deterministic version of the N-HITS algorithm. ***N-HITS-P*** refers to its univariate probabilistic extension, trained by minimizing the Negative Log-Likelihood (NLL) under a Laplacian assumption. ***N-HITS-P-MV*** designates the multivariate probabilistic forecasting variant, optimized with respect to the multivariate Laplacian NLL.

3.3.2.3 Temporal Fusion Transformer (TFT)

Developed through a collaboration between Google Cloud AI and the University of Oxford, the Temporal Fusion Transformer (TFT) [45] leverages self-attention mechanisms to capture long-term dependencies and model complex temporal dynamics in multi-time series forecasting tasks. The model addresses a central challenge in modern forecasting problems: how to effectively combine heterogeneous inputs, such as static metadata, historical observations, and known future variables, while producing accurate and interpretable probabilistic predictions over multiple future time steps. Its primary strengths include:

- **Support for multiple time series:** Efficiently processes thousands of concurrent univariate or multivariate time series.
- **Probabilistic Multi-Horizon Forecasting:** Generates multi-step predictions for target variables, complemented by robust prediction intervals for uncertainty quantification.
- **Heterogeneous Data integration:** Seamlessly incorporates a variety of inputs, including static metadata and time-varying exogenous variables.

At its core, TFT integrates recurrent neural networks, attention mechanisms, and specialized gating components into a unified architecture that balances performance and interpretability. Traditional sequence models such as recurrent neural networks or Long Short-Term Memory networks (LSTMs) are effective at capturing local temporal dependencies but struggle with long-range relationships and often act as black boxes. Transformer-based models, on the other hand, excel at modelling long-term dependencies through attention but can be computationally expensive and less tailored to structured time series inputs. TFT combines the strengths of both paradigms by using LSTM layers for local sequence encoding and a self-attention mechanism for selectively focusing on relevant time steps when making forecasts.

The TFT architecture is underpinned by several sophisticated components designed for adaptability and interpretability:

- **Gating mechanisms:** These components facilitate adaptive depth by allowing the model to bypass non-essential architectural pathways, effectively tailoring complexity to the specific requirements of the dataset.
- **Variable selection networks**, providing instance-wise selection of relevant features, dynamically weighting the most significant inputs at each discrete time step.
- **Static covariate encoders:** These modules transform static metadata into context vectors, which serve to condition and modulate the model's temporal dynamics.

- **Hybrid temporal processing:** To capture multi-scale dependencies, the model utilizes a sequence-to-sequence layer for local patterns and an interpretable multi-head attention block to isolate long-range relationships.

- **Probabilistic Forecasting:** The model outputs quantile-based intervals, providing a robust estimation of the target variable's distribution across the forecast horizon.

Interpretability is a central design goal of TFT and is achieved through several mechanisms embedded in the architecture. Variable importance can be assessed through the outputs of the variable selection networks, revealing which features contribute most to predictions across time. Temporal attention weights provide insights into which historical periods are most relevant for forecasting future values. Additionally, the separation of static and time-varying components allows practitioners to analyze how fixed attributes influence dynamic behaviour. This level of transparency distinguishes TFT from many other deep learning approaches and makes it more suitable for applications where model explainability is required.

TFT is designed for probabilistic forecasting. It typically predicts multiple quantiles of the target distribution, enabling the construction of prediction intervals and providing a measure of uncertainty. This is particularly valuable in decision-making contexts where risk assessment is critical. The training objective is usually based on quantile loss, which encourages accurate estimation of different parts of the conditional distribution rather than just the mean.

Loss function: TFT is trained by jointly minimizing the quantile loss, summed across all the quantile outputs and all the beams.

$$L(\Omega, \boldsymbol{W}) = \sum_{i=1}^{N} \sum_{y \in \Omega} \sum_{q \in Q} \sum_{\tau=1}^{T_{max}} \frac{QL\left(D_{i,t}, \widehat{D}_{i,t}(q, t-\tau, \tau), q\right)}{NH|Q|},$$

where Ω denotes the domain of training data, **W** represents the weights of TFT model, Q is the set of output quantiles, and $(\cdot)_+ = \max(0,\cdot)$. H denotes the number of time steps in the multiple step ahead horizon forecast and *N* denotes the number of Beams.

Remark on notation: Throughout this deliverable, the following notation is adopted. ***TFT*** refers to its univariate probabilistic version, trained by minimizing the average quantile loss. ***TFT-MV*** designates the multivariate probabilistic forecasting variant.

3.3.3 Evaluation results

This section presents the evaluation results of bandwidth (BW) allocation strategies using the Hispasat dataset described in D4.1 [40], which captures residential broadband demand across six satellite beams in Latin America. We compare a baseline static allocation approach against the dynamic allocation strategies driven by demand AI-based forecasting models.

Reference scenario

In the **baseline scenario**, each beam is provisioned with a **fixed bandwidth** determined as a constant margin above its historical peak demand. While simple and robust, this approach does not adapt to temporal variations in traffic and may lead to systematic over-provisioning or underutilization of the radio resources.

In this baseline configuration, each satellite beam is assigned a fixed bandwidth defined as a constant margin above its historical peak demand. This value is interpreted as the maximum capacity that the beam can sustain. The allocated bandwidth $D_{\text{static}}$for beam $b$is expressed as:

$$D_{\text{static}}(b) = \alpha \cdot \max_t D_b(t),$$

where $D_b(t)$ represents the demand observed in the *b*-th beam at time t and $\alpha = 1.15$ is a fixed overprovisioning factor (i.e., 115%). This factor is selected to approximate the theoretical upper bound of the bandwidth capacity of each beam. This configuration emulates the behaviour of a system with fixed power or transponder allocation. While it ensures robustness under peak demand conditions, it typically **leads to substantial underutilization during periods of low demand**. As such, static allocation serves as a conservative reference case for assessing the efficiency improvements achievable through dynamic allocation strategies.

Dynamic allocation (AI-driven case)

In contrast to the static scenario, the **dynamic allocation framework** adjusts bandwidth assignments based on predicted traffic demand generated by the different aforementioned model-driven forecasting techniques. This approach enables a more responsive and potentially efficient use of resources, as allocations can track demand fluctuations over time. **For probabilistic forecasting methods, bandwidth provisioning is derived from selected demand quantiles (e.g., P99)**, ensuring that allocation decisions account for uncertainty and provide a desired level of service reliability.

In this adaptive configuration, bandwidth allocation per beam is determined based on demand predictions generated by each the forecasting models. For each prediction horizon, the dynamically allocated bandwidth D_{dyn} is defined as:

$$D_{\text{dyn}}(b,t) = \min\left(\widehat{D}_b(t), C_{\text{max}}(b)\right),$$

where $\widehat{D}_b(t)$ denotes the forecasted demand for beam b at time t. In the case of point forecasting models, $\widehat{D}_b(t)$ corresponds to a single-value estimate, whereas for probabilistic models it represents a selected upper quantile (e.g., P99). The term $C_{\text{max}}(b)$ indicates the maximum physical bandwidth available for beam b. Herein, we consider that the maximum physical bandwidth available per beam is the one corresponding to the static allocation, i.e., D_{static}. This scenario reflects a flexible payload configuration in which bandwidth resources are dynamically adapted to anticipated demand, while they remain constrained by hardware limitations. By design, it achieves a balance between service reliability, ensured through conservative (i.e., high-quantile) demand estimates with probabilistic forecasts, and spectral efficiency, by mitigating persistent overprovisioning. Throughout the section, **99-th percentile is considered for probabilistic models**.

The following results evaluate how these approaches compare in terms of efficiency, service level adherence and robustness under varying traffic conditions.

Considered hyperparameters

Context window *L*= *7* days (1 week) with 1 hour sampling, horizon window 1 day (with 1 hour sampling *H*=24h). Hyperparameter tuning has been implemented for the different models.

N-BEATS: The N-BEATS model is configured with a generic architecture consisting of 30 stacks, each containing a single block with 4 fully connected layers and a uniform width of 256 units. This deep hierarchical structure employs a doubly residual learning mechanism to iteratively refine forecasts by processing the backcast residuals across all 120 total layers. Training is executed over 100 epochs with a batch size of 800.

N-BEATS-P: # stacks =60, # blocks =2, # layers=8, # epochs=100, batch size = 32.

N-BEATS-P-MV: #stacks=60, #blocks=2, #layers=8, epochs=100, batch size=32.

Loss function for training: 1) **N-BEATS**: MSE, 2) **N-BEATS-MV**: MSE 2) **N-BEATS-P**: Laplacian NLL, 3) **N-BEATS-P-MV**: Multivariate Laplacian NLL.

N-HITS: Number of stacks =6, Number of blocks =2 with 4 fully-connected layers per block, 100 epochs, dropout=0.1, activation='ReLU'.

Loss function for training: 1**) N-HITS**: MSE, 2) **N-HITS-MV**: MSE, 3) **N-HITS-P**: Laplacian NLL, 3) **N-HITS-P-MV**: Multivariate Laplacian NLL.

TFT: hidden size=128, layers=2, attention heads=8, dropout=0.1, batch size=32, epochs=100.

Loss function for training: **TFT** and **TFT-MV**: Average quantile loss.

Numerical results

Next, we assess the performance of the multivariate probabilistic forecasting techniques for multi-beam satellites in dynamic allocation, when compared to univariate techniques, both deterministic and probabilistic. Figure 24 below shows the Pareto trade-off between bandwidth savings and under-provisioning risk for the evaluated AI/ML techniques.

The y-axis represents the percentage of bandwidth savings, defined as the relative reduction in allocated capacity when transitioning from a static (fixed) allocation strategy to a dynamic allocation scheme. This can be expressed as:

$$\text{BW Saving}_{\text{alloc}}(\%) = \left(1 - \frac{D_{dyn}}{D_{static}}\right) \cdot 100.$$

The x-axis shows the under-provisioning rate, defined as the fraction of time instances in which the predicted allocated bandwidth under the dynamic scheme falls below the observed demand. This metric is averaged across all beams and all the time steps within the multi-step forecasting horizon, and is given by:

$$Underprov(\%) = \frac{\sum_b \sum_t \mathbf{1}\{\widehat{D}_b(t) < D_b(t)\}}{NH} \cdot 100,$$

where $\mathbf{1}\{\cdot\}$is an indicator function that equals 1 when the predicted bandwidth is lower than the actual demand, and 0 otherwise. *N* denotes the number of beams and *H* is the number of time steps in the horizon forecast. This metric captures the proportion of instances in which demand is not met with the dynamic allocation scheme, thereby indicating potential Quality of Service degradation and violations of Service Level Agreements.

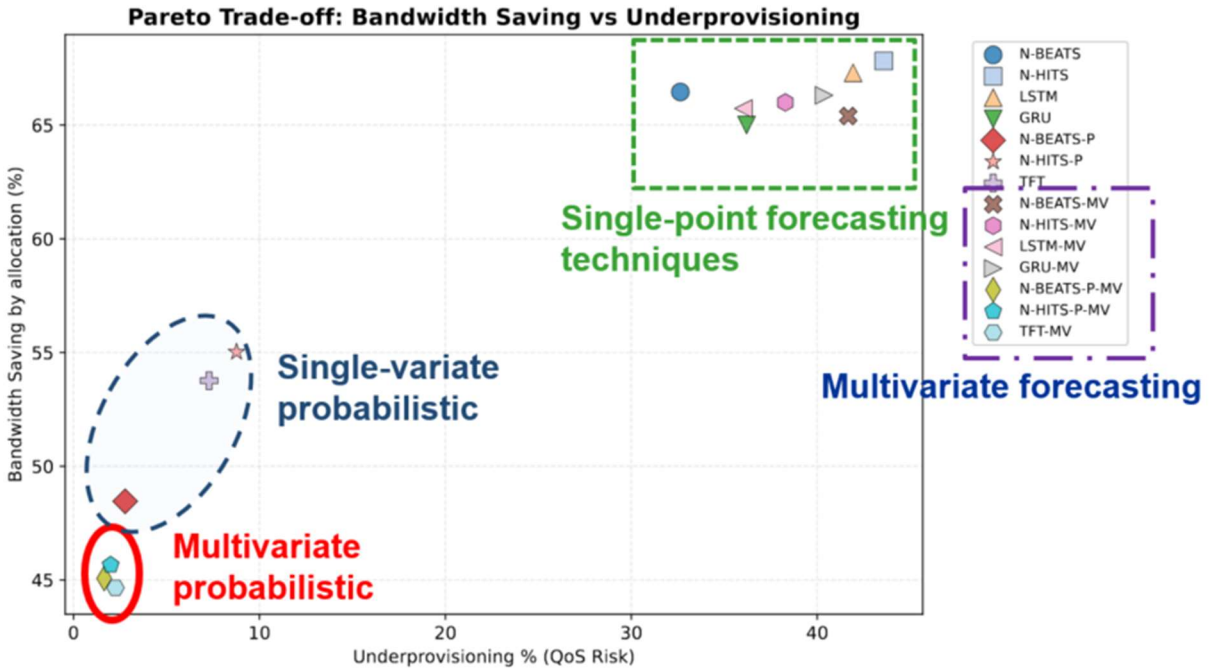


Figure 24. Pareto trade-off between bandwidth saving and under-provisioning (QoS risk)

As shown in Figure 24 above, deterministic models, *both univariate (LSTM, GRU, N-BEATS, N-HITS) and multivariate (N-BEATS-MV, N-HITS-MV, LSTM-MV, GRU-MV), cluster on the high-saving high-risk corner (upper corner, right side)*. This is mainly due to the fact these methods are trained to minimize the Mean Squared Error. On the contrary, probabilistic techniques occupy the low-risk, moderate-saving region, reducing significantly, the probability of under-provisioning ratio (i.e., the probability of not serving the demand). Interestingly, within probabilistic techniques (99th percentile), **AI-based multivariate probabilistic models (N-BEATS-P-MV, N-HITS-P-MV and TFT-MV) reduce overprovisioning compared to single-variate counterparts (N-BEATS-MV, N-HITS-MV, TFT)**, **reducing significantly the under-provisioning rate** of resources. Therefore, multivariate probabilistic techniques exhibit a **good trade-off: significant BW saving (~45%) and reduced under-provisioning rate (less than 2%).** Since probabilistic techniques are trained to optimize probabilistic losses (average quantile loss and NLL), better performance guarantees can be achieved in terms of the served demand.

Figure 25 presents two key operational metrics for the multi-beam satellite scenario: (i) the mean percentage of bandwidth savings achieved through dynamic allocation across the evaluated AI/ML techniques, and (ii) the corresponding under-provisioning ratio. The left-hand side of the figure depicts univariate approaches (including both single- and multi-variate variants), while the right-hand side shows multivariate techniques.

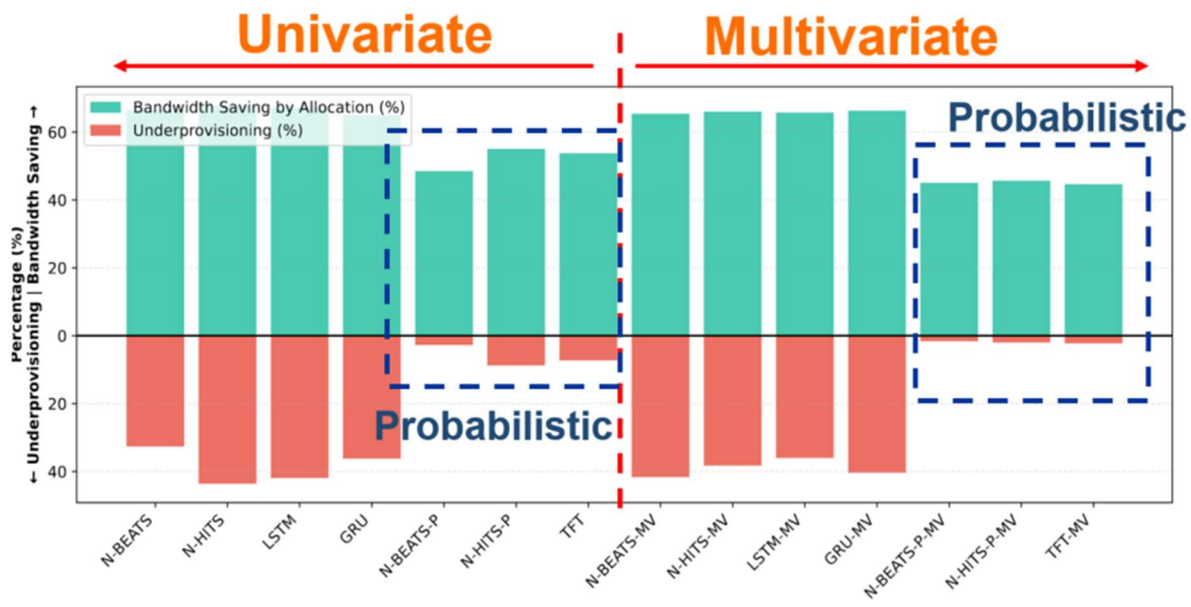


Figure 25. Mean Bandwidth saving and underprovisioning ratio for the assessed techniques.

Probabilistic methods, both univariate and multivariate, consistently reduce under-provisioning, thereby improving Quality of Service (QoS) and compliance with Service Level Agreements (SLAs), albeit at the expense of lower bandwidth savings. This behaviour stems from the fact that deterministic approaches optimize point-error metrics (e.g., MSE or MAE), whereas probabilistic models leverage distributional information, such as high quantiles (e.g., 99th percentile), enabling more conservative and informed allocation decisions. Moreover, **multivariate probabilistic models (i.e., N-BEATS-P-MV, N-HITS-P-MV, TFT-MV) further decrease under-provisioning compared to their univariate counterparts, achieving a more favourable trade-off between bandwidth savings and service reliability.** For the sake of simplicity and to improve the clarity of the figure, P-KAN was omitted from the visual representation. However, Figure 10 demonstrates that P-KAN is able to achieve an under-provisioning performance comparable to the other univariate probabilistic techniques, while maintaining a highly lightweight architecture, potentially suitable for on-board satellite processing. The extension of P-KAN to provide multivariate probabilistic forecasts is future work.

Let us now better analyze the performance of the probabilistic forecasting techniques more in detail. Figure 26 evaluates the ability of different forecasting models to capture extreme upper-tail behaviour, specifically the 99th percentile, using the quantile loss at $q = 0.99$ as the performance metric. Lower quantile loss values indicate better tail-risk coverage and more accurate estimation of extreme demand.

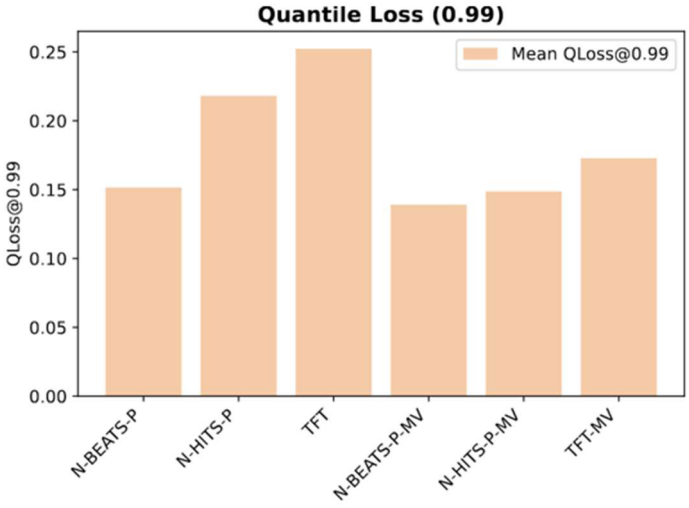


Figure 26. Quantile loss for 99-th percentile.

The results show that univariate models (e.g., N-BEATS-P, N-HiTS-P, TFT) exhibit higher quantile loss, indicating a systematic underestimation of tail events and a failure to adequately capture extreme demand fluctuations. **In contrast, all multivariate variants (denoted as "-MV") achieve lower quantile loss values, demonstrating improved calibration of the upper tail.**

Among the evaluated approaches, multivariate models consistently outperform their univariate counterparts, confirming the benefit of incorporating cross-beam dependencies. In particular, **multivariate probabilistic methods provide the best performance, as they leverage joint information across beams to better anticipate extreme events.** Overall, the figure highlights that multivariate conditioning significantly enhances extreme-quantile estimation, reducing tail risk and improving the reliability of forecasts in high-demand scenarios. Among all probabilistic the models, N-BEATS-P-MV provides the best performance.

4 AI TECHIQUES FOR NEAR-REAL TIME RRM

In this Section, we discuss the application of Artificial Intelligence (AI) and Machine Learning (ML) techniques in the near-real time RIC to support digital user-centric beamforming or other elements on-board the NTN node(s). In particular: i) we design a clustering-based scheduling algorithm for user-centric beamforming, aimed at reducing the computational complexity of the schedulers developed in [2] while maintaining similar performance; and ii) we design a Neural Network (NN) aimed at estimating the Signal-to-Interference-plus-Noise Ratio (SINR) of each user scheduled in the same time slot after the application of user-centric beamforming.

4.1 SYSTEM MODEL

The system model is extensively detailed in D4.6 [2], ,for the standalone NTN node, but for the sake of clarity and self-consistency, we report in the following the most relevant assumptions and observations.

We consider the Distributed Beamforming Management (DBFM) architecture with Option 4 (or higher layer) functional split solutions: the Radio Resource Management (RRM) and the computation and application of the beamforming coefficients are all performed on-board the NTN node serving the target service area. The N_{UE} users in the service area have each a different traffic request, denoted as $C_k^{(REQ)}$, $k = 1, \dots, N_{UE}$, obtained according to one of the methodologies designed and discussed in Section 4.2 of [2].

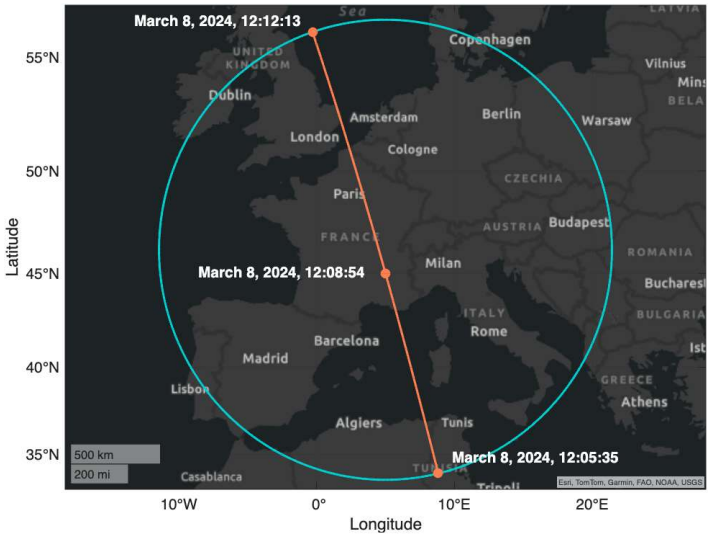


Figure 27: Service area and ground track for standalone user-centric beamforming with $(lat_S = 45°N, lon_S = 5°E)$, $\varepsilon_S = 30°$, $h_{sat} = 1000$ km (5G-STARDUST polar constellation, $i = 99°$).

The NTN node is equipped with the Direct Radiating Array (DRA) containing $N_{R,sat} = 512$ radiating elements[1], designed in WP3, [26]. The service area is defined so as to guarantee that when the NTN node is above its centre, the elevation angle for the users at the edge is

[1] In this document, we consider standalone NTN nodes. Thus, the number of radiating elements per NTN node is equal to the total number of radiating elements from a potential swarm (see [2]), *i.e.*, $N_R = N_{R,sat}$.

$\varepsilon_S = 30°$. Figure 27, reported here from [2], depicts the considered service area and the corresponding NTN node ground track.

In a reporting time slot, the NTN node receives ancillary information from the visible users: the estimated Channel State Information (CSI) for CSI-based beamforming and the estimated location for location-based beamforming. The channel coefficient in a generic time slot t between the i-th on-ground UE and the n-th on-board radiating element can be computed as:

$$\mathrm{h}_{i,n}^{(t)} = \frac{\mathrm{g}_{i,n}^{(tx,t)}\mathrm{g}_{i,n}^{(rx,t)}}{\frac{4\pi \mathrm{d}_i^{(t)}}{\lambda}\sqrt{L_i^{(t)}\kappa B_i T_i}} e^{-j\frac{2\pi}{\lambda}\mathrm{d}_i^{(t)}}, i = 1, \dots, N_{UE}, n = 1, \dots, N_R \quad (1)$$

where:

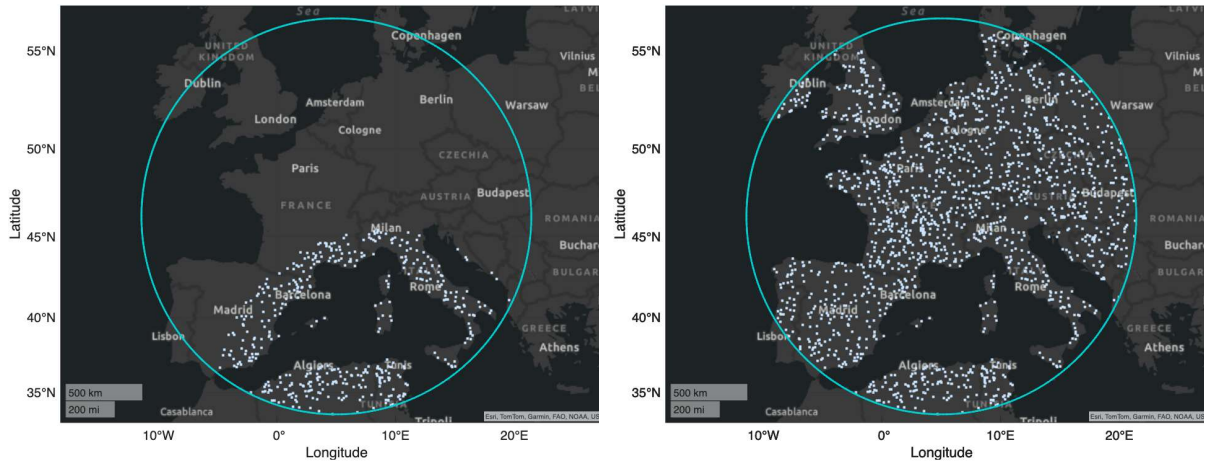
- $L_i^{(t)}$ is a term including all stochastic losses (shadowing, atmospheric losses, scintillation, and clutter loss in Line-of-Sight, LoS, or Non-LoS, NLoS, conditions, as per 3GPP TR 38.811, [46], and TR 38.821, [10]);
- $\mathrm{g}_{i,n}^{(tx,t)}$ and $\mathrm{g}_{i,n}^{(rx,t)}$ represent the transmitting and receiving complex antenna patterns between the n-th element and the i-th UE, respectively (the antenna model is extensively detailed in [47]);
- $\kappa B_i T_i$ represents the thermal noise power, with B_i being the user bandwidth, κ Boltzmann's constant, and T_i the equivalent noise temperature;
- $e^{-j\frac{2\pi}{\lambda}\mathrm{d}_i^{(t)}}$ represents the phase shift due to the slant range $\mathrm{d}_i^{(t)}$.

While CSI-based beamforming is based on direct estimates of the channel coefficient in eq. (1) performed at the UE, denoted as $\hat{\mathrm{h}}_{i,n}^{(t_R)}$ in the following, location-based solutions infer the channel coefficient from the estimated location:

$$\hat{\mathrm{h}}_{i,n}^{(t_R)} = \frac{\hat{\mathrm{g}}_{i,n}^{(tx,t_R)}\hat{\mathrm{g}}_{i,n}^{(rx,t_R)}}{\frac{4\pi \hat{\mathrm{d}}_i^{(t_R)}}{\lambda}\sqrt{\kappa B_i T_i}} e^{-j\frac{2\pi}{\lambda}\hat{\mathrm{d}}_i^{(t_R)}} \quad (2)$$

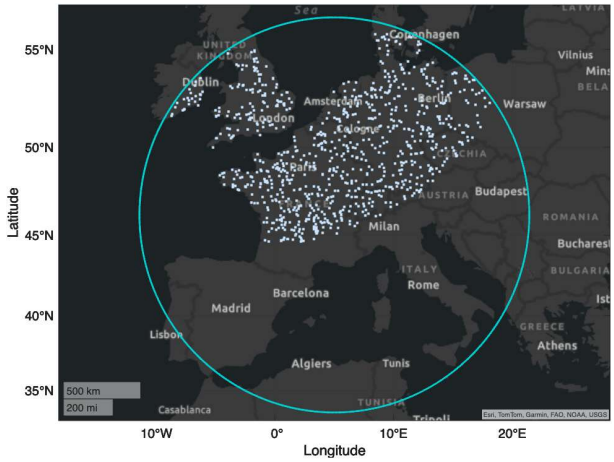
where the hat symbol over the parameters indicates their estimate based on the location, and again we refer to a generic reporting instant t_R.

Below, we report an extract of [2] to introduce the nomenclature and assumptions that are common to the entire work on user-centric beamforming and instrumental to discuss the solutions designed in Sections 4.2.3 and 4.3.3 of this document. In this framework, please note that: i) N_B denotes the maximum number of simultaneously active beams in a generic time slot; ii) $N_{sched}^{(slot)}$ denotes the number of time slots in a scheduling interval, with N_{sched} being the total number of scheduling intervals (as such, we have a total of $N_{sched}N_{sched}^{(slot)}$ time slots for a single NTN node pass over the service area).



(a) Visible users at 12:05:35

(b) Visible users at 12:08:54



(c) Visible users at 12:12:13

Figure 28: Example of visible users according to the NTN node position on its orbit with $(lat_S = 45°N, lon_S = 5°E)$, $\varepsilon_S = 30°$, $h_{sat} = 1000$ km (5G-STARDUST polar constellation, $i = 99°$).

Let us denote by N_{UE} the total number of users in the service area, obtained from one of the databases extensively discussed in Section 4.2 of [2]. In the generic reporting instant t_R, *i.e.*, when users shall estimate the ancillary information and report it, only a subset of the N_{UE} users is in the visibility of the NTN node, *i.e.*, with an elevation angle above a system target ε_t. If we denote as $\varepsilon_i^{(t_R)}$ the elevation angle of the i-th user in the generic reporting instant t_R, then the number of users that are in visibility and can report ancillary information, *i.e.*, the UEs that can potentially be scheduled by the RRM algorithm as they have a valid report, is given by:

$$N_{UE}^{(t_R)} = \left|\left\{i: \varepsilon_i^{(t_R)} \geq \varepsilon_t, i = 1, \dots, N_{UE}\right\}\right| \tag{3}$$

An example of visible users depending on the NTN node position is shown in Figure 28. Based on the latest received information and the specific RRM algorithm (designed in the next sections), in the generic scheduling instant $t_S > t_R$ (this condition ensures that scheduling is based on the latest report) the NTN node defines the $N_B \times N_{sched}^{(slot)}$ scheduling matrix $\mathbf{S}^{(m)}$, which

indicates the visible users to be served in each of the upcoming $N_{sched}^{(slot)}$ time slots, up to N_B per time slot. The (b,t)-th element $s_{b,t}$ of the scheduling matrix contains the index of the user to be served via the generic b-th user-centric beam in the generic t-th time slot.

Assuming that the RRM algorithm provided the scheduling matrix $\mathbf{S} = \left[\mathbf{S}^{(1)}, \dots, \mathbf{S}^{(m)}, \dots, \mathbf{S}^{(N_{sched})}\right]$, in the generic t-th transmission time slot the NTN node computes the beamforming coefficients by: i) selecting the N_B-dimensional t-th column of the scheduling matrix $\mathbf{S}$, $\mathbf{s}_{:,t}$; and ii) creating the estimated channel matrix at time slot t, $\widehat{\mathbf{H}}^{(t)}$, from the available ancillary information. The estimated channel matrix provides, on each row, the estimated CSI vector of the users to be served in the t-th time slot; thus, denoting by $\hat{\mathbf{h}}_{s_{b,t},:}^{(t)} = \left[\hat{\mathrm{h}}_{s_{b,t},1}^{(t)}, \dots, \hat{\mathrm{h}}_{s_{b,t},n}^{(t)}, \dots, \hat{\mathrm{h}}_{s_{b,t},N_R}^{(t)}\right]$ the complex channel coefficients between the user scheduled for the b-th beam in the t-th time slot, provided by $s_{b,t}$, and the $N_R = N_{R,sat}$ radiating elements on-board, we have that:

$$\widehat{\mathbf{H}}^{(t)} = \left[\left(\hat{\mathbf{h}}_{s_{1,t},:}^{(t)}\right)^T, \dots, \left(\hat{\mathbf{h}}_{s_{b,t},:}^{(t)}\right)^T, \left(\hat{\mathbf{h}}_{s_{N_B,t},:}^{(t)}\right)^T\right]^T \tag{4}$$

This matrix is the input to the beamforming algorithm, which we generically denote as a function $\mathcal{B}(\cdot)$ providing the $N_R \times N_B$ beamforming matrix $\mathbf{B}^{(t)} = \mathcal{B}\left(\widehat{\mathbf{H}}^{(t)}\right)$. The details on the considered beamforming algorithms are available in [47] and [2]. As in [2], in the following we consider the Minimum Mean Square Error (MMSE) and Conventional Beamforming (CBF) algorithms; as for the former, we consider legacy MMSE (CSI-based) and Location Based MMSE (LB-MMSE) solutions. Finally, in terms of the power distribution, based on the extensive performance assessment in [47], we consider only the Per Antenna Constraint (PAC), in which each radiating element is assumed to transmit the same amount of power $\frac{P_{av}}{N_{R,sat}}$, with P_{av} being the total available power on-board the NTN node. In fact, in the considered system PAC is the power distribution strategy providing the best performance when taking into account the achievable spectral efficiency, the exploitation of the on-board available power, and the limitation of the power at the input of the on-board amplifiers to avoid non-linearities. For the sake of completeness, Table 7 summarises the equations of the considered beamforming algorithms and the power distribution strategy.

Table 7: Summary of the considered beamforming algorithms and power distribution strategy.

Algorithm	Category	Equation	Estimated channel coefficient
MMSE	CSI-based	$\mathbf{B}^{(t)} = \left(\widehat{\mathbf{H}}^{(t)}\right)^{\boldsymbol{H}} \left(\widehat{\mathbf{H}}^{(t)}\left(\widehat{\mathbf{H}}^{(t)}\right)^{\boldsymbol{H}} + \alpha \mathbf{I}_{N_B}\right)^{-1}$	Estimation of all terms in eq. (1)
LB-MMSE	Location-based		Estimation of all non-stochastic terms in eq. (2)
CBF	Location-based	$\mathbf{b}_{:,k}^{(t)} = \frac{1}{\sqrt{N_R}} \sum_{n=1}^{N_R} e^{-jk_0 \mathbf{r}_n \cdot \mathbf{p}_i^{(t)}} = \frac{1}{\sqrt{N_R}} \Phi^*\left(u_i^{(t)}, v_i^{(t)}\right)$	Estimation of all non-stochastic terms in eq. (2)
PAC power distribution: $\widetilde{\mathbf{B}}^{(t)} = \sqrt{\frac{P_{av}}{N_R}} \mathrm{diag}\left(\frac{1}{\left\|\mathbf{b}_{1,:}^{(t)}\right\|}, \dots, \frac{1}{\left\|\mathbf{b}_{N_R,:}^{(t)}\right\|}\right) \mathbf{B}^{(t)}$			

NOTE 1: $\alpha = \frac{N_R}{P_{av}}$ is the MMSE beamforming regularisation factor

NOTE 2: in CBF, $\mathbf{r}_n$ represents the position of the n-th radiating element on the array and $\mathbf{p}_i^{(t)}$ the (u,v) coordinates (direction cosines) of the i-th user at the t-th time instant

The normalised beamforming matrix $\widetilde{\mathbf{B}}^{(t)}$ is the one used to transmit the beamformed signals to the scheduled users. Denoting by $\mathbf{x}^{(t)} = \left[\mathrm{x}_1^{(t)}, \dots, \mathrm{x}_{N_B}^{(t)}\right]^T$ the N_B-dimensional vector of unit-

variance symbols to be transmitted to the scheduled users in the generic t-th slot, the signal received by the generic k-th user can be written as:

$$\mathrm{y}_k^{(t)} = \underbrace{\mathbf{h}_{k,:}^{(t)} \tilde{\mathbf{b}}_{:,k}^{(t)} \mathrm{x}_k^{(t)}}_{\text{intended}} + \underbrace{\sum_{\substack{\ell=1 \\ \ell \neq k}}^{N_B} \mathbf{h}_{k,:}^{(t)} \tilde{\mathbf{b}}_{:,\ell}^{(t)} \mathrm{x}_\ell^{(t)}}_{\text{interfering}} + \mathrm{z}_k^{(t)} \tag{5}$$

where $z_k^{(t)}$ is a circularly symmetric Gaussian random variable (r.v.) with zero mean and unit variance, which is licit observing that the channel coefficients defined above are normalised to the noise power. It shall be noticed that, in eq. (5), the beamforming matrix is computed referring to the ancillary information obtained in the latest reporting period, while the channel encountered during the actual transmission is clearly, more or less, different. More specifically, the larger the aging interval $t - t_R$, the less aligned the beamforming matrix $\tilde{\mathbf{B}}^{(t)}$ will be to the actual channel conditions during the transmission, *i.e.*, to the $N_B \times N_R$ actual complex channel matrix $\mathbf{H}^{(t)} = \left[\left(\mathbf{h}_{\mathrm{s}_{1,t},:}^{(t)}\right)^T, \ldots, \left(\mathbf{h}_{\mathrm{s}_{b,t},:}^{(t)}\right)^T, \ldots, \left(\mathbf{h}_{\mathrm{s}_{N_B,t},:}^{(t)}\right)^T\right]^T$. Notably, the aging interval is always upper bounded by the reporting periodicity, *i.e.*, $\Delta t = t - t_R \leq T_{rep}$. Equation (5) can be easily extended to obtain the column vector of all received symbols during the t-th time slot:

$$\mathbf{y}^{(t)} = \mathbf{H}^{(t)} \tilde{\mathbf{B}}^{(t)} \mathbf{x}^{(t)} + \mathbf{z}^{(t)} \tag{6}$$

From eq. (6), we can compute the $N_B \times N_B$ power transfer matrix as, [47], [2]:

$$\mathbf{A}^{(t)} = \left|\mathbf{H}^{(t)} \tilde{\mathbf{B}}^{(t)}\right|^2 \tag{7}$$

which provides, in the generic (i, j)-th element, the power transmitted from the NTN node to the j-th user and received by the i-th user, normalised to the noise thermal power. Thus, all diagonal elements, $i = j$, represent the normalised intended power level, while all off-diagonal elements represent the normalised interfering power levels; in particular, all off-diagonal elements on a given row provide the interference received by the i-th user due to the other $N_B - 1$ transmissions scheduled in the same time slot. Bearing this in mind, it is straightforward to obtain the following Key Performance Indicators (KPIs) for the generic k-th user:

- The Signal-to-Noise Ratio (SNR) of the k-th user in the t-th time slot, given by the k-th diagonal element of the power transfer matrix:

$$\mathrm{SNR}_{k,t} = \mathrm{a}_{k,k}^{(t)} \tag{8}$$

- The Interference-to-Noise Ratio (INR) of the k-th user in the t-th time slot, given by the sum of the off-diagonal elements of the k-th row in the power transfer matrix:

$$\mathrm{INR}_{k,t} = \sum_{\substack{\ell=1 \\ \ell \neq k}}^{N_B} \mathrm{a}_{k,\ell}^{(t)} \tag{9}$$

- The **SINR**, obtained from the SNR and INR in equations (8) and (9), respectively:

$$\mathrm{SINR}_{k,t} = \frac{\mathrm{SNR}_{k,t}}{1 + \mathrm{INR}_{k,t}} = \frac{\left\|\mathbf{h}_{k,:}^{(t)} \tilde{\mathbf{b}}_{:,k}^{(t)}\right\|^2}{1 + \sum_{\ell=1, \ell \neq k}^{N_B} \left\|\mathbf{h}_{k,:}^{(t)} \tilde{\mathbf{b}}_{:,\ell}^{(t)}\right\|^2} \tag{10}$$

From the above SINR, the rate achieved by the k-th user can be evaluated either from the Shannon bound formula or from the adopted MCS. In this framework, 3GPP TR 38.803, [48], indicates that the spectral efficiency for system-level simulations can be computed as:

$$\eta_{k,t} = \begin{cases} 0, \mathrm{SINR}_{k,t} < \mathrm{SINR}_{min} \\ \rho \cdot \log_2(1 + \mathrm{SINR}_{k,t}), \mathrm{SINR}_{min} \leq \mathrm{SINR}_{k,t} \leq \mathrm{SINR}_{max} \\ \rho \cdot \log_2(1 + \mathrm{SINR}_{max}), \mathrm{SINR}_{k,t} > \mathrm{SINR}_{max} \end{cases} \quad (11)$$

where $\mathrm{SINR}_{min} = -10$ dB and $\mathrm{SINR}_{max} = 30$ dB are the minimum and maximum SINR values, respectively, and ρ is a factor representing the implementation losses. Since this factor is a multiplicative term outside the Shannon formula, we can assume $\rho = 1$ as a different value would simply act as scaling factor of all the results, without impacting the observations and the relationships among the various techniques and scenarios.

It shall be noticed that the spectral efficiency computed in eq. (11) represents the asymptotic spectral efficiency, *i.e.*, the spectral efficiency achieved assuming that the generic k-th user is granted a resource (user-centric beam) in all the available time slots and not just a subset of them (depending on the RRM strategy). To take the performance of the specific RRM algorithm into account, we need to consider the experienced spectral efficiency per user, which is the sum of achieved spectral efficiencies weighted by the percentage of channel use (*i.e.*, a weighted average of the asymptotic spectral efficiencies). To compute this quantity, let us denote by $\boldsymbol{\eta}_k$ the vector of spectral efficiencies achieved by the k-th user during the satellite pass, *i.e.*, $\boldsymbol{\eta}_k = \left[\eta_{k,1}, \dots \eta_{k,t}, \dots \eta_{k,N_{tot}^{(slot)}}\right]$ where $N_{tot}^{(slot)}$ is the total number of time slots (as previously stated, the satellite is passing over the service area for a time period equal to $T = N_{tot}^{(slot)} T_{slot}$). Let us also denote by $T_{vis,k} = N_k^{(slot)} T_{slot}$ the total visibility time for the generic k-th user; $T_{vis,k}$ represents the total amount of time in which the user can be served by this satellite (in other periods, it will be scheduled by another NTN node in the constellation). The experienced spectral efficiency of the k-th user is thus obtained as:

$$\bar{\eta}_k = \sum_{t=1}^{N_{tot}^{(slot)}} \frac{\eta_{k,t} T_{slot}}{T_{vis,k}} = \frac{1}{N_k^{(slot)}} \sum_{t=1}^{N_{tot}^{(slot)}} \eta_{k,t} \quad (12)$$

Finally, since we are operating with Full Frequency Reuse (FFR), all the users are allocated the entire bandwidth B in each time slot. Thus, we can compute the total achieved capacity of the generic k-th user as:

$$\mathrm{C}_k^{(A)} = B \cdot \bar{\eta}_k = \frac{B}{N_k^{(slot)}} \sum_{t=1}^{N_{tot}^{(slot)}} \eta_{k,t} = \sum_{t=1}^{N_{tot}^{(slot)}} \mathrm{C}_{k,t}^{(A)} \quad (13)$$

where in the last formulation we brought the bandwidth in the summation to obtain the achieved capacity as the sum of all the achieved capacities of the user in its allocated time slots. This metric provides the total achieved capacity at system-level during a full NTN node pass over the service area, for the generic k-th user. Differently from [41], in which random scheduling algorithms were considered because the traffic request from the users was uniform, here we consider non-uniform traffic requests. In particular, denoting as $\mathrm{C}_k^{(REQ)}$ the capacity requested by the generic k-th user, we introduce the following metrics:

- The achieved sum capacity, providing the total achieved capacity for all users (*i.e.*, the total capacity provided at system-level by the NTN node in a service period T):

$$\mathrm{C}^{(A)} = \sum_{k=1}^{N_{UE}} \mathrm{C}_k^{(A)} = B \sum_{k=1}^{N_{UE}} \bar{\eta}_k = \frac{B}{N_k^{(slot)}} \sum_{t=1}^{N_{tot}^{(slot)}} \sum_{k=1}^{N_{UE}} \eta_{k,t} \tag{14}$$

- The unmet capacity of the k-th user, measuring the amount of capacity that the user requested but was not satisfied by the NTN node in the service period T:

$$\mathrm{C}_k^{(U)} = \max\left(\mathrm{C}_k^{(R)} - \mathrm{C}_k^{(A)}, 0\right) \tag{15}$$

  This metric is null when $\mathrm{C}_k^{(R)} - \mathrm{C}_k^{(A)} < 0$, which corresponds to a scenario in which the k-th user was allocated at least the capacity that was requested. From this metric, which will be extensively exploited by the scheduling algorithms designed in the next section, we can derive the unmet sum capacity, which provides the total amount of capacity that remained unmet by the NTN node after the service period T:

$$\mathrm{C}^{(U)} = \sum_{k=1}^{N_{UE}} \mathrm{C}_k^{(U)} = \sum_{k=1}^{N_{UE}} \max\left(\mathrm{C}_k^{(R)} - \mathrm{C}_k^{(A)}, 0\right) \tag{16}$$

  It is convenient to define an additional metric to be exploited in designing the scheduling algorithms below, namely the unmet capacity of the k-th user at time t:

$$\mathrm{C}_{k,t}^{(U)} = \max\left(\mathrm{C}_k^{(R)} - \sum_{i=1}^{t-1} \mathrm{C}_{k,i}^{(A)}, 0\right) \tag{17}$$

- The excess sum capacity, which provides a measure of the amount of capacity that has been exceeded compared to the actual requests of the users:

$$\mathrm{C}^{(E)} = \max\left(\mathrm{C}^{(A)} - \mathrm{C}^{(R)}, 0\right) = \max\left(\sum_{k=1}^{N_{UE}} \left(\mathrm{C}_k^{(A)} - \mathrm{C}_k^{(R)}\right), 0\right) \tag{18}$$

  This metric might be non-null because the RRM algorithms, as extensively discussed in the next sections, will reintroduce users for which the capacity request was already met in the scheduling loop, in case resources are still available.

These metrics, and in specific cases the corresponding Cumulative Distribution Functions (CDFs), will be used as the main KPIs to assess the performance of CA-Priority Queue Scheduler (PQS) and the comparison with the benchmark PQS.

4.2 CLUSTERING-AIDED SCHEDULER FOR USER-CENTRIC DIGITAL BEAMFORMING

4.2.1 Rationale

In [2], two scheduling algorithms were designed and extensively assessed: i) the Priority Queue Scheduler (PQS), based on the definition of N_{PC} mutually exclusive Priority Classes (PCs), and corresponding queues; and ii) the Priority Rank Scheduler (PRS), with a slightly larger computational complexity compared to PQS, but without PCs as each user is ranked based on a single metric obtained from the currently unmet capacity and residual visibility time.

Both algorithms provide a good performance in terms of achieved and unmet capacity, in particular in low to medium traffic scenarios; moreover, the relatively simple approach followed in designing both solutions is also promising in terms of computational complexity, when comparing them to more advanced optimisation techniques calling for integer programming or graph theory. In this framework, as extensively discussed in Section 4.3.1.3 of [2]:

- Both PQS and PRS have a preparation phase before the iterative allocation of users in the time slots, leading to a fixed term in the big-O complexity:

$$\mathcal{O}_{fixed}^{(PQS)} = \mathcal{O}(N_{sche}\ N_{UE} N_{PC}) \tag{19}$$

$$\mathcal{O}_{fixed}^{(PRS)} = \mathcal{O}(N_{sched} N_{UE} \log_2 N_{UE}) \tag{20}$$

- Both PQS and PRS have an iterative phase, in which the schedulers verify that the users are not located too close or too similar from a channel perspective. These conditions are respectively verified based on: i) a geographical scheduling constraint, ρ_{GEO}, which represents the minimum distance among co-slot users as a multiple of the beam radius; and ii) a CSI scheduling constraint ρ_{CSI}, representing the maximum Coefficient of Correlation (CoC) among the CSI vectors of co-slot users. Since the two scheduling algorithms iteratively verify the satisfaction of the selected scheduling condition, the computational complexity of this phase was computed per iteration:

$$\mathcal{O}_{iter}^{(PQS)} = \mathcal{O}_{iter}^{(PRS)} = \mathcal{O}(N_{UE}) \tag{21}$$

The cost of the iterative phase of the algorithms in eq. (21) represents the big-o complexity of a single iteration. The number of iterations clearly depends on the probability of finding a user that meets the geographical or CSI scheduling constraint for each of the N_B available beams in a time slot. For instance, assuming that for each beam to be defined we need to iterate, on average, N_{iter} times the process, the total cost of the iterative phase will be $\mathcal{O}(N_{sched} N_{iter} N_{UE})$ for both algorithms; since we are multiplying the number of users and the number of scheduling periods in the computational complexity by N_{iter}, approaching the worst case of N_{UE} iterations would be significantly detrimental for the total computational complexity.

The costly iterative phase in both PQS and PRS is aimed at separating users in the 2D geographical or $2N_R$-dimensional CSI spaces: users too close in the selected space would lead to an ill-conditioned channel matrix that, once used for user-centric beamforming, ultimately leads to an SINR reduction. In this context, a very flexible and powerful ML technique can be a feasible solution in the near-real time RIC to reduce the computational burden of this scheduling step: clustering. In fact, clustering is an unsupervised ML technique in which a set of observations is automatically analysed and partitioned into N_C groups (clusters): objects (observations) in the same cluster exhibit a larger similarity with respect to those included in another cluster, with a similarity measured in the space defined by the considered features (the characteristics of the objects based on which we decide if they are similar or not).

The basic idea for the application of clustering to scheduling is the following: assuming that the feature space $\mathcal{F}$ is representing either the location or the CSI vector of each user, selecting one user from two different clusters is implicitly guaranteeing that the two users are sufficiently different (at least most likely, as we will see) in terms of their location or CSI vectors. This would allow to avoid costly iterative solutions in the scheduling algorithms, since users are *a priori* grouped (clustered) based on their similarity.

4.2.2 Clustering basics

Depending on the type of beamforming technique that we are considering, the feature space will have a different number of dimensions and meaning: i) in location-based techniques (LB-MMSE and CBF), the feature space $\mathcal{F}$ has two dimensions, *i.e.*, the u and v coordinates of the users; ii) in CSI-based solutions (MMSE), the feature space $\mathcal{F}$ has $2N_R$ dimensions representing the real and imaginary parts of each of the N_R channel coefficients between a generic UE and the on-board radiating elements. Thus, the generic k-th user is represented by: i) its current location in direction cosines for location-based techniques; or ii) its channel vector split into real and imaginary parts in CSI-based solutions. It shall be noticed that splitting the real and imaginary parts of the complex channel coefficient is required as the clustering algorithms operate in the real space when computing the distances among objects in the selected feature space. Based on these considerations, in the following, we denote as

$$\mathcal{F}_{GEO}^{(t_R)} = \{(u_k, v_k)\}_{k=1}^{N_{UE}^{(t_R)}} \tag{22}$$

the observation space for location-based techniques in the generic reporting instant t_R, containing the coordinates of all the $N_{UE}^{(t_R)}$ visible users in that slot, and by:

$$\mathcal{F}_{CSI}^{(t_R)} = \left\{\hat{\mathbf{h}}_{k,:}^{(t)}\right\}_{k=1}^{N_{UE}^{(t_R)}} \tag{23}$$

the observation space for CSI-based techniques in the generic reporting instant t_R containing the real and imaginary parts of the channel coefficients between all the $N_{UE}^{(t_R)}$ visible users in that slot and the on-board radiating elements. In this case, alternative solutions to have a real-valued observation space are clearly possible; for instance, instead of separating the real and imaginary parts of each channel coefficient, clustering can also be performed considering the amplitude and phases of each coefficient (*i.e.*, the polar representation): $\hat{\mathbf{h}}_{k,:}^{(t)} = \left[\left|\hat{\mathrm{h}}_{k,1}^{(t)}\right|, \dots, \left|\hat{\mathrm{h}}_{k,n}^{(t)}\right|, \dots, \left|\hat{\mathrm{h}}_{k,N_R}^{(t)}\right|, \angle\hat{\mathrm{h}}_{k,1}^{(t)}, \dots, \angle\hat{\mathrm{h}}_{k,n}^{(t)}, \dots, \angle\hat{\mathrm{h}}_{k,N_R}^{(t)}\right]$.

Let us denote by $\Pi(\cdot)$ the clustering (partitioning) function. Then, this function provides, for each visible user, the index of the cluster (partition) obtained based on the user position in the selected feature space:

$$\Pi\left(\mathcal{F}_x^{(t_R)}\right) = \{\mathrm{c}_k\}_{k=1}^{N_{UE}^{(t_R)}}, x = GEO, CSI \tag{24}$$

where c_k is the cluster (partition) index of the k-th visible user. When $\mathrm{c}_k = j$, it means that the k-th visible user belongs to the j-th partition π_j, $j = 1, \dots, N_C$. The partitions defined by the partitioning function Π shall satisfy the following two conditions:

- each user exclusively belongs to a single cluster, *i.e.*, $\pi_q \cap \pi_j = \emptyset$, $\forall q, j = 1, \dots, N_C, q \neq j$;
- the union of all partitions corresponds to the set of users, *i.e.*, $\bigcup_{j=1}^{N_C} \pi_j = \left\{1, \dots, N_{UE}^{(t_R)}\right\}$.

Figure 29 shows an example of clustering in the geographical space, *i.e.*, with observations belonging to the space $\mathcal{F}_{GEO}^{(t_R)}$, assuming $N_C = 5$ clusters and the polar orbit constellation of 5G-STARDUST. Figure 30 shows the result of clustering in the CSI space, *i.e.*, based on $\mathcal{F}_{CSI}^{(t_R)}$. The limited number of considered clusters is to allow an easier interpretation of the figures.

The ring-shaped partitioning outcome is due to the assumption of clear-sly conditions and the period phase shift introduced by the slant range in the channel coefficients.

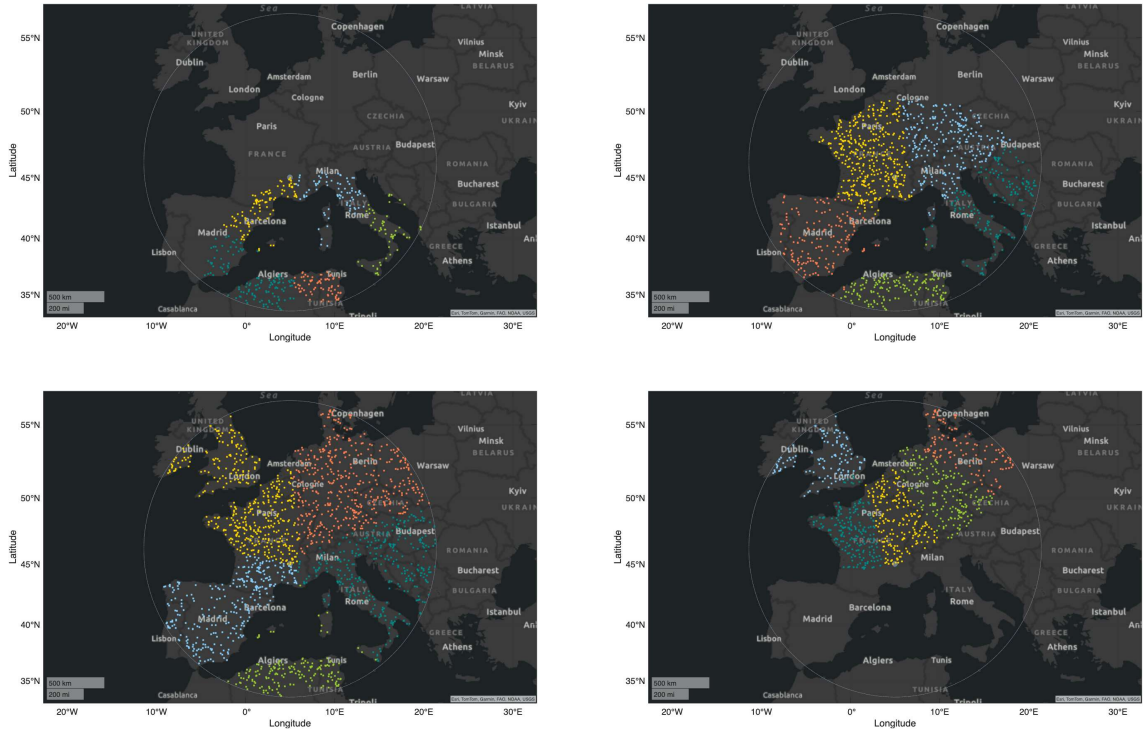


Figure 29: Example of clustering in the geographical observation space, $\mathcal{F}_{GEO}^{(t_R)}$, with $N_C = 5$ clusters.

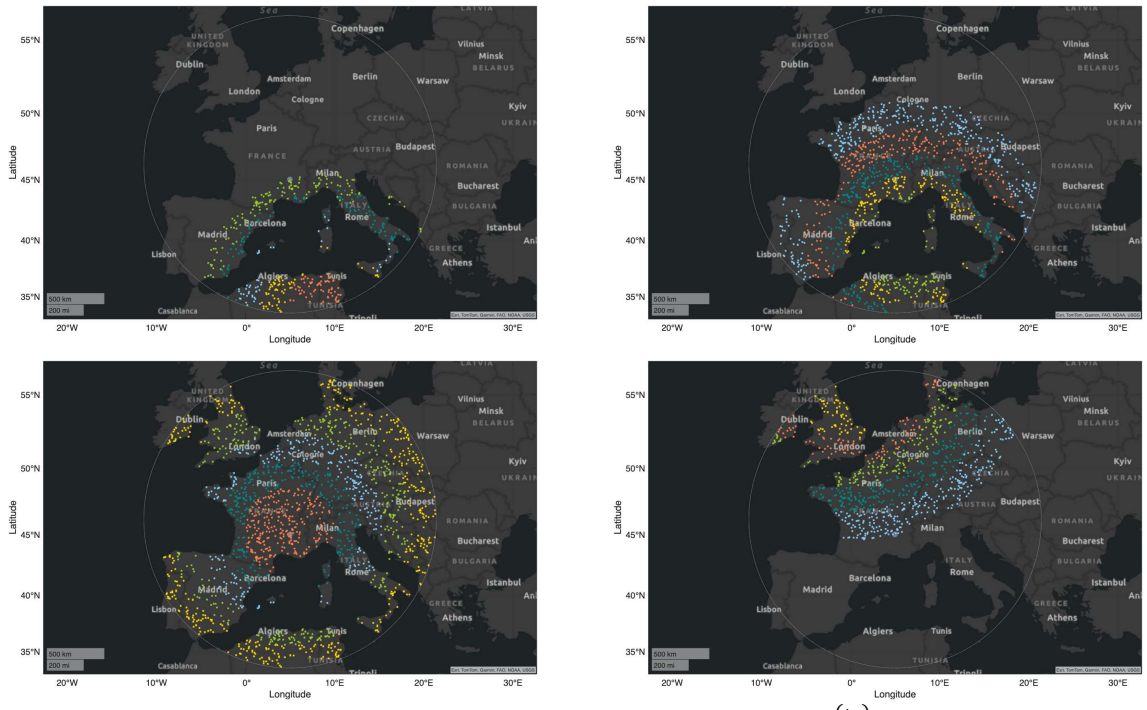


Figure 30: Example of clustering in the CSI observation space, $\mathcal{F}_{CSI}^{(t_R)}$, with $N_C = 5$ clusters.

4.2.3 Clustering-Aided PQS (CA-PQS)

The design approach for **Clustering Aided (CA) scheduling algorithms**, is shown in Figure 31. At the beginning of the scheduling interval, the NTN node clusters the visible users into N_C

clusters. Then, for each of the identified clusters, a dedicated, scheduler is in charge of managing the packet queues. For instance, relying on the PQS or PRS algorithms designed in D4.5, this means having a dedicated PQS or PRS scheduler for each cluster; while each cluster has its own scheduler, their complexity is reduced thanks to a two-fold advantage provided by the clustering step: i) clustering the visible users already provides an implicit separation in the geographical or CSI spaces and, thus, there is no need in the dedicated PQS or PRS per cluster to iteratively check the satisfaction of the geographical or CSI scheduling constraint (one of the most demanding tasks); and ii) there is no need to keep track of the users that have already been tested with the selected scheduling constraint.

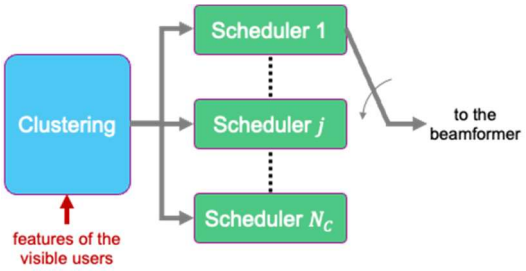
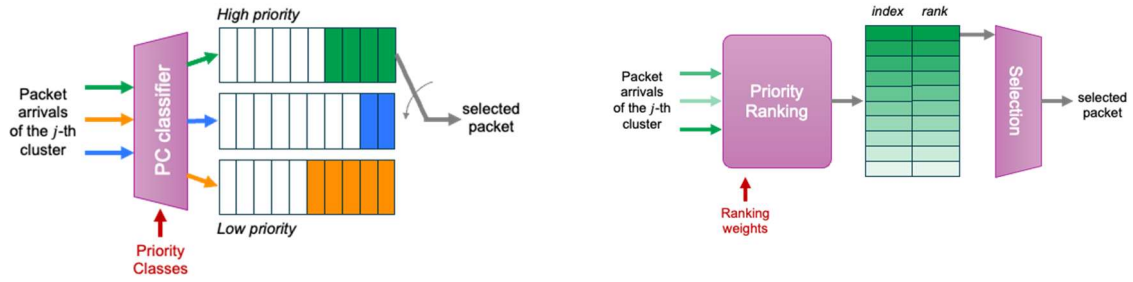


Figure 31: High-level diagram of Clustering Aided scheduling.

The high-level diagrams of the simplified PQS and PRS are shown in Figure 32, where we can indeed notice that, compared to the benchmark solutions in D4.5, there is neither need to consider the scheduling constraints nor to keep track of the barred users (users already tested that are not satisfying the selected constraint).



(a) simplified PQS

(b) simplified PRS

Figure 32: High-level diagrams of PQS (left) and PRS (right) for Cluster Aided scheduling.

Algorithm 1 shows the pseudo-code of a generic CA scheduler, in which the number of clusters is equal to or larger than the number of beams: $N_C \geq N_B$. The required steps are described hereafter[2]:

- As input to the scheduler, the NTN node shall be provided with: i) the visibility vector $\mathbf{v}_{vis}^{(t)}$, providing the indexes of the users that are currently visible; ii) the allocation vector $\mathbf{a}$ containing the number of already allocated time slots per user; and iii) the current amount of unmet capacity per user. Finally, any additional information that is needed by the specific scheduler implemented at cluster level shall also be provided. The

[2] Please note that a detailed description of several variables are provided in D4.5 and not duplicated here for the sake of conciseness. The reader is encouraged to read D4.5 for more information, if needed.

scheduler operates in two phases, namely an inter-cluster phase to select N_B clusters and an intra-cluster phase to schedule users from the selected clusters. The inter-cluster phase here considered makes use of the reported observation space $\mathcal{F}_x^{(t_R)}$ and an optimized selection weight w_s. For the intra-cluster scheduling phase, several user scheduling techniques can be implemented; PQS and PRS, for instance, make use of the following information, [49]:

- the set of mutually exclusive Priority Class conditions $\{\Omega_q\}_{q=1}^{N_{PC}}$ for PQS;
- the capacity and visibility weights ψ_C, ψ_V with $\psi_C + \psi_V = 1$ for PRS.

Algorithm 1: Clustering Aided scheduler

Input • visibility vector $\mathbf{v}_{vis}^{(t)}$, with $t = 1 + N_{sche}^{(slot)}(m-1), \dots, mN_{sched}^{(slot)}$ • $N_{UE} \times 1$ allocation vector $\mathbf{a}$ • current unmet capacity per user $\mathrm{C}_{k,N_m}^{(U)}$, $k = 1, \dots, N_{UE}$ • additional information for the cluster-level scheduler • additional information for the cluster selection procedure		
Output • $N_B \times N_{sched}^{(slot)}$ scheduling matrix $\mathbf{S}^{(m)}$, $m = 1, \dots, N_{sched}$		

1	Apply the selected clustering algorithm to obtain the cluster indexes: $\Pi\left(\mathcal{F}_x^{(t_0)}\right) = \{\mathrm{c}_k\}_{k=1}^{N_{UE}^{(t_0)}}, x = GEO, CSI$
	for $j = 1, \dots, N_C$:
	Remove from π_j all users with null unmet capacity or that are no longer visible: $\tilde{\pi}_j \leftarrow \pi_j \setminus \delta_j$
	if $\mathrm{Card}(\tilde{\pi}_j) = 0$:
	$\tilde{\pi}_j \leftarrow \pi_j$
	end if
	end for
		Clustering is performed at the beginning of the scheduling interval only: $t_0 = 1 + N_{sched}^{(slot)}(m-1)$
		
		If the cluster is empty, we re-initialise it

for $t = 1 + N_{sched}^{(slot)}(m-1), \dots, mN_{sched}^{(slot)}$:		
2	**if** $N_C > N_B$ Apply the inter-cluster scheduling algorithm on the clusters → obtain S_C **end if**	Inter-cluster scheduling phase
3	$j = 1$ **for** $k \in S_C$: apply the intra-cluster scheduling algorithm on cluster π_k → obtain $\mathrm{s}_{j,t}$ $j \leftarrow j + 1$ **end for**	Intra-cluster scheduling phase
4	update the allocation number vector $\mathbf{a}$: $\mathrm{a}_k = \mathrm{a}_k + 1, \forall k \in \mathbf{s}_{:,t}$	We increment the number of allocations per user based on the scheduled slot
end for		The scheduling matrix for the current scheduling period, $\mathbf{S}^{(m)}$, is complete

- At the output, the scheduling algorithm defines the users to be served from each of the available N_B beams for the next $N_{sched}^{(slot)}$ time slots.

- The selected clustering algorithm provides the partitions π_j, $j = 1, \dots, N_C$. In the following, we assume that the generic j-th partition π_j is the set of indexes of the users belonging to the j-th cluster. After the partitioning, we shall focus only on those users that need to be served. Thus, we exclude from the cluster all users for which the traffic request was already satisfied or that are now beyond visibility, denoted by δ_j. In fact, these users have already been served and do not need additional traffic if there is still unmet capacity in the cluster. If the cluster is completely empty after this reduction, then the cluster is re-initialised to the original partition so as to avoid wasting potential beams (resources).

  - It shall be noticed that this is something different from the PQS or PRS solutions designed in D4.5. For those algorithms, in fact, if no additional user could be found while guaranteeing that the scheduling constraint is met, less than N_B user-centric beams were generated. In this case, we choose a different design path: we want to exploit all the available beams in each single time slot that is available.

- If the selected clustering algorithm generates a number of clusters larger than the number of beams, $N_C \geq N_B$, an inter-cluster scheduling algorithm is executed at each time slot to form the set of scheduled clusters S_C. As shown in Algorithm 2, the selection is iteratively performed based on two metrics: 1) the total unmet capacity in each non-selected cluster, and 2) the aggregated distance between already selected clusters and each non-selected cluster. Based on the user traffic profile, one metric can more severely affect the achieved and unmet capacity than the other. Thus, the two metrics are normalized to unitary sum, leading to the normalized total unmet capacity vector $\tilde{\mathbf{c}}_{N_m}^{(U)}$ and the normalized aggregated distance vector $\tilde{\boldsymbol{d}}$, and weighted through the optimized parameter w_s. The set of non-scheduled clusters $\overline{S_C}$ can then be sampled once per iteration as $\omega\left(\overline{S_C}, 1, w_s \tilde{\mathbf{c}}_{N_m}^{(U)} + (1 - w_s)\,\tilde{\boldsymbol{d}}\right)$ which, extending the terminology introduced in D4.5, indicates the random selection with probability mass function $w_s \tilde{\mathbf{c}}_{N_m}^{(U)} + (1 - w_s)\,\tilde{\boldsymbol{d}}$ of a single element.

- The selected intra-cluster scheduling algorithm can then be implemented, to select the user to be served in the current time slot for the j-th cluster, $s_{j,t}$.

Algorithm 2: Inter-cluster scheduling algorithm.

Input

- current unmet capacity per visible cluster user $C_{k,N_m}^{(U)}$, $k \in \tilde{\pi}_j$
- cluster-level cosine/Haversine distance matrix $\mathbf{D}$

Output

- set of selected clusters: S_C

1 **If** first time slot ($t = 1 + N_{sched}^{(slot)}(m-1)$):

Compute ancillary information for the inter-cluster scheduling algorithm

for $j = 1, \dots, N_C$:

Compute the location/CSI of each cluster centre:

$$\tilde{\mathcal{F}}_{j,x}^{(t_R)} = \frac{1}{Card(\tilde{\pi}_j)} \sum_{k \in \tilde{\pi}_j} \mathcal{F}_{k,x}^{(t_R)}$$

Compute cluster-level unmet capacity:

$$\tilde{C}_{j,N_m}^{(U)} = \sum_{k \in \tilde{\pi}_j} C_{k,N_m}^{(U)}$$

end for

Compute the cosine/Haversine distance matrix between cluster centres:

$d_{i,j} = dist_x\left(\tilde{\mathcal{F}}_{i,x}^{(t_R)}, \tilde{\mathcal{F}}_{j,x}^{(t_R)}\right),\ \forall i,j \in 1,\dots,N_C,\ x = GEO, CSI$

else refer to past $\tilde{\mathbf{C}}_{N_m}^{(U)}$ and $\boldsymbol{d}$

2 Initialize set of selected clusters:

$$S_C = \left\{\arg\max_j \tilde{C}_{j,N_m}^{(U)}\right\}$$

for $j = 2,\dots,N_B$:

$\overline{S_C} \leftarrow \{1,\dots,N_C\} \backslash S_C$

Iteratively fill the set of selected clusters S_C based on the inter-cluster distance and the aggregated unmet capacity until it contains N_B clusters

Compute the normalized sum of the inter-cluster distance between each non-selected clusters and all selected clusters:

$$\tilde{d}_i = \frac{\sum_{s \in S_C} d_{i,s}}{\sum_{k \in \overline{S_C}} \sum_{s \in S_C} d_{k,s}}, \forall i \in \overline{S_C}$$

Normalize the cluster-level unmet capacity:

$$\tilde{C}_{i,N_m}^{(U)} \leftarrow \frac{\tilde{C}_{i,N_m}^{(U)}}{\sum_{k \in \overline{S_C}} \tilde{C}_{k,N_m}^{(U)}}, \forall i \in \overline{S_C}$$

$S_C \leftarrow S_C \cup \omega\left(\overline{S_C}, 1, w_s \tilde{\mathbf{C}}_{N_m}^{(U)} + (1 - w_s)\,\tilde{\boldsymbol{d}}\right)$

Sample $\overline{S_C}$ once based on the weighted sum of $\tilde{\mathbf{C}}_{N_m}^{(U)}$ and $\tilde{\boldsymbol{d}}$

end for

Algorithm 3: Intra-cluster scheduling algorithm: cluster-level PQS.

Input

- current unmet capacity per visible cluster user $C_{k,N_m}^{(U)},\ k \in \tilde{\pi}_j$
- set of mutually exclusive Priority Class conditions $\{\Omega_q\}_{q=1}^{N_{PC}}$

Output

- user for the current time slot from the j-th cluster: $s_{j,t}$

1 **If** first time slot $(t = 1 + N_{sched}^{(slot)}(m-1))$:

Classification into Priority Classes only for the first time slot in the scheduling period

for $q = 1,\dots,N_{PC}$:

initialize $\tilde{\pi}_{j,q} = \emptyset$

for $k \in \tilde{\pi}_j$:

Iterative addition to the PC for each user in the cluster

$\tilde{\pi}_{j,q} \leftarrow k$ if $k \in \Omega_q$

end for

end for

else refer to past $\tilde{\pi}_{j,q}$

2 **for** $q = 1,\dots,N_{PC}$:

If $\mathrm{Card}\left(\tilde{\pi}_{j,q}\right) > 0$:

If the PC is not empty, we schedule the user, otherwise we move to the next one

select $s_{j,t} = \omega(\tilde{\pi}_{j,q}, 1)$
break
end for

In the following, we focus on the PQS algorithm for intra-cluster scheduling. However, it shall be noticed that the CA scheduler discussed above can be flexibly designed with any cluster-level algorithm, *i.e.*, also the PRS proposed in D4.5 or other solutions. Algorithm 2 provides the pseudo-code of the PQS modified to operate at cluster level (*i.e.*, to be aligned with the high-level diagram shown in Figure 32(a) above):

- The first step clearly involves the identification of the priority classes. In particular, we need to classify the users in the generic j-th cluster, provided in set $\tilde{\pi}_j$, in N_{PC} PCs, $\{\Omega_q\}_{q=1}^{N_{PC}}$. Please note that this step shall only be performed if this is the first time slot in the scheduling period. In fact, the unmet capacity in PQS is updated after each scheduling period and there would be no specific reason to have more frequent computations of the PCs. If this is not the first time slot in the scheduling period, then the scheduler can simply rely on the previously defined classes.

- Then, in order of priority, if the PC is empty, one random user is selected from the set, otherwise the search for a user moves to the next class. In this respect, please note that:

  - The random selection is denoted as $\omega(\tilde{\pi}_{j,q}, 1)$, which, as introduced in D4.5, indicates the random selection with a uniform distribution of a single element from set $\tilde{\pi}_{j,q}$.

  - Since the re-initialisation of empty clusters is performed before the intra-cluster scheduler (*i.e.*, in Algorithm 3), it is by design not possible at this stage that the scheduler moves through all priority classes without finding at least one user. As soon as a user is found, the cluster-level algorithm can provide the index of the user for this beam, denoted as $s_{j,t}$.

- In the cluster-level PRS, the scheduler shall compute the priority rank table for the cluster and then select the first user in the table, which contains the one with highest priority by definition (please see D4.5 for more details on the ranking mechanism).

The design of specific clustering algorithms is out of the scope of this document. As such, we evaluate the performance of the CA-PQS algorithm exploiting well-known clustering algorithms: **kmeans++**, **kmedoids++**, **Hierarchical Agglomerative Clustering (HAC)**, **Spectral Clustering**, and **Density-Based Spatial Clustering of Applications with Noise (DBSCAN)**. While a detailed description of these algorithms is out of the scope of this document, we can briefly summarise their main characteristics:

- kmeans is a very popular clustering algorithm that partitions the input dataset into a pre-defined number of clusters ($N_C = N_B$ in our system). It is based on an iterative process in which: i) initially, N_C random initial centroids (cluster centers) are chosen: ii) each observation in the input dataset is assigned to the cluster of the closest centroid; iii) the centroids are re-computed as the barycentre of the cluster members; and iv) steps 2-3 are iterated as long as there are no cluster swaps. The kmeans++ algorithm is a modified version of kmeans in which the random initialisation step is adjusted so as to guarantee that the initial centroids are more well distributed in the feature space.

- The main advantages of this algorithm are: i) its simplicity, since it is very easy to understand and implement; ii) the efficiency, as the computational complexity is $\mathcal{O}(nkdi)$ for clustering n vectors with d dimensions each in k clusters, requiring i iterations; iii) the scalability, as it can handle large datasets very effectively; and iv) its versatility, as it can be used for a wide range of applications.

- As for the challenges, or disadvantages, we shall mention: i) the need to provide the number of clusters as input to the algorithm (not an issue in our system, anyway); ii) the sensitivity to the initialisation of the centroids, which also with the kmeans++ might be problematic; iii) it assumes hyper-spherical clusters, so if data is distributed not in a uniform way in the feature space the clusters might be not very representative; iv) it is sensitive to outliers, which might negatively bias the update of the centroids; and v) it risks to converge to local, not global, minima.

- kmedoids is a partitioning clustering algorithm closely related to k-means that also requires the number of clusters N_C as input, but represents each cluster by a medoid, *i.e.*, an actual observation from the dataset, rather than by its barycentre. The Partitioning Around Medoids algorithm with the same initialization as kmeans++ is implemented: i) N_C initial medoids are randomly chosen; ii) each observation is assigned to the cluster of the closest medoid; iii) for each cluster, select as new medoid the observation within the cluster that minimises the total within-cluster distance; and iv) steps 2–3 are iterated until no medoid swaps are made. The kmedoids++ variant employs the same initialization step as kmeans++.

  - The main advantages of this algorithm are: i) robustness to outliers, since medoids are actual data points and are less influenced by extreme values than centroids; ii) interpretability, because cluster centres are real observations that can be inspected directly; and iii) often better performance than k-means on datasets with categorical variables or when the mean is not directly interpretable.

  - As for the challenges, or disadvantages, we shall mention: i) the need to provide the number of clusters N_C a priori; ii) higher computational cost than kmeans for classical implementations due to the evaluation of candidate non-medoid points, leading to a computational complexity $\mathcal{O}(n^2kdi)$; iii) sensitivity to the initial medoid selection and the risk of converging to a local, not global, optimum; and iv) although more robust to outliers than k-means, k-medoids can still struggle with complex cluster shapes and very high-dimensional data (as in the case of CSI).

- HAC is a partitioning technique that builds a hierarchy of clusters by iteratively merging the closest pairs of clusters based on a distance metric. It starts with a scenario in which each observation in the input dataset constitutes an individual cluster; then: i) the distance among all clusters is computed; ii) the two closest clusters are merged together; and iii) steps 1-2 are iterated until a predefined number of clusters is obtained, or if a single cluster remains (*i.e.*, all points have been merged together). This approach results in a dendrogram, a tree-like structure that visually represents the relationships between clusters and allows to either visually or analytically "cut" the tree based on the desired metric, which can be the number of clusters or a maximum dissimilarity when merging two clusters, to name two of the most popular. While we here implement only a cut based on a predefined number of clusters, we test the algorithm both in the case $N_C = N_B$ (denoted "HAC") and $N_C = 3N_B$ (denoted "HAC-3").

  - The main advantages of HAC are: i) the computation of a dendrogram, providing the visual representation of the input data structure, which makes it easy to understand the relationship between clusters; ii) the fact that it does not necessarily need the

number of clusters a priori; iii) it is capable of capturing nested clusters, *i.e.*, clusters within clusters; and iv) it has a very intuitive interpretation, thanks to the dendrogram.

  - The above advantages come at the cost of: i) a computationally demanding procedure, with a complexity that, in non-efficient implementations, can be as large as $\mathcal{O}(n^3)$; ii) the sensitivity to noise and outliers, which risk to distort the agglomerative operations; iii) once the clusters are merged, they cannot be separated, potentially leading to sub-optimal decisions; and iv) since it is computationally demanding, it does not scale well with the number of observations in the dataset.

- Spectral clustering is a graph-based approach that finds clusters by exploiting the eigenstructure of a similarity graph constructed from the data. The workflow is the following: i) construct a similarity matrix that encodes pairwise similarities between observations (*e.g.*, cosine similarity for CSI); ii) generate the an unweighted graph based on the similarity matrix and a minimum similarity threshold ε_s and determine the number of connected components N_C; iii) compute the unnormalized Laplacian matrix; iii) compute the first N_C eigenvectors of the Laplacian corresponding to the smallest eigenvalues to embed the original observations into an N_C-dimensional spectral subspace; and iv) apply kmeans to the rows of the spectral embedding to obtain final cluster assignments with N_C clusters.

  - The main advantages of this algorithm are: i) the ability to detect non-convex and complex shaped clusters that are not well captured by centroid-based methods; ii) flexibility through the affinity construction, as the notion of similarity can be adapted to domain knowledge (*e.g.*, using cosine similarity on CSI); iii) strong theoretical connections to graph partitioning and manifold learning, which often lead to high-quality partitions in practice; and iv) good empirical performance on small to medium sized datasets where the similarity graph and eigendecomposition are tractable.

  - As for the challenges, or disadvantages, we shall mention: i) the need to choose design parameters, including the similarity threshold ε_s, which strongly affect results; ii) the requirement to compute an eigendecomposition of an $n \times n$ matrix, which is computationally and memory intensive, leading to an overall computational complexity $\mathcal{O}(n^2 d + kn^2 + nk^2 i)$; and iii) sensitivity to noise and to poorly chosen similarity parameters, which can produce spurious graph connectivity and degrade the spectral embedding.

- DBSCAN is a density-based clustering algorithm that discovers clusters as high-density regions separated by areas of lower density, and does not require the number of clusters as input. The algorithm operates as follows: i) choose two parameters, ε_r (neighbourhood radius) and $N_{P,min}$ (minimum points to form a dense region); ii) classify each point as a core point (if at least $N_{P,min}$ points lie within its ε_r-neighbourhood), a border point (reachable from a core point but with fewer than $N_{P,min}$ neighbours), or noise; iii) form clusters by starting from unvisited core points and recursively adding all points density-reachable from them (core points connect via overlapping ε_r-neighbourhoods, border points are attached but do not expand clusters); and iv) continue until all points are labeled either as belonging to a cluster or as noise. DBSCAN thus naturally identifies arbitrarily shaped clusters and flags outliers; however, as every point corresponds to a user, we set $N_{P,min} = 1$ to ensure that all users are considered for scheduling. Thus, the algorithm initializes all points as core points and successively aggregates density-reachable pairs.

- The main advantages of this algorithm are: i) the ability to find clusters of arbitrary shape, *i.e.*, not restricted to spherical clusters; ii) it does not require specifying the number of clusters N_C in advance; iii) automatic identification of noise/outliers as points not belonging to any dense region; and iv) good practical performance on spatial and low-dimensional datasets, with efficient implementations achieving roughly $\mathcal{O}(n\log(n))$ using spatial indexing structures ($\mathcal{O}(n^2 d)$ otherwise).

- As for the challenges, or disadvantages, we shall mention: i) the sensitivity to the hyperparameter ε_r, as poor choices can merge distinct clusters or fragment a single cluster into many parts; ii) difficulty handling data with varying density levels, since one global ε_r typically cannot accommodate multiple density scales; and iii) degraded behaviour in high dimensional spaces where distance concentration makes meaningful density estimation problematic (not an issue with GEO-based reports).

Notably, other existing scheduling algorithms can be exploited with the proposed CA solution, or even new ones can be designed to better tailor the partitioning process to the NTN ecosystem. This study includes this selection since these are among the most used and known clustering algorithms in the literature, but other solutions can be easily embedded. Furthermore, please note that spectral clustering and DBSCAN have not been considered for al type of reports:

- As spectral clustering is based on the extraction of low dimensional embeddings from high dimensional observations, we apply spectral clustering only in the case of CSI reports.

- As the DBSCAN algorithm has been designed for spatial clustering and is effective only in low dimensional spaces, we only apply it to location-based reports and not to CSI reports.

4.2.4 Computational complexity

Before assessing the performance of CA-PQS, let us compute the big-o computational complexity of the designed scheduler. With respect to HAC and kmeans++, in line with the above discussion:

- The big-o complexity of kmeans++ is $\mathcal{O}(nkdi)$ for clustering n vectors with d dimensions each in k clusters, requiring i iterations. In our case, we consider a default maximum number of iterations $i = 100$. Thus, in our system, we obtain:

$$\mathcal{O}_{kmeans} = \mathcal{O}(nkdi) = \mathcal{O}(iN_{dim}N_{UE}N_C) \tag{25}$$

  where $N_{dim} = 2$ for location-based algorithms and $N_{dim} \sim N_R$ for CSI-based algorithms (512 considering the 5G-STARDUST antenna).

- The big-o complexity of kmedoids++ is $\mathcal{O}(n^2 kdi)$ for clustering n vectors with d dimensions each in k clusters, requiring i iterations. As for kmeans, we consider a default maximum number of iterations $i = 100$. Thus, in our system, we obtain:

$$\mathcal{O}_{kmedoids} = \mathcal{O}(n^2 kdi) = \mathcal{O}(iN_{dim}N_{UE}^2 N_C) \tag{26}$$

- As for HAC, the computational complexity of the baseline implementations is pretty large, $\mathcal{O}(n^3)$. However, in particular during the last years, several efficient algorithms have been designed to implement it. In particular, methods exploiting quadtress have been demonstrated to achieve a complexity $\mathcal{O}(n^2)$. Thus, we can assume:

$$\mathcal{O}_{HAC} = \mathcal{O}(N_{UE}^2) \tag{27}$$

- The big-o complexity of spectral clustering is $(n^2 d + kn^2 + nk^2 i)$ for clustering n vectors with d dimensions each in k clusters, requiring i kmeans iterations. As before, we consider a default maximum number of iterations $i = 100$; furthermore, the similarity matrix is directly obtained based on the CSI vectors of size N_R. Thus, in our system, we obtain:

$$\mathcal{O}_{spectral} = \mathcal{O}(n^2 d + kn^2 + nk^2 i) = \mathcal{O}(N_R N_{UE}^2 + N_C N_{UE}^2 + N_{UE} N_C^2 i) \tag{28}$$

- Finally, as for DBSCAN, the computational complexity is $\mathcal{O}(n \log(n))$ for clustering n low dimensional vectors. Thus, in our system, we obtain:

$$\mathcal{O}_{DBSCAN} = \mathcal{O}(n \log(n)) = \mathcal{O}(N_{UE} \log(N_{UE})) \tag{29}$$

Algorithm 4 reports the computation steps in the CA scheduler in Algorithm 1 with the related big-o complexities. We can notice that:

- We first have to apply the clustering algorithm, with complexity in equations (25) through (29) depending on the chosen algorithm.
- The identification of the users to be deleted (met traffic requirements or out of visibility) is discussed in D4.5 and it has a complexity of $\mathcal{O}(N_{UE})$ if it has to be performed on all users. Also accordingly updating the set of users costs $\mathcal{O}(N_{UE})$. This operation is actually performed on each cluster; assuming, on average, N_{UE}/N_C users per cluster, we repeat these operations N_C times with a complexity $\mathcal{O}(N_{UE}/N_C)$. Ultimately, the big-o complexity of this step is $\mathcal{O}(N_{UE})$.
- In the general case of $N_C \geq N_B$ (*e.g.*, when spectral clustering or DBSCAN are used)
- The next step is to iteratively select one user per cluster in the intra-cluster scheduling step, with a generic cost $N_B \mathcal{O}_{sched}$ that we will compute below for CA-PQS.
- The update of the allocation vector, again from D4.5, costs $\mathcal{O}(N_B)$.

Algorithm 4: Computational complexity of the steps in CA schedulers.

	Apply the selected clustering algorithm to obtain the cluster indexes: $\Pi\left(\mathcal{F}_x^{(t_R)}\right) = \{c_k\}_{k=1}^{N_{UE}^{(t_0)}}, x = GEO, CSI$	$\mathcal{O}(N_{dim} N_{UE} N_C i)$ [kmeans] $\mathcal{O}(N_{dim} N_{UE}^2 N_C i)$ [kmedoids] $\mathcal{O}(N_{UE}^2)$ [HAC] $\mathcal{O}(N_R N_{UE}^2 + N_C N_{UE}^2 + N_{UE} N_C^2 i)$ [spectral] $\mathcal{O}(N_{UE} \log(N_{UE}))$ [DBSCAN]
	for $j = 1, \ldots, N_C$: Remove from π_j all users with null unmet capacity or that are no longer visible: $\tilde{\pi}_j \leftarrow \pi_j \setminus \delta_j$ **if** $\mathrm{Card}(\tilde{\pi}_j) = 0$: $\tilde{\pi}_j \leftarrow \pi_j$ **end for**	$\mathcal{O}(N_{UE})$
		$\mathcal{O}(N_{dim} N_{UE} N_C i)$ [kmeans]

	$\mathcal{O}(N_{dim}N_{UE}^2N_Ci)$ [kmedoids] $\mathcal{O}(N_{UE}^2)$ [HAC] $\mathcal{O}(N_RN_{UE}^2 + N_CN_{UE}^2 + N_{UE}N_C^2i)$ [spectral] $\mathcal{O}(N_{UE}\log(N_{UE}))$ [DBSCAN]
for $t = 1 + N_{sched}^{(slot)}(m-1), \dots, mN_{sched}^{(slot)}$:	
if $N_C > N_B$ Apply the inter-cluster scheduling algorithm on the clusters → obtain S_C **end if**	$\mathcal{O}_{inter}$
$j = 1$ **for** $k \in S_C$: apply the intra-cluster scheduling algorithm on cluster π_k → obtain $s_{j,t}$ $j \leftarrow j + 1$ **end for**	$N_B\mathcal{O}_{intra}$
update the allocation number vector $\mathbf{a}$: $a_k = a_k + 1, \forall k \in \mathbf{s}_{:,t}$	$\mathcal{O}(N_B)$
end for	
	$N_{sched}^{(slot)}\left(\mathcal{O}_{inter} + N_B\mathcal{O}_{intra} + \mathcal{O}(N_B)\right)$

The total computational complexity is thus:

$$\mathcal{O}(N_{dim}N_{UE}N_Bi) + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} + N_{sched}^{(slot)}\mathcal{O}(N_B) \sim \mathcal{O}(N_{dim}N_{UE}N_Bi) + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} \quad (30)$$

for kmeans under the assumption that $N_C = N_B$;

$$\mathcal{O}(N_{dim}N_{UE}^2N_Bi) + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} + N_{sched}^{(slot)}\mathcal{O}(N_B) \sim \mathcal{O}(N_{dim}N_{UE}^2N_Bi) + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} \quad (31)$$

for kmedoids under the assumption that $N_C = N_B$;

$$\mathcal{O}(N_{UE}^2) + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} + N_{sched}^{(slot)}\mathcal{O}(N_B) \sim \mathcal{O}(N_{UE}^2) + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} \quad (32)$$

for HAC under the assumption that $N_C = N_B$;

$$\mathcal{O}(N_{UE}^2) + N_{sched}^{(slot)}\mathcal{O}_{inter} + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} + N_{sched}^{(slot)}\mathcal{O}(N_B) \sim \\ \sim \mathcal{O}(N_{UE}^2) + N_{sched}^{(slot)}\mathcal{O}_{inter} + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} \quad (33)$$

for HAC under the more general assumption that $N_C \geq N_B$ (including HAC-3);

$$\mathcal{O}(N_RN_{UE}^2 + N_CN_{UE}^2 + N_{UE}N_C^2i) + N_{sched}^{(slot)}\mathcal{O}_{inter} + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} + N_{sched}^{(slot)}\mathcal{O}(N_B) \sim \\ \sim \mathcal{O}(N_RN_{UE}^2 + N_CN_{UE}^2 + N_{UE}N_C^2i) + N_{sched}^{(slot)}\mathcal{O}_{inter} + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} \quad (34)$$

for spectral clustering under the more general assumption that $N_C \geq N_B$; and

$$\mathcal{O}(N_{UE}\log(N_{UE})) + N_{sched}^{(slot)}\mathcal{O}_{inter} + N_{sched}^{(slot)}N_B\mathcal{O}_{intra} + N_{sched}^{(slot)}\mathcal{O}(N_B) \sim \\ \sim \mathcal{O}(N_{UE}\log(N_{UE})) + N_{sche}^{(slot)}\mathcal{O}_{inter} + N_{sche}^{(slot)}N_B\mathcal{O}_{intra} \quad (35)$$

for DBSCAN, where in all equations we observed that $\mathcal{O}(N_B)$ is a term that is dominated by the others. We thus need the big-o complexity of the inter-cluster algorithm and the intra-cluster simplified PQS algorithm to compute the final complexity. Referring to Algorithm 5, we have:

- Only at the first time slot, we need to compute the location/CSI of each cluster centre, the total unmet capacity of each cluster, and obtain the cosine/Haversine distance matrix between cluster centres. These operations have an overall cost $\mathcal{O}(N_{UE} + N_C^2)$.
- At each time slot, we need to iteratively compute the sum distance between the already selected cluster centres and each of the non-selected centres, with total cost $\mathcal{O}\left(\sum_{j=1}^{N_B} j(N_C - j)(N_C - j + 1)\right) \sim \mathcal{O}(N_B^2 N_C^2)$. Similarly, we need to extract and normalize to unit sum the cluster-level unmet capacity for non-scheduled centres, with total cost $\mathcal{O}\left(\sum_{j=1}^{N_B}(N_C - j)^2\right) \sim \mathcal{O}(N_B N_C^2)$.

The cost of the inter-cluster phase is thus:

$$\mathcal{O}_{inter} = \mathcal{O}(N_{UE} + N_C^2) + \mathcal{O}(N_B^2 N_C^2) + \mathcal{O}(N_B N_C^2) \sim \mathcal{O}(N_B^2 N_C^2) \tag{36}$$

Algorithm 5: Computational complexity of the inter-cluster scheduling phase.

1	**If** first time slot $(t = 1 + N_{sched}^{(slot)}(m-1))$:	
	for $j = 1, \dots, N_C$:	$\mathcal{O}(N_{UE})$
	Compute the location/CSI of each cluster centre: $\tilde{\mathcal{F}}_{j,x}^{(t_R)} = \frac{1}{Card(\tilde{\pi}_j)} \sum_{k \in \tilde{\pi}_j} \mathcal{F}_{k,x}^{(t_R)}$	$\mathcal{O}(N_{UE})$
	Compute the cluster-level unmet capacity: $\tilde{C}_{j,N_m}^{(U)} = \sum_{k \in \tilde{\pi}_j} C_{k,N_m}^{(U)}$	
	end for	
	Compute the cosine/Haversine distance matrix between cluster centres: $d_{i,j} = dist_x\left(\tilde{\mathcal{F}}_{i,x}^{(t_R)}, \tilde{\mathcal{F}}_{j,x}^{(t_R)}\right),\ \forall i,j \in 1, \dots, N_C,\ x = GEO, CSI$	$\mathcal{O}(N_C^2)$
	else refer to past $\tilde{\mathbf{C}}_{N_m}^{(U)}$ and $\boldsymbol{d}$	
		$\mathcal{O}(N_{UE} + N_C^2)$
2	Initialize set of selected clusters: $S_C = \left\{\arg\max_j \tilde{C}_{j,N_m}^{(U)}\right\}$	$\mathcal{O}(N_C)$
	for $j = 2, \dots, N_B$: $\overline{S_C} \leftarrow \{1, \dots, N_C\} \backslash S_C$	

	Compute the normalized sum of the inter-cluster distance between each non-selected clusters and all selected clusters: $\tilde{d}_i = \frac{\sum_{s\in S_C} d_{i,s}}{\sum_{k\in\overline{S_C}}\sum_{s\in S_C} d_{k,s}}, \forall i \in \overline{S_C}$	At iteration j: $\mathcal{O}(j(N_C - j)(N_C - j + 1))$
	Normalize the cluster-level unmet capacity: $\tilde{C}^{(U)}_{i,N_m} \leftarrow \frac{\tilde{C}^{(U)}_{i,N_m}}{\sum_{k\in\overline{S_C}} \tilde{C}^{(U)}_{k,N_m}}, \forall i \in \overline{S_C}$	At iteration j: $\mathcal{O}((N_C - j)^2)$
	$S_C \leftarrow S_C \cup \omega\left(\overline{S_C}, 1, w_s \tilde{\mathbf{C}}^{(U)}_{N_m} + (1 - w_s)\,\tilde{\boldsymbol{d}}\right)$	
	end for	
		$\mathcal{O}(N_B^2 N_C^2)$

As for the intra-cluster phase, referring to Algorithm 6, we have:

- Only at the first time slot, we need to assign the users of each selected cluster to the PCs. This operation, from D4.5, costs $\mathcal{O}(N_{UE}N_{PC}/N_B)$.
- The check on the cardinality of the cluster and the selection of a pseudo-random value from a set cost both $\mathcal{O}(N_{UE}/N_B)$.

The cost of the PQS algorithm is thus:

$$\mathcal{O}_{intra} = \mathcal{O}\left(\frac{N_{UE}N_{PC}}{N_B}\right) + \mathcal{O}\left(\frac{N_{UE}}{N_B}\right) + \mathcal{O}\left(\frac{N_{UE}}{N_B}\right) \sim \mathcal{O}\left(\frac{N_{UE}N_{PC}}{N_B}\right) \tag{37}$$

Algorithm 6: Computational complexity of the cluster-level PQS algorithm.

1	**If** first time slot ($t = 1 + N^{(slot)}_{sched}(m-1)$): **for** $q = 1, \dots, N_{PC}$: initialize $\tilde{\pi}_{j,q} = \emptyset$ **for** $k \in \tilde{\pi}_j$: $\tilde{\pi}_{j,q} \leftarrow k$ if $k \in \Omega_q$ **end for** **end for** **else** refer to past $\tilde{\pi}_{j,q}$	$\mathcal{O}(N_{UE}N_{PC}/N_B)$
		$\mathcal{O}(N_{UE}N_{PC}/N_B)$
2	**for** $q = 1, \dots, N_{PC}$: **If** $\mathrm{Card}(\tilde{\pi}_{j,q}) > 0$: select $s_{j,t} = \omega(\tilde{\pi}_{j,q}, 1)$ **break** **end for**	$\mathcal{O}(N_{UE}/N_B)$ $\mathcal{O}(N_{UE}/N_B)$
		$\mathcal{O}(N_{UE}/N_B)$

Combining equation (30) with (36) and (37), and taking into account all the scheduling periods and time slots per scheduling period, we finally obtain for kmeans:

$$\begin{aligned}\mathcal{O}_{kmeans} &= N_{sched}\left(\mathcal{O}(N_{dim}N_{UE}N_B i) + N_{sche}^{(slot)}N_B\mathcal{O}\left(\frac{N_{UE}N_{PC}}{N_B}\right)\right)\\ &= \mathcal{O}(N_{sched}N_{dim}N_{UE}N_B i) + \mathcal{O}\left(N_{sched}N_{sched}^{(slot)}N_{UE}N_{PC}\right)\end{aligned}$$

In this case, we need to distinguish two cases, based on the considered feature space. In fact, when we consider clustering in the space of the channel coefficients, we can state that $N_{dim} \gg N_{sched}^{(slot)}$; moreover, we can also expect that the number of beams is (slightly) larger than the number of priority classes, *i.e.*, $N_B > N_{PC}$. As such, the first term becomes dominant:

$$\mathcal{O}_{kmeans}^{(CSI)} \sim \mathcal{O}(N_{sche}\ N_R N_{UE} N_B i) \tag{38}$$

On the other hand, when we consider clustering in the geographical space, we have $N_{dim} = 2 \ll N_{sched}^{(slot)}$. In this case, even though $N_B > N_{PC}$ and $i = 100$ in the worst case as previously mentioned, the impact is negligible and the dominant term is the second one:

$$\mathcal{O}_{kmeans}^{(GEO)} \sim \mathcal{O}\left(N_{sched}N_{sched}^{(slot)}N_{UE}N_{PC}\right) \tag{39}$$

With respect to kmedoids, combining equation (31) with (36) and (37):

$$\begin{aligned}\mathcal{O}_{kmedoids} &= N_{sche}\ \left(\mathcal{O}(N_{dim}N_{UE}^2N_B i) + N_{sche}^{(slot)}N_B\mathcal{O}\left(\frac{N_{UE}N_{PC}}{N_B}\right)\right)\\ &= \mathcal{O}(N_{sche}\ N_{dim}N_{UE}^2N_B i) + \mathcal{O}\left(N_{sche}\ N_{sche}^{(slot)}N_{UE}N_{PC}\right)\end{aligned}$$

The same considerations made for kmeans in the case of CSI reports are also valid for kmedoids. However, when we consider clustering in the geographical space, the increased complexity with respect to N_{UE} leads to the first term still being dominant with respect to the second as $N_{UE} \gg N_{sched}^{(slot)}$. Thus, we have:

$$\mathcal{O}_{kmedoids}^{CSI} \sim \mathcal{O}(N_{sched}N_R N_{UE}^2 N_B i) \tag{40}$$

in the case of CSI reports, and:

$$\mathcal{O}_{kmedoids}^{GEO} \sim \mathcal{O}(N_{sched}N_{UE}^2 N_B i) \tag{41}$$

in the case of location reports.

With respect to HAC, we distinguish two cases. In the first case, we set $N_C = N_B$, leading to:

$$\begin{aligned}\mathcal{O}_{HAC} &= N_{sched}\left(\mathcal{O}(N_{UE}^2) + N_{sche}^{(slot)}N_B\mathcal{O}\left(\frac{N_{UE}N_{PC}}{N_B}\right)\right)\\ &= \mathcal{O}(N_{sch}\ \ N_{UE}^2) + \mathcal{O}\left(N_{sched}N_{sched}^{(slot)}N_{UE}N_{PC}\right)\end{aligned}$$

From the extensive discussions in D4.5, we can state that in the vast majority of scenarios the number of priority classes will be limited to a few units and in the considered system $N_{sc}^{(slot)}$ can be a few hundreds (*e.g.*, 200 slots means 2 seconds). As such, $N_{UE} \gg N_{PC}$ and, thus:

$$\mathcal{O}_{HAC} \sim \mathcal{O}(N_{sched}N_{UE}^2) \tag{42}$$

In the case of HAC-3, where we generate a number of clusters $N_C = 3N_B$, combining equation (33) with (36) and (37) we obtain:

$$\mathcal{O}_{HAC-3} = N_{sched}\left(\mathcal{O}(N_{UE}^2) + N_{sched}^{(slot)} N_B \mathcal{O}\left(\frac{N_{UE}N_{PC}}{N_B}\right) + N_{sched}^{(slot)}\mathcal{O}(N_B^2 N_C^2)\right)$$

$$= \mathcal{O}(N_{sched}N_{UE}^2) + \mathcal{O}\left(N_{sched}N_{sched}^{(slot)}N_{UE}N_{PC}\right) + \mathcal{O}\left(N_{sche}\ N_{sched}^{(slot)}N_B^2 N_C^2\right)$$

While the number of users N_{UE} is clearly larger than either N_B or N_C, we observed that $N_{UE} \leq N_B N_C = 3N_B^2$ and, thus, we obtain:

$$\mathcal{O}_{HAC-} \sim \mathcal{O}\left(N_{sche}\ N_{sched}^{(slot)}N_B^2 N_C^2\right) \tag{43}$$

For what regards spectral clustering, we can combine equation (34) with (36) and (37), resulting in the following:

$$\mathcal{O}_{Spectral} = N_{sched}\left(\mathcal{O}(N_R N_{UE}^2 + N_C N_{UE}^2 + N_{UE}N_C^2 i) + N_{sch}^{(slot)} N_B \mathcal{O}\left(\frac{N_{UE}N_{PC}}{N_B}\right) + N_{sched}^{(slot)}\mathcal{O}(N_B^2 N_C^2)\right)$$
$$= \mathcal{O}(N_{sched}N_R N_{UE}^2) + \mathcal{O}(N_{sched}N_C N_{UE}^2) + \mathcal{O}(N_{sched}N_{UE}N_C^2 i) + \mathcal{O}\left(N_{sched}N_{sched}^{(slot)}N_{UE}N_{PC}\right)$$
$$+ \mathcal{O}\left(N_{sched}N_{sched}^{(slot)}N_B^2 N_C^2\right)$$

As previously mentioned, spectral clustering is suitable for CSI reports and $N_R \gg N_C$; thus, $\mathcal{O}(N_{sched}N_C N_{UE}^2) \ll \mathcal{O}(N_{sched}N_R N_{UE}^2)$. Furthermore, as previously stated, it is fair to assume $N_{UE} \gg N_{PC}$ and $N_R \gg N_{sched}^{(slot)}$: thus, we also have $\mathcal{O}\left(N_{sched}N_{sched}^{(slot)}N_{UE}N_{PC}\right) \ll \mathcal{O}(N_{sched}N_R N_{UE}^2)$. Finally, we observed that N_C is directly proportional to N_{UE} and, in general, $N_C \sim N_{UE}/10$. As such, also considering $i = 100$ in the worst case, we have $\mathcal{O}(N_{sched}N_{UE}N_C^2 i) \sim \mathcal{O}(N_{sched}N_{UE}^3)$ and $\mathcal{O}\left(N_{sched}N_{sched}^{(slot)}N_B^2 N_C^2\right) \sim \mathcal{O}(N_{sched}N_B^2 N_{UE}^2) \ll \mathcal{O}(N_{sched}N_{UE}N_C^2 i)$. Thus, we obtain:

$$\mathcal{O}_{Spectral} \sim \mathcal{O}(N_{sche}\ N_{UE}N_C^2 i) \tag{44}$$

In the case of DBSCAN, equation (35) can be combined with (36) and (37), resulting in the following:

$$\mathcal{O}_{DBSCAN} = N_{sche}\ \left(\mathcal{O}(N_{UE}\log(N_{UE})) + N_{sche}^{(slot)} N_B \mathcal{O}\left(\frac{N_{UE}N_{PC}}{N_B}\right) + N_{sched}^{(slot)}\mathcal{O}(N_B^2 N_C^2)\right)$$
$$= \mathcal{O}(N_{sc}\quad N_{UE}\log(N_{UE})) + \mathcal{O}\left(N_{sche}\ N_{sc}^{(slot)}N_{UE}N_{PC}\right) + \mathcal{O}\left(N_{sche}\ N_{sche}^{(slot)}N_B^2 N_C^2\right)$$

Having observed $N_C \sim N_{UE}/10$ as in spectral clustering, the following formulation can be directly obtained:

$$\mathcal{O}_{DBSCAN} \sim \mathcal{O}\left(N_{sche}\ N_{sche}^{(slot)}N_B^2 N_C^2\right) \tag{45}$$

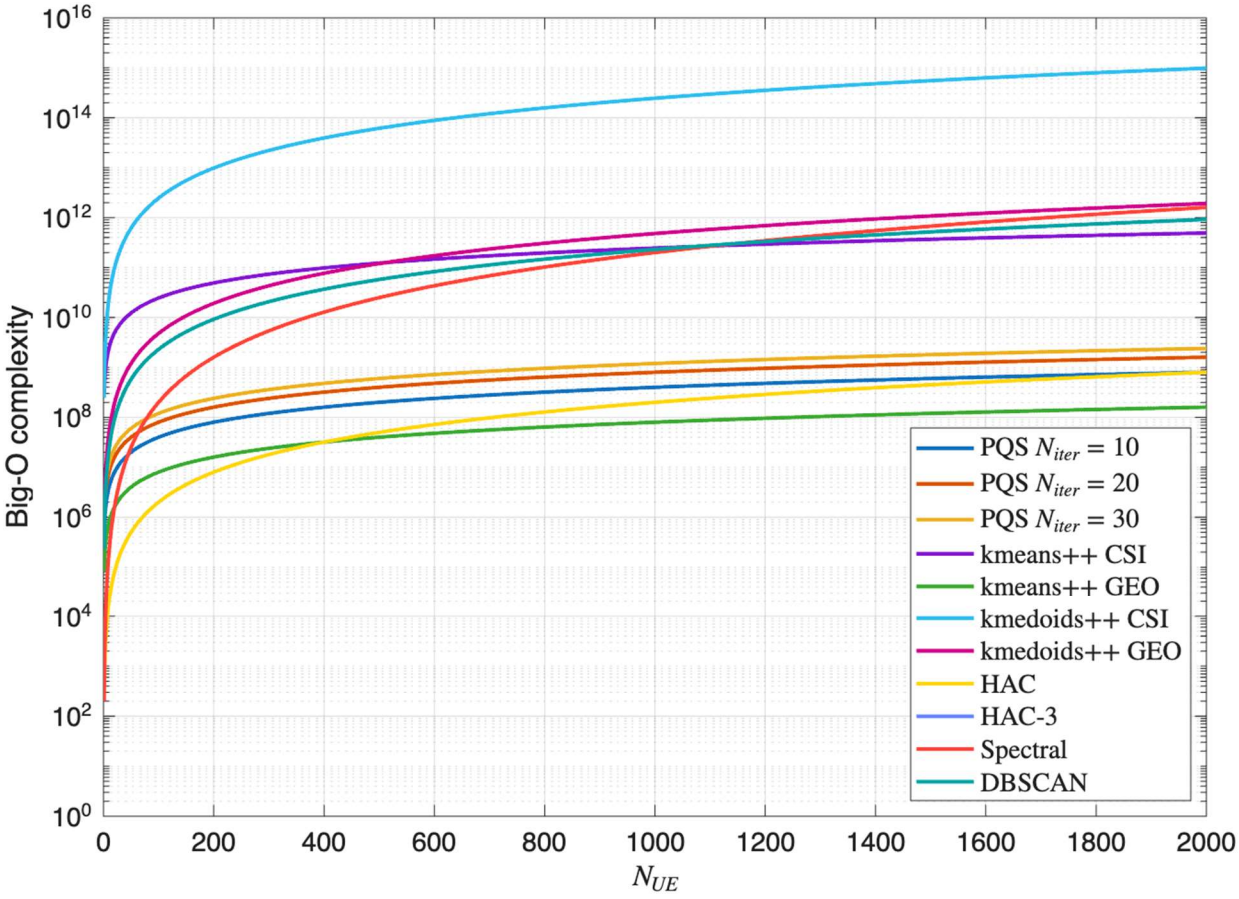


Figure 33: Comparison of the computational complexity of the benchmark PQS algorithm (D4.5) and the CA-PQS with the considered clustering algorithms. Note: HAC-3 is overlapped with DBSCAN.

From D4.5 Section 5.3.1.3, the complexity of the benchmark PQS algorithm is given by:

$$\mathcal{O}_{PQS} = \mathcal{O}\left(N_{UE} N_{iter} N_{sched} N_{sche}^{(slot)}\right) \tag{33}$$

in which N_{iter} denotes the number of iterations to find a user meeting the selected scheduling constraint. Figure 33 shows a comparison of the computational complexities of the benchmark PQS and the CA-PQS implemented with both all reported clustering algorithms. Please note that the HAC-3 and DBSCAN curves overlap. In this example, we considered $N_{iter} = 10, 20, 30$, $N_{PC} = 2$, $N_{sch} = 200$, and $N_{sche}^{(slot)} = 200$. These values are in line with those obtained/assumed during the performance assessment in Tasks 4.2 and 4.3. Additionally, we set $i = 100$ as a worst-case scenario for kmeans++, kmedoids++, and spectral clustering, and $N_C = N_{UE}/10$ for spectral clustering and DBSCAN. It can be noticed that CA-PQS with kmedoids++ in the CSI space has a significantly larger computational complexity with respect to other solutions; this is a combination of the dimensionality of the feature space in this configuration and the quadratic dependence with respect to N_{UE}. Nonetheless, even in the GEO space the computational complexity of kmedoids++ is significantly larger than PQS and comparable to that of kmeans++ in the CSI space. The remaining clustering algorithms that operate under the assumption $N_C = N_B$, *i.e.*, HAC (regardless of the type of report) and kmeans++ in the GEO space, are able to achieve a computational complexity consistently lower than the benchmark PQS algorithm. In particular, kmeans++ is the best clustering algorithm from the computational complexity point of view for $N_{UE} \geq 400$, which corresponds to most of the cases observed. On the opposite, all clustering algorithms assuming $N_C \geq N_B$

lead to a significantly higher complexity due to either the clustering phase (spectral clustering) or the inter-cluster scheduling phase (HAC-3 and DBSCAN).

4.2.5 Performance assessment

Table 8 summarises the simulation parameters for the performance assessment of CA-PQS, which are in line with those considered for the non-AI based system in D4.5. Table 9 provides the configuration of the CA-PQS algorithm, while Table 10 reports the parameters required by spectral clustering and DBSCAN, which were empirically optimized. The parameter w_s employed in the inter-cluster scheduling phase was optimized to minimize the unmet capacity for each clustering algorithm, user report space (GEO or CSI), beamforming technique (MMSE or CBF), and C_{min}/C_{max} pair; the result of this optimization is reported in Table 11 for HAC-3 in the GEO space, Table 12 for HAC-3 in the CSI space, Table 13 for spectral clustering in the CSI space, and Table 14 for DBSCAN in the GEO space. The configuration of the benchmark PQS algorithm designed in D4.5 is the optimal one, obtained in D4.5 and reported in Table 15 and Table 16 for the sake of completeness, with the geographical and CSI scheduling constraints, respectively. As discussed above, the designed CA-PQS does not need to define any constraint, since the users' separation in the geographical or CSI space is obtained through clustering.

Table 8: Simulation parameters for CA-PQS.

Parameter	Value	Comments
Sub-Carrier Spacing (SCS)	120 kHz	In this case, 132 PRBs are available
Operating Frequency Range	FR2 (Ka-band)	See 3GPP TR 38.821
Carrier frequency	20 GHz	See 3GPP TR 38.821
User bandwidth B	190.08 MHz	$132 \cdot 120 \cdot 12$ kHz
Scenario	Digital Divide	See Section 5.2.1.2 in D4.5
Minimum capacity request, C_{min}	5,10,20 Mbps	See Section 5.2.1.2 in D4.5
Maximum capacity request, C_{max}	100,300,500 Mbps	See Section 5.2.1.2 in D4.5
Number of radiating elements $N_{R,sat}$	512	See Section 2.3.1 of D4.3
Power per radiating element $P_{av,el}$	65 mW	See Section 2.3.1 of D4.3
Total power on-board P_{av}	15.2218 dBW	$10\log_{10}(0.0065 \cdot 512)$
Minimum user elevation angle ε_t	30°	See Section 4.2.3 of D4.3
Service area elevation angle ε_S	30°	See Section 4.2.3 of D4.3
Channel type	clear-sky	See Section 2.4.2.1 of D4.3 and 3GPP TR 38.811
Reporting periodicity T_{rep}	80 ms (8 slots)	See Section 5.3.1.1 in D4.5
Transmission time slot T_{slot}	10 ms	See Section 5.3.1.1 in D4.5
NTN node altitude	1000 km	See Section 2.1 of D4.3
Service area centre	(45°N, 5°E)	See Section 4.2.3 of D4.3
UE terminal	VSAT	See Section 2.3.2 of D4.3 and 3GPP TR 38.811

Table 9: Configuration parameters for the CA-PQS algorithm.

Parameter	Value	Comments
Scheduling periodicity T_{sched}	2 s (200 slots)	See Section 5.3.1.1 in D4.5
Maximum residual visibility σ_{vis}	50	With $T_{slot} = 10$ ms, it corresponds to 0.5 s
Unmet capacity factor σ_{cap}	2	

Number of Priority Classes N_{PC}	2	See Section 5.3.1.2 in D4.5

Table 10: Clustering parameters for the CA-PQS algorithm.

Parameter	Value	Comments
Similarity matrix threshold ε_S	0.7	For spectral clustering
Minimum number of points $N_{P,min}$	1	For DBSCAN
Neighbourhood radius ε_r	0.03	For DBSCAN

Table 11: Optimal threshold value with HAC-3 clustering and geographical scheduling.

Scheduler configuration	C_{min}	(LB-)MMSE			CBF		
		C_{max}					
		100 Mbps	300 Mbps	500 Mbps	100 Mbps	300 Mbps	500 Mbps
$\sigma_{vis} = 50$ ms $\sigma_{\text{cap}} = 2$	5 Mbps	0.0	0.7	0.9	0.0	0.7	0.9
	10 Mbps	0.7	0.8	0.8	0.7	0.8	0.8
	20 Mbps	0.6	0.5	0.4	0.5	0.5	0.5

Table 12: Optimal threshold value with HAC-3 clustering and CSI scheduling.

Scheduler configuration	C_{min}	(LB-)MMSE			CBF		
		C_{max}					
		100 Mbps	300 Mbps	500 Mbps	100 Mbps	300 Mbps	500 Mbps
$\sigma_{vis} = 50$ ms $\sigma_{\text{cap}} = 2$	5 Mbps	0.1	0.8	0.9	0.2	0.8	0.9
	10 Mbps	0.9	0.9	0.9	0.9	0.9	0.9
	20 Mbps	0.8	0.8	0.8	0.8	0.8	0.8

Table 13: Optimal threshold value with spectral clustering and CSI scheduling.

Scheduler configuration	C_{min}	(LB-)MMSE			CBF		
		C_{max}					
		100 Mbps	300 Mbps	500 Mbps	100 Mbps	300 Mbps	500 Mbps
$\sigma_{vis} = 50$ ms $\sigma_{\text{cap}} = 2$	5 Mbps	0.5	0.7	0.5	0.5	0.7	0.5
	10 Mbps	1.0	1.0	1.0	1.0	1.0	1.0
	20 Mbps	0.6	1.0	0.8	0.6	1.0	0.8

Table 14: Optimal threshold value with DBSCAN clustering and geographical scheduling.

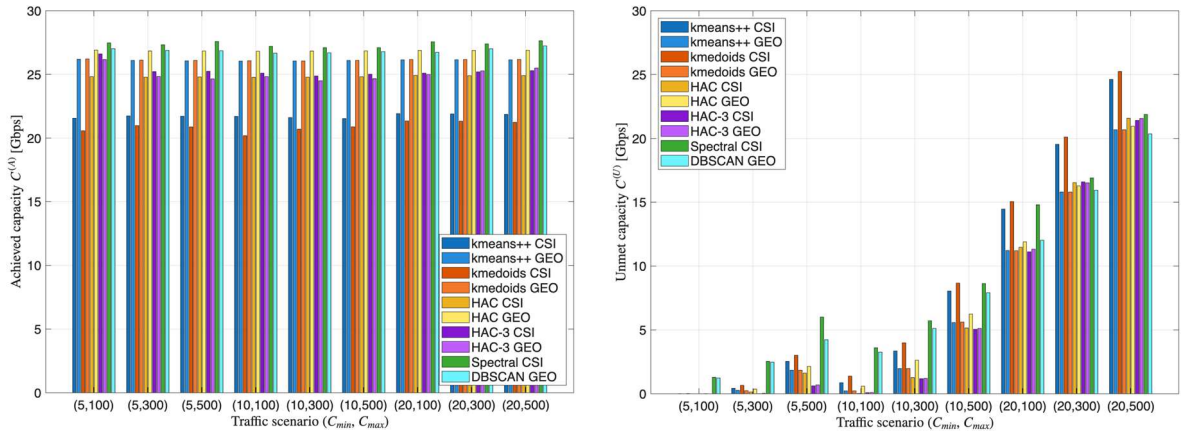
Scheduler configuration	C_{min}	(LB-)MMSE			CBF		
		C_{max}					
		100 Mbps	300 Mbps	500 Mbps	100 Mbps	300 Mbps	500 Mbps
$\sigma_{vis} = 50$ ms $\sigma_{\text{cap}} = 2$	5 Mbps	0.7	0.9	1.0	0.7	0.9	1.0
	10 Mbps	1.0	1.0	1.0	1.0	1.0	1.0
	20 Mbps	1.0	1.0	1.0	1.0	1.0	1.0

Table 15: Optimal performance of the benchmark PQS algorithm with geographical scheduling constraint.

Scheduler configuration	C_{min}	(LB-)MMSE			CBF		
		C_{max}					
		100 Mbps	300 Mbps	500 Mbps	100 Mbps	300 Mbps	500 Mbps
$\sigma_{vis} = 50$ ms $\sigma_{\text{cap}} = 2$	5 Mbps	3	4	4	3	4	4
	10 Mbps	4	4	4	4	4	4
	20 Mbps	4	4	4	4	4	4

Table 16: Optimal performance of the benchmark PQS algorithm with CSI scheduling constraint.

Scheduler configuration	C_{min}	(LB-)MMSE			CBF		
		C_{max}					
		100 Mbps	300 Mbps	500 Mbps	100 Mbps	300 Mbps	500 Mbps
$\sigma_{vis} = 50$ ms $\sigma_{\text{cap}} = 2$	5 Mbps	0.09	0.05	0.06	0.07	0.05	0.05
	10 Mbps	0.05	0.05	0.05	0.04	0.04	0.04
	20 Mbps	0.04	0.04	0.04	0.04	0.04	0.04



(a) Achieved capacity $C^{(A)}$

(b) Unmet capacity $C^{(U)}$

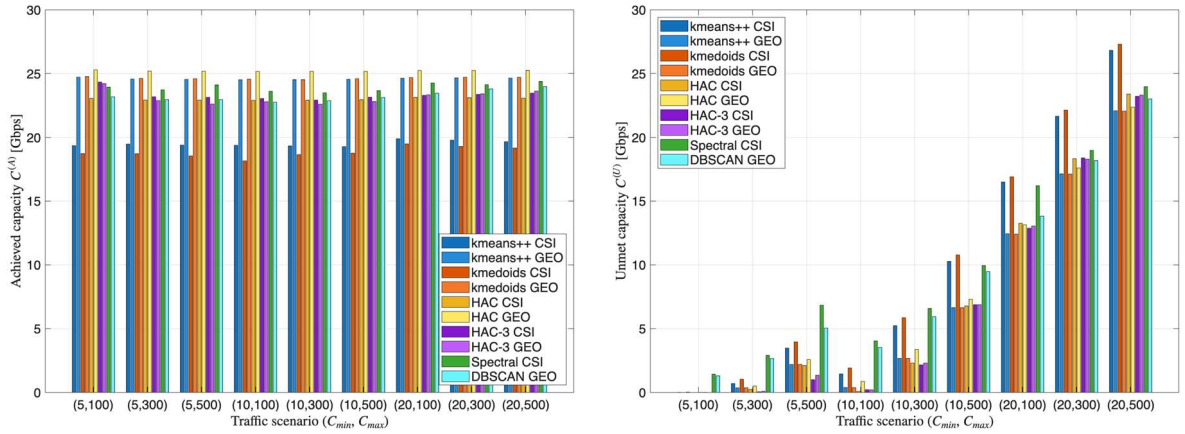
Figure 34: Achieved and unmet capacity with CA-PQS for (LB-)MMSE beamforming.

Figure 34 and Figure 35 show the achieved and unmet capacity with (LB-)MMSE and CBF beamforming, respectively. We can observe the following main performance trends:

- In line with the analyses discussed in D4.5 for the PQS and PRS algorithms, and as expected based on previous studies, MMSE-based solutions provide a better performance compared to digital beam steering (CBF), in terms of both the achieved and unmet capacities. In most of the cases, the difference is quite limited, in the order of 1-1.5 Gbps. However, spectral clustering and DBSCAN lead to a more significant gap of around 3.5 Gbps.

- Differently from the benchmark PQS algorithm discussed in D4.5, fixing the scheduling algorithm, the achieved capacity of CA-PQS is almost constant for all the considered traffic scenarios. Despite considering also in this case a relatively simple implementation of the priority classes, with $N_{PC} = 2$, CA-PQS has a significant advantage compared to PQS: at the beginning of each scheduling period, the clusters are computed again based on the received reports from the visible users. This allows to take into account the updated status of the geographical locations in direction cosines or the CSI of the users, and tailor the clusters to the current conditions. With PQS, the geographical or CSI scheduling constraint is fixed and, as we have extensively discussed in D4.5, different optimal values are obtained depending on the traffic conditions. From a certain perspective, clustering the users is equivalent to having an adaptive scheduling constraint that is automatically adjusted based on the current ancillary information, with a clear performance benefit.

- Also with CA-PQS, the unmet capacity increases for increasing traffic because the scheduler is not able to find enough resources to satisfy the large requested traffic.



(a) Achieved capacity $C^{(A)}$

(b) Unmet capacity $C^{(U)}$

Figure 35: Achieved and unmet capacity with CA-PQS for CBF beamforming.

Observing the clustering algorithms, we can notice that for both kmeans++, kmedoids++, and HAC geographical clustering outperforms a partitioning based on the channel coefficients. This might seem puzzling, considering how the CSI scheduling constraint approach for PQS and PRS in D4.5 outperformed the geographical one, but there is a clear motivation: the **curse of dimensionality**. In clustering, as the number of dimensions (features) increases, the distance metric that is used by the partitioning algorithm becomes less meaningful; this leads to a performance loss due to: i) distance metrics becoming dominated by noise rather than by meaningful data structures to be exploited for partitioning; ii) the definition of the clusters might become ambiguous, making it difficult to determine the cluster boundaries; and iii) data points in large dimension spaces tend to become more sparse. Recalling that kmeans++ and kmedoids++ tend to identify hyper-spherical clusters, the latter point motivates the very poor performance of these algorithms when considering CSI, compared to HAC that exploits trees of clusters that can still reveal some relationships also with sparse data points. Thus, while both kmeans++, kmedoids++, and HAC show a poor performance with channel feature spaces, only the former two are particularly affected by the curse of dimensionality. Furthermore, HAC-3 shows that increasing the number of clusters for hierarchical clustering in the CSI space further alleviates this effect, leading to an improved achieved capacity and

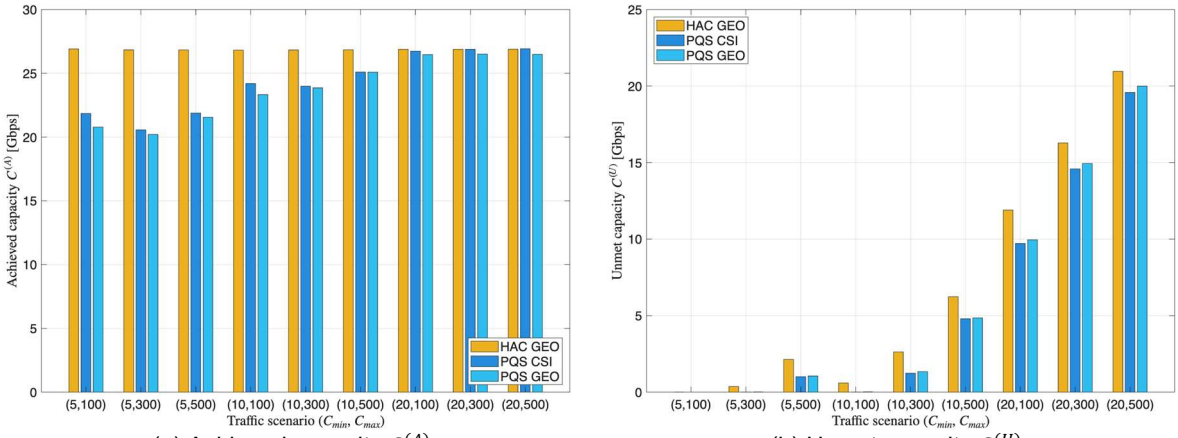
unmet capacity at low traffic scenarios. In particular, it should be noted that HAC-3 achieves the lowest unmet capacity in low traffic scenarios. Finally, spectral clustering leads to the highest achieved capacity by computing a low-dimensional representation of the channel features, leading to the identification of more meaningful features for a more effective clustering. Nonetheless, this comes at the expense of a significant increase in unmet capacity, particularly in low traffic scenarios. To improve the clusters quality, a strategy to periodically adapt the spectral clustering similarity threshold ε_S to the channel features' distribution could be developed.

In all traffic conditions, a **geographical feature space** thus provides, in general, a better performance in both the achieved and residual unmet capacity. As for spectral clustering in the CSI space, DBSCAN leads to the highest achieved capacity in the geographical feature space but fails at reducing the unmet capacity in low traffic scenarios. Comparing all considered clustering algorithms for this feature space, we can notice that **HAC** has a slightly better performance and can be considered as the optimal partitioning strategy in the considered system. To compare the results of CA-PQS with the optimal performance of the benchmark PQS in D4.5, we consider this configuration.

Figure 36 and Figure 37 report the comparison between the optimal CA-PQS performance and that obtained in the benchmark PQS in D4.5, again in terms of achieved and unmet capacity, with (LB-)MMSE and CBF, respectively. We can observe two clear trends:

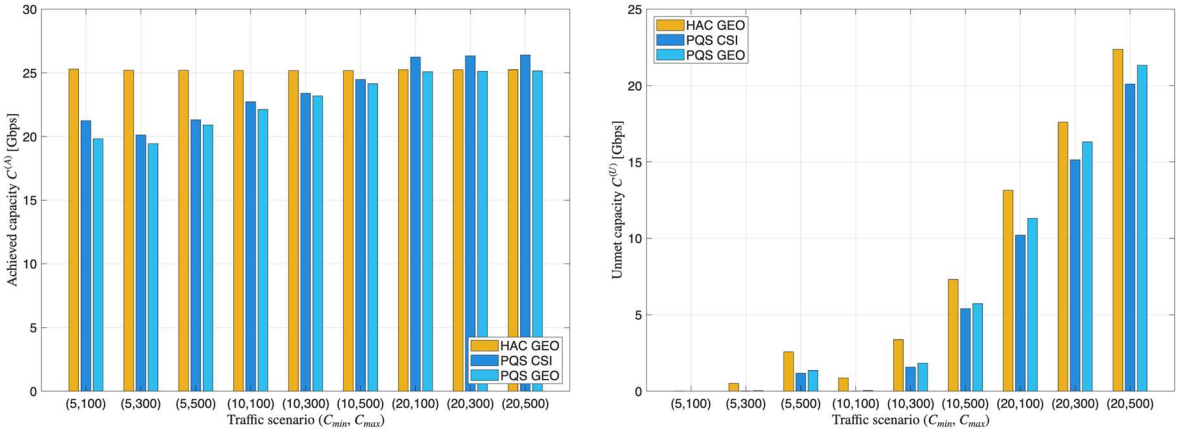
- The achieved capacity obtained with CA-PQS is significantly larger compared to the benchmark PQS algorithm, in particular in low-medium traffic conditions. When the traffic increases, the performance of the two algorithms becomes similar.

- In terms of residual unmet capacity, we can notice that the benchmark PQS algorithm is always slightly better compared to CA-PQS, in particular with the CSI scheduling constraint (which in D4.5 was shown to outperform the geographical one).



(a) Achieved capacity $C^{(A)}$ (b) Unmet capacity $C^{(U)}$

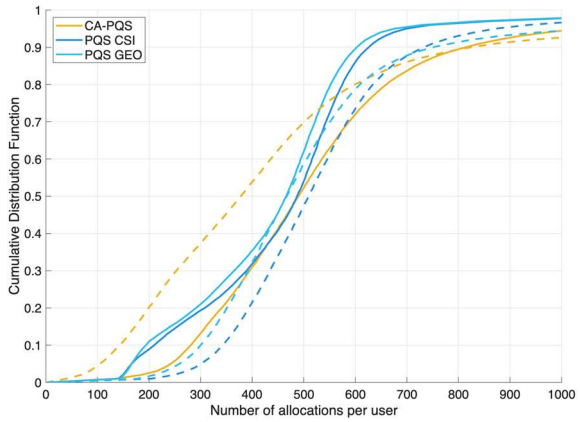
Figure 36: Comparison of the achieved and unmet capacity with CA-PQS (kmeans++ GEO) optimal PQS with (LB-)MMSE beamforming.



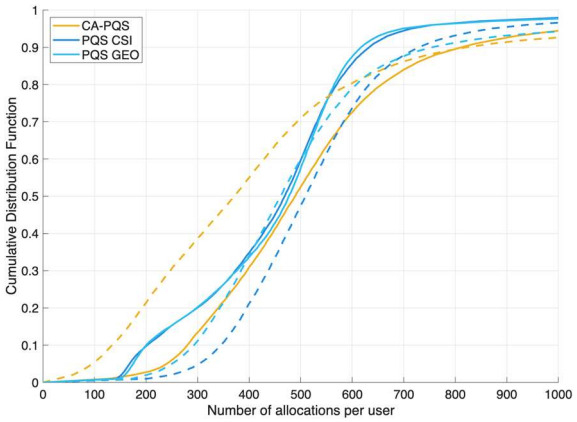
(a) Achieved capacity $C^{(A)}$ (b) Unmet capacity $C^{(U)}$

Figure 37: Comparison of the achieved and unmet capacity with CA-PQS (kmeans++ GEO) optimal PQS with CBF beamforming.

These two performance trends can be motivated observing that: i) as mentioned above, CA-PQS re-defines the clusters at the beginning of each scheduling period, allowing the scheduler to better tailor the resource allocation to the current geographical/CSI conditions and provide a stable achieved capacity; ii) CA-PQS always provides N_B users per time slot, *i.e.*, all the available user-centric beams are always generated and illuminated in each time slot. The latter point is particularly important; in fact, CA-PQS is always selecting one user per cluster and, since it is based on the defined partitioning, no scheduling constraint is implemented. However, it is possible that in a generic time slot the scheduler randomly picks two users from adjacent clusters that are in any case close to each other (*i.e.*, at the boundary between the two clusters). Thus, while providing an advantage in terms of computational complexity and stability of the achieved capacity, in some cases an ill-conditioned channel matrix can be provided to the beamformer with a performance loss; the effect is clear with the unmet capacity, while it is masked in the achieved capacity thanks to the increased number of allocations, as shown in Figure 38; in this figure, we can notice that:



(a) (LB-)MMSE



(b) CBF

Figure 38: Comparison of the CDF of the number of allocations per user with CA-PQS (HAC GEO) optimal PQS with (LB-)MMSE and CBF beamforming. Traffic configuration: $(5{,}100)$ *Mbps (solid line) and* $(20{,}500)$ *Mbps (dashed line).*

- In low traffic scenarios, CA-PQS allocates on average more resources per user. As reported, the benchmark PQS leaves some resources unused in particular in low traffic conditions, as the users with residual unmet capacity are limited: the scheduler decides to limit the users per time slot to increase the SINR, since there are enough slots to serve everyone. In CA-PQS, when a cluster has no user with unmet capacity, the cluster is built again without verifying the traffic request, thus always using all resources independently of the actual requests (which are in any case prioritised).

- When the traffic requests increase, the PQS algorithm tends to allocate more resources to the users, exploiting all or almost all of the unused beams that were present in the low traffic scenario: this time increasing the SINR per slot while reducing the number of allocations per user is not optimal, and the opposite approach is chosen by the scheduler.

- It is interesting to observe the behaviour of the allocations with CA-PQS when the traffic increases. As discussed above, when the traffic request is limited, the scheduler still allocates all beams to users; since the traffic requests are soon met, the remaining resources are allocated to other users leading to an increased number of allocations per user on average. When the requests are large, the users demanding more capacity are dominating the scheduling queues in each cluster; the result is that the average allocations per users are decreased and a small subset of the users in the service area drain the resources (this can be observed by the crossing between the low traffic and large traffic CDFs for large allocations).

4.2.6 Conclusions and future enhancements

The objective of this study was that of exploiting AI/ML solutions to improve the scheduling algorithms designed in D4.5 in terms of computational complexity, while maintaining the performance in terms of achieved and unmet capacity.

The designed CA-PQS (or CA-PRS, even though the performance has not been assessed in the latter case) meets these requirements: the computational complexity has been quite significantly reduced and the performance has been improved in all low-medium traffic scenarios. As for the large traffic scenarios, the performance without clustering is still on par

or better, in particular when considering the unmet capacity. This is implicitly motivated by the design of the CA-PQS algorithm: i) all the available user-centric beams are always generated and exploited; and ii) the control of too similar/close (in the CSI/geographical spaces) users in the same time slot is implicitly left to the clustering algorithm. The second observation, in particular, leads to some cases in which even if two users belong to two adjacent clusters, their similarity is still too large (*i.e.*, they are close at the boundary of their clusters).

To circumvent this issue, future enhancements of the designed solution might include the introduction of more advanced control capabilities in the inter-cluster scheduling phase based on the geographical or CSI scheduling constraints introduced in D4.5. The current inter-cluster algorithm enables the optimization of the trade-off between achieved capacity and unmet capacity by means of a dedicated parameter, leading to improved performance with respect to scheduling algorithms in which the number of clusters, N_C, is fixed to N_B. However, this comes at the expense of an increased computational complexity, which dominates the overall CA-PQS complexity when either HAC-3 or DBSCAN are implemented. An interesting direction is also that of designing a **hierarchical scheduler** in which the number of clusters that is generated at the beginning of the CA-PQS procedure, N_C, is not fixed a priori. One possible strategy is to evaluate the inconsistency coefficient of each node in the dendrogram and to aggregate the leaves (*i.e.*, the users) under a node which inconsistency lies under a predefined threshold. Another option is to aggregate leaves in a similar fashion based on the distance between each leaf and the node above it. Both strategies require the optimization of a dedicated threshold, ensuring that neither too many nor few clusters are generated. It is worth mentioning that such thresholds may need to be adapted based on the number of users in visibility, introducing an additional degree of freedom in the algorithm design and optimization. Finally, **Deep Reinforcement Learning** (DRL) techniques could be explored to directly map user reports to the set of scheduled users by means of a neural network.

4.3 AI-AIDED SINR ESTIMATION

4.3.1 Rationale

The SINR achieved on each communication link plays a major role in the effectiveness of the data exchange. On the one hand, the achievable rate in communication systems is limited by the SINR through the Shannon formula, *e.g.*, equation (11), effectively limiting the amount of bits that can be sent per time and bandwidth unit. On the other hand, procedures in communication systems rely on SINR-related metrics to take decisions, *e.g.*, handovers from one cell to another can be carried out once the measured Reference Signal Received Quality (RSRQ) overcome a certain threshold, [50]. Such proxy for link quality can typically be measured at the terminal using reference signals; however, NTNs introduce complications during the reporting phase: Geostationary-class satellites are characterized by a large propagation delay which can result in outdated reports, *e.g.*, in the case of user mobility, while Non-Geostationary-class satellites at lower orbit claim lower latency but introduce aging effects due to their inherent orbital movement. Furthermore, reporting occupies communication resources that can otherwise dedicated to data transmission: *e.g.*, scheduling reports in location-based beamforming NTN systems only contain the users' geographical coordinates, which can be estimated at the terminal by means of GNSS and thus do not require the transmission of dedicated pilot data from the NTN's side.

Focusing on user-centric beamforming NTN systems, the assessment of the SINR strongly relies on the set of users scheduled during the same time slot. Indeed, once a group of users is scheduled and the beamforming matrix is computed, either by means of MMSE, LB-MMSE, or CBF, the SINR achieved by each user can be evaluated using equation (10) (considering the theoretical clear sky channel formula based on the user's location in case of LB-MMSE).

However, such evaluation is based on the MMSE equation, which has a computational complexity in the order of $\mathcal{O}\left(N_{UE,sched}^2 N_R\right)$, where $N_{UE,sch}$ is the number of users scheduled during the same time slot and N_R represents the number of radiating elements on board of the satellite payload (see Section 4.3.4 for further details). Thanks to the function approximation capabilities of neural networks, the SINR achieved by a group of scheduled users in a user-centric beamforming NTN system may be evaluated based on the scheduling report with decreased computational complexity. For this purpose, the popular attention mechanism can be employed to identify relationships between users' scheduling reports, thus extracting information related to potential inter-user interference. Such algorithm can be pre-trained and deployed in the Near-RT RIC, providing a useful metric for tasks such as user scheduling (*e.g.*, to select a group of users from a candidate set based on the group's mean achievable SINR).

4.3.2 Self-attention basics

The attention mechanism was first introduced in the context of natural language processing to let algorithms selectively process sentences based on the relationships between words, [51]. Attention maps a triplet of vectors, namely the query $\boldsymbol{q}$, key $\boldsymbol{k}$, and value $\boldsymbol{v}$, to an output vector $\boldsymbol{a}$. The output vector $\boldsymbol{a}$ is, in general, computed as a weighted sum of $\boldsymbol{v}$, where the weights represent the compatibility of the query with the corresponding key. Intuitively, a mask is extracted from the query and the key and is then applied to the value vector, thus including only the relevant values in the output. The most popular version, named Scaled Dot-Product Attention (SDPA), was presented in [52] and is the key function of the famous transformer architecture at the basis of modern Large Language Models (LLMs). Considering a set of m queries and n pairs of unordered keys and values simultaneously, represented by the matrices $\mathbf{Q} \in \mathbb{R}^{m \times d_k}$, $\mathbf{K} \in \mathbb{R}^{n \times d_k}$, and $\mathbf{V} \in \mathbb{R}^{n \times d_v}$, respectively, the SDPA can be computed as:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = Softmax\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}} \cdot \mathbf{M}\right)\mathbf{V}, \tag{34}$$

where d_k is the dimension of the queries and keys vectors, d_v is the size of the values vectors, and $\mathbf{M} \in \mathbb{R}^{m \times n}$ is an input matrix that masks out values that should not be considered for the computation of the attention scores. Clearly, the first term of the matrix multiplication represents the weights, which are normalized to 1 for each query through the softmax function:

$$Softmax(\boldsymbol{h})_i = \frac{e^{h_i}}{\sum_{j=1}^{n} h_j}. \tag{35}$$

In the case of Self-Attention (SA), $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are computed from the same input matrix $\mathbf{J} \in \mathbb{R}^{n \times d}$, typically representing a sequence of $n = m$ tokens with embedding size d, by means of three Fully Connected (FC) layers with d_k or d_v neurons applied to each token separately:

$$\begin{aligned} \mathbf{Q} &= \mathbf{J}\mathbf{W}^Q, \mathbf{W}^Q \in \mathbb{R}^{d \times d_k} \\ \mathbf{K} &= \mathbf{J}\mathbf{W}^K, \mathbf{W}^K \in \mathbb{R}^{d \times d_k} \\ \mathbf{V} &= \mathbf{J}\mathbf{W}^V, \mathbf{W}^V \in \mathbb{R}^{d \times d_v}. \end{aligned} \tag{36}$$

Furthermore, the authors in [52] observed that the SDPA mechanism is able to extract better attention scores if separately applied to different sets of queries, keys, and values computed on the same input matrix. Thus, $\mathbf{J}$ can be linearly projected h times in parallel using a different set of weights for each head, represented by the matrices $\{\mathbf{W}_{\mathrm{i}}^Q\}_{i=1}^h$, $\{\mathbf{W}_{\mathrm{i}}^K\}_{i=1}^h$, and $\{\mathbf{W}_{\mathrm{i}}^V\}_{i=1}^h$, to obtain a set of h queries matrices $\{\mathbf{Q}_i\}_{i=1}^h = \mathbf{J}\{\mathbf{W}_{\mathrm{i}}^Q\}_{i=1}^h$ and a set of h pairs of keys and values

matrices $\{\mathbf{K}_i\}_{i=1}^{h} = \mathbf{J}\{\mathbf{W}_i^K\}_{i=1}^{h}$ and $\{\mathbf{V}_i\}_{i=1}^{h} = \mathbf{J}\{\mathbf{W}_i^V\}_{i=1}^{h}$, with $d = h \cdot d_k$ holding true. h SDPA modules are implemented to obtain h separate attention scores, which can then be concatenated and processed with a last FC layer with weights matrix $\mathbf{W}^O \in \mathbb{R}^{d_v \times d}$ to obtain the Multi-Headed Self Attention (MHSA) matrix (Figure 39):

$$MHSA = Concatenate(head_1, head_2, \dots, head_h)\mathbf{W}^O, \quad (37)$$
$$head_i = Attention(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i).$$

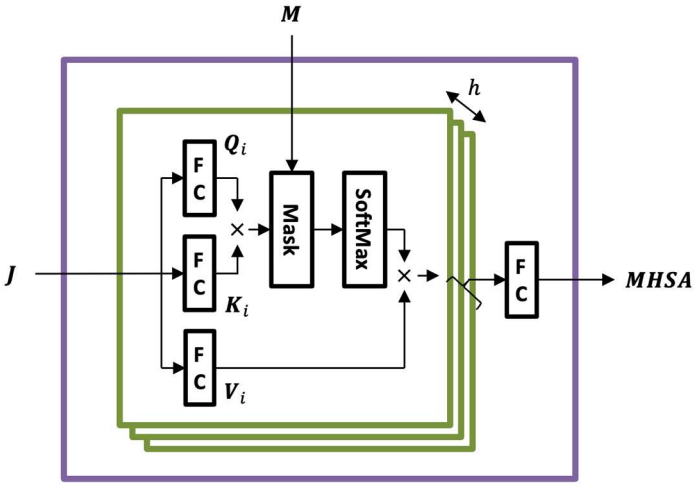


Figure 39: Multi-Headed Self-Attention diagram.

4.3.3 DMHSA model for SINR estimation in user-centric beamforming systems

The considered deep learning approach to SINR estimation is a Dual MHSA (DMHSA) model. The neural network takes as input a matrix $\mathbf{F} \in \mathbb{R}^{N_{UE,sched} \times \delta}$ containing a set of δ features for each of the N_{UE} scheduled users (with $N_{UE,sched} \leq N_B$, where $N_B = 24$ is the maximum number of beams that the considered antenna array can generate). It should be noted that a different number of users may be scheduled in different samples within a batch: to ensure that computation in the DMHSA model can be efficiently parallelized, a binary mask $\mathbf{m}^{(DMHSA)} = \left[\mathbf{1}^{1 \times N_{UE}}, \mathbf{0}^{1 \times (N_B - N_{UE,sche})}\right] \in \mathbb{R}^{1 \times N_B}$ (where $\mathbf{1}^{a \times b}$ and $\mathbf{0}^{a \times b}$ represent the $a \times b$ matrix containing only ones and zeros, respectively) is also given as input to the model and applied to each operation for each input sample. Thus, the model applies a function $f^{(DHMSA)}(\cdot)$ that maps the features matrix $\mathbf{F}$ to the corresponding vector of estimated SINR $\widehat{\mathbf{Y}}^{(DMHSA)} = f^{(DMHSA)}(\mathbf{F}, \mathbf{m}^{(DMHSA)})$ based on the binary mask. For a generic user k, the corresponding features vector $\mathbf{F}_{k,:} \in \mathbb{R}^{\delta}$ is generated as follows:

- In the case of CSI-based beamforming, the polar representation is selected. To reduce the dimensionality of the data, three types of information are concatenated to form the features vector: 1) the standardized argument of the user's CSI vector (*i.e.*, the phase of the CSI divided by its standard deviation $\sigma_\varphi = \pi/\sqrt{3}$, under the assumption $\angle\hat{\mathrm{h}}_{k,n}^{(t)} \sim U(-\pi, \pi)\ \forall k, n$), $\left[\angle\hat{\mathrm{h}}_{k,1}^{(t)}/\sigma_\varphi, \dots, \angle\hat{\mathrm{h}}_{k,n}^{(t)}/\sigma_\varphi, \dots, \angle\hat{\mathrm{h}}_{k,N_R}^{(t)}/\sigma_\varphi\right]$; 2) the standardized mean squared absolute value of the user's CSI vector, $\left|\tilde{\mathrm{h}}_k^{(t)}\right|^2 = (\sum_{n=1}^{N_R}\left|\hat{\mathrm{h}}_{k,n}^{(t)}\right|^2 - N_R\mu_h)/N_R\sigma_h$, with μ_h and σ_h being standardization parameters to be determined statistically; and 3) the ratio $N_{UE,sched}/N_B$. It should be noted that, given the considered channel model, $\left|\tilde{\mathrm{h}}_k^{(t)}\right|^2 = \left|\hat{\mathrm{h}}_{k,n}^{(t)}\right|^2\ \forall n \in 1, \dots, N_R$ as the path between a user and each radiating element

is assumed to be subject to the same attenuation phenomena. The features vector with CSI reporting has size $\delta^{(CSI)} = N_R + 2$ and can be expressed as:

$$\mathbf{F}_{k,:}^{(CSI)} = \left[\frac{\angle\hat{\mathrm{h}}_{k,1}^{(t)}}{\sigma_\varphi}, \ldots, \frac{\angle\hat{\mathrm{h}}_{k,n}^{(t)}}{\sigma_\varphi}, \ldots, \frac{\angle\hat{\mathrm{h}}_{k,N_R}^{(t)}}{\sigma_\varphi}, \left|\tilde{\mathrm{h}}_k^{(t)}\right|^2, \frac{N_{UE,sche}}{N_B}\right]. \quad (38)$$

- In the case of location-based beamforming, the user's location is computed in the u-v reference system based on the location report. Then, the pair of coordinates $[u_k, v_k]$ is concatenated with the ratio $N_{UE,sche}\ /N_{beams}$. Thus, the features vector with location reporting has size $\delta^{(GEO)} = 3$ and can be expressed as:

$$\mathbf{F}_{k,:}^{(GEO)} = \left[u_k, v_k, \frac{N_{UE,sched}}{N_B}\right]. \quad (39)$$

It is worth mentioning that the addition of the $\frac{N_{UE}}{N_{beams}}$ ratio provides information related to the amount of interference to be expected, thus improving the SINR estimation accuracy.

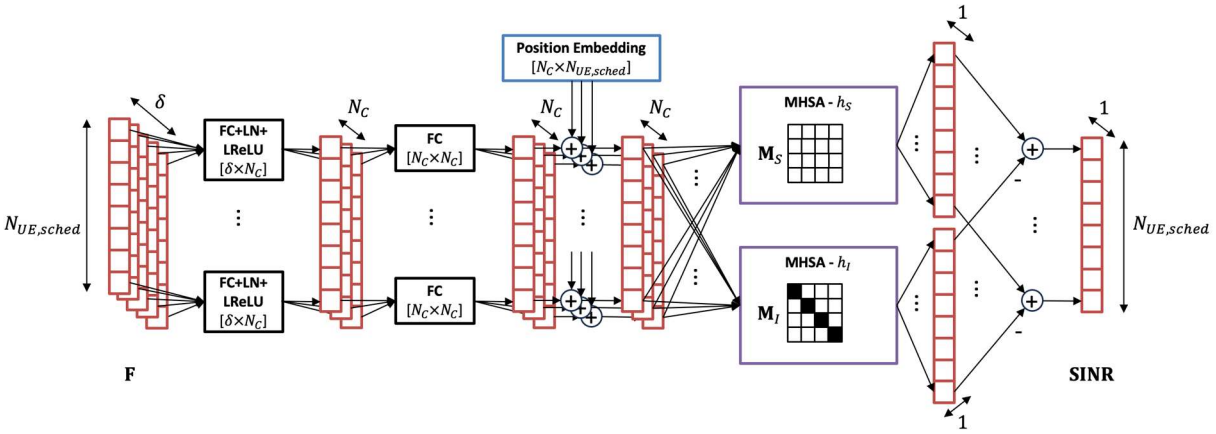


Figure 40: DMHSA SINR estimation model.

The deep learning model employed for SINR estimation is depicted in Figure 40. In the first section, a cascade of layers extracts embeddings from the features matrix $\mathbf{F}$, acting on each user's features vector independently. The sequence includes:

- FC layers with $N_C = 8$ neurons, which perform a matrix multiplication between their input vector $\mathbf{x}^{(FC)}$ and a matrix of learnable weights $\mathbf{W}^{(FC)}$ and sums the output with a learnable bias vector $\mathbf{b}^{(FC)}$:

$$\mathbf{y}^{(FC)} = \mathbf{x}^{(FC)}\mathbf{W}^{(FC)} + \mathbf{b}^{(FC)}. \quad (40)$$

- Layer Normalization (LN) layers, which normalize their input vector $\mathbf{x}^{(LN)}$ with the mean $\mu^{(LN)}$ and standard deviation $\sigma^{(LN)}$ of $\mathbf{x}^{(LN)}$ and apply two vectors of learnable parameters $\boldsymbol{\gamma}^{(LN)}$ and $\boldsymbol{\beta}^{(LN)}$:

$$\mathbf{y}^{(LN)} = \boldsymbol{\gamma}^{(LN)} \cdot \frac{\mathbf{x}^{(LN)} - \mu^{(LN)}}{\sigma^{(LN)} + \epsilon} + \boldsymbol{\beta}^{(LN)},$$
$$\mu^{(LN)} = \frac{1}{N_C} \sum_{n=1}^{N_C} \mathbf{x}_n^{(LN)}, \tag{41}$$
$$\sigma^{(LN)} = \sqrt{\frac{1}{N_C} \sum_{n=1}^{N_C} \left(\mathbf{x}_n^{(LN)} - \mu^{(LN)}\right)^2},$$

where ϵ is an arbitrary small value for numerical stability.

- Leaky Rectified Linear Unit (LReLU), which introduces non-linearities in the model by scaling each negative element of its input $\mathbf{x}^{(LReLU)}$ with a small value $\alpha^{(LReLU)}$:

$$\mathbf{y}_i^{(LRELU)} = \begin{cases} \mathbf{x}_i^{(LReLU)} & if\ \mathbf{x}_i^{(LReLU)} \geq 0 \\ \alpha^{(LReLU)} \mathbf{x}_i^{(LReLU)} & if\ \mathbf{x}_i^{(LReLU)} < 0 \end{cases} \tag{42}$$

- Position Embedding (PE), which creates a trainable vector of N_C weights for each user index to be summed to the corresponding user's generated features. In LLMs, PE can be used to inherently embed information regarding the distance between tokens (*i.e.*, word embeddings) in a sequence (*i.e.*, in a sentence), which provides aid to the MHSA in better understanding relationships between tokens. In SINR estimation, PE provides additional information on the number of scheduled users, thus improving the neural network's understanding of the amount of interference to be expected. Figure 41 reports the heatmap of the learnt PE vectors in the case of location-based SINR estimation, highlighting more extreme changes in PE for users with high index. Indeed, only the first $N_{UE,sche}$ columns are used when $N_{UE,sche}$ users are scheduled.

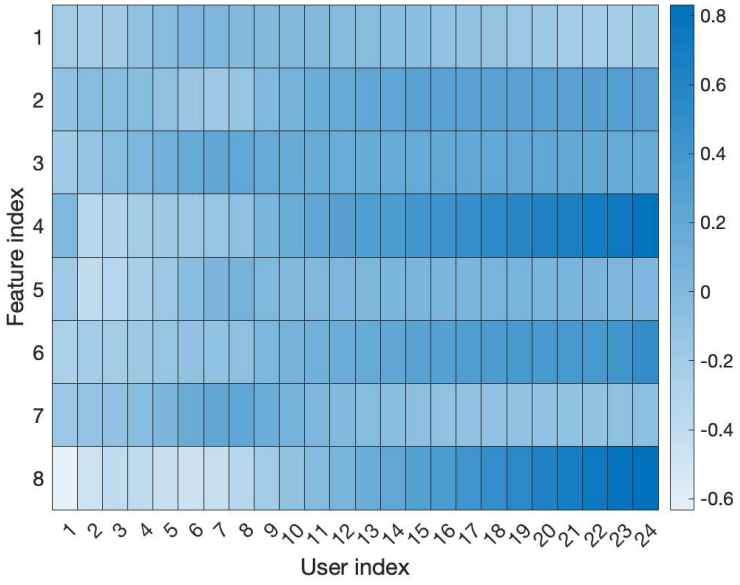


Figure 41: Example of learnt PE vectors ($N_C = 8$).

The obtained vectors are then fed to two parallel MHSA + FC modules that operate on the set of features of $N_{UE,sched}$ users. Based on equation (10), the rationale behind such structure is

to assess for each user the SNR in dB with the first module and $10\log_{10}(1+INR_k)$ with the second module. The first module employs $h_S = 4$ heads ($d_k = d_v = \frac{N_C}{h} = 2$) and does not include an attention mask (*i.e.*, $\mathbf{M}_S = \mathbf{1}^{N_{UE,sc} \times N_{UE,sch}}$) to extract all of the interactions between users; the second one includes $h_I = 4$ heads ($d_k = d_v = \frac{N_C}{h} = 2$) and an attention mask $\mathbf{M}_I = \mathbf{1}^{N_{UE,sche} \times N_{UE,sc}} - \mathbf{I}^{N_{UE,sc} \times N_{UE,sched}}$, where $\mathbf{I}^{a\times a}$ is the $a \times a$ identity matrix, in order to exclude the interactions between a user and itself and including only interference-related attention scores. The output of each MHSA step is fed to a dedicated FC layer with a single output neuron, which is applied to each user's features separately to compute the SNR or INR-related quantities. The standardized SINR in dB is computed by subtracting the second module's output from the first's, and the final output can be obtained by means of application of properly estimated standardization parameters.

4.3.4 Computational complexity

Regardless of the choice between CSI-based beamforming and location-based beamforming, the SINR can evaluated through equation (10) (with the theoretical CSI being computed based on the users' geographical location in case of location-based beamforming. When assessing the MMSE beamforming algorithm's complexity (Table 7), which is at the base of the SINR evaluation, two main sources of complexity can be identified:

- The matrix multiplication between the CSI matrix $\widehat{\mathbf{H}}^{(t)} \in \mathbb{C}^{N_{UE,sche} \times N_R}$ and its conjugate transpose $\widehat{\mathbf{H}}^{(t)}\left(\widehat{\mathbf{H}}^{(t)}\right)^{\boldsymbol{H}}$, with complexity $\mathcal{O}\left(N_{UE,sche}^2 N_R\right)$; and
- The computation of the inverse of a $N_{UE} \times N_{UE}$ complex matrix, $\left(\widehat{\mathbf{H}}^{(t)}\left(\widehat{\mathbf{H}}^{(t)}\right)^{\boldsymbol{H}} + \alpha \mathbf{I}_{N_B}\right)^{-\mathbf{1}}$, with complexity $\mathcal{O}\left(N_{UE,sche}^3\right)$.

Thus, the overall complexity of the SINR estimation through the computation of the MMSE beamforming equation is in the order of $\mathcal{O}\left(N_{UE,sched}^2 N_R\right) + \mathcal{O}\left(N_{UE,sche}^3\right)$. However, the number of scheduled users $N_{UE,sched}$ is typically limited with respect to the number of radiating elements N_R, *i.e.*, $N_{UE,sche} \ll N_R$. With this consideration, the MMSE complexity reduces to:

$$\mathcal{O}_{MMSE} = \mathcal{O}\left(N_{UE,sche}^2 N_R\right). \tag{43}$$

The computational complexity of the DMHSA model can be mainly attributed to FC and MHSA layers:

- The first FC layer embeds δ features into a vector of size N_C for each user, resulting in a computational complexity $\mathcal{O}\left(\delta N_{UE,sche} N_C\right)$.
- The second FC layers further refines the N_C features for each user, resulting in a computational complexity $\mathcal{O}\left(N_{UE,sc} N_C^2\right)$.
- For each head, the MHSA layers firstly apply linear projections to compute the queries, keys, and values matrices, achieving complexity $\mathcal{O}\left(N_{UE,sche} N_C d_k\right) = \mathcal{O}\left(N_{UE,sched} \frac{N_C^2}{h}\right)$ given $d_k = \frac{N_C}{h}$; the subsequent computations in the SDPA involve a matrix multiplication between queries and keys ($\mathcal{O}\left(N_{UE,sche}^2 d_k\right) = \mathcal{O}\left(N_{UE,sched}^2 \frac{N_C}{h}\right)$) and between the softmax outputs and the values ($\mathcal{O}\left(N_{UE,sch}^2 d_v\right) = \mathcal{O}\left(N_{UE,sched}^2 \frac{N_C}{h}\right)$); finally, the heads outputs are concatenated to form a vector of $N_C = h \cdot d_k$ features and passed through

one final FC layer ($\mathcal{O}(N_{UE,sched}N_C^2)$). Thus, the computational complexity of the MHSA layers is $h \cdot \left[2\mathcal{O}\left(N_{UE,sched}\frac{N_C^2}{h}\right) + \mathcal{O}\left(N_{UE,sche}^2\ \frac{N_C}{h}\right)\right] + \mathcal{O}(N_{UE,sch}\ \ N_C^2) = \mathcal{O}(N_{UE,sched}^2 N_C + N_{UE,sched}N_C^2)$. Notably, the MHSA complexity does not depend on the number of attention heads h.

- The last pair of FC layers generates one feature from a vector of length N_C for each user, resulting in a computational complexity $\mathcal{O}(N_{UE,sche}\ N_C)$.

Thus, the overall computational complexity with the DMHSA is:

$$\mathcal{O}_{DMHSA} = \mathcal{O}(\delta N_{UE,sched}N_C) + \mathcal{O}(N_{UE,sche}^2\ N_C + N_{UE,sche}\ N_C^2). \quad (44)$$

In case of CSI-based beamforming, $\delta^{(CSI)} = N_R + 2$; thus, the computational complexity becomes $\mathcal{O}(N_{UE,sched}N_C N_R) + \mathcal{O}(N_{UE,sched}^2 N_C + N_{UE,sched}N_C^2)$. Considering that $N_{UE,sched} \ll N_R$, the complexity reduces to $\mathcal{O}(N_{UE,sched}N_C N_R + N_{UE,sched}N_C^2)$. Furthermore, the choice of $N_C = 8$ makes so that $N_C \ll N_R$; thus, the overall computational complexity for SINR estimation with the CSI-DMHSA model is:

$$\mathcal{O}_{DMHSA}^{(CSI)} = \mathcal{O}(N_{UE,sched}N_C N_R), \quad (45)$$

which, given $N_C < N_{UE,sched}$ (typically $N_{UE,sched} = N_B = 24$), implies that the complexity is effectively reduced with respect to the evaluation through the computation of the MMSE beamforming equation.

In case of location-based beamforming, $\delta^{(GEO)} = 3$; thus, the computational complexity becomes $\mathcal{O}(N_{UE,sched}^2 N_C + N_{UE,sched}N_C^2)$. With the same consideration on N_C as above, the complexity of the GEO-DMHSA model reduces to:

$$\mathcal{O}_{DMHSA}^{(GEO)} = \mathcal{O}(N_{UE,sched}^2 N_C), \quad (46)$$

providing a great computational complexity reduction for SINR estimation. The results of the computational complexity assessment are summarized in Table 17.

Table 17: Computational complexity for SINR estimation.

SINR estimation method	Computational complexity
MMSE-based	$\mathcal{O}_{SINR-MMSE} = \mathcal{O}(N_{UE,sched}^2 N_R)$
DMHSA CSI-based	$\mathcal{O}_{DMHSA}^{(CSI)} = \mathcal{O}(N_{UE,sched}N_C N_R)$
DMHSA location-based	$\mathcal{O}_{DMHSA}^{(GEO)} = \mathcal{O}(N_{UE,sched}^2 N_C)$

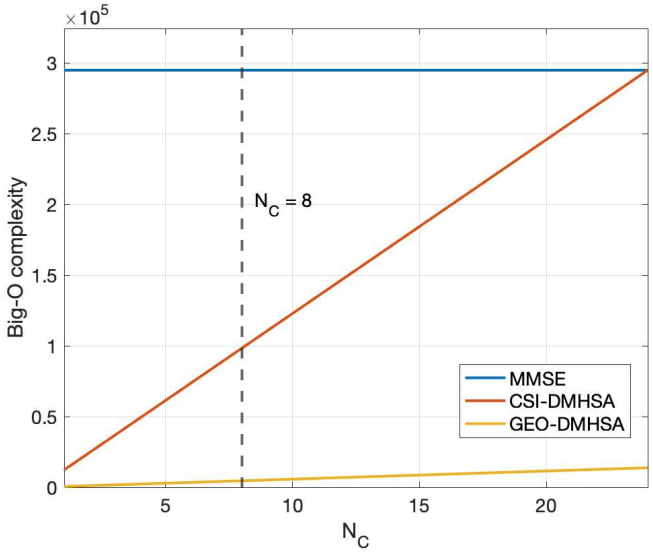


Figure 42: Computational complexity as a function of N_C.

Figure 42 reports the Big-O computational complexity achieved by the two DMHSA models (CSI-based and location-based in orange and yellow, respectively) and the MMSE baseline (blue line) as a function of the number of channels N_C, *i.e.*, the size of the embeddings of each user's features vector; the grey dotted line represents the value set in the trained models, *i.e.*, $N_C = 8$. The plot shows that the reduced features vector in the GEO-DMHSA model ensures that the complexity remains orders of magnitude lower than that of MMSE-based SINR estimation: at $N_C = 8$, the location-based algorithm achieves a complexity of $\approx 4.6 \cdot 10^3$ against the $\approx 2.9 \cdot 10^5$ required by the benchmark. The CSI-DMHSA model also ensures a reduction in complexity to $\approx 9.8 \cdot 10^4$; however, the large features vector makes so the complexity in the CSI-based case scales much faster than in the location-based case with respect to N_C: while longer embeddings could improve the SINR estimation accuracy, the increased complexity may disfavour the application of the CSI-DMHSA method. This is not the case with GEO-DMHSA, where even with embeddings of 24 elements the complexity is at one order of magnitude under the MMSE-based SINR estimation. It should also be mentioned that the CSI-DMHSA and GEO-DMHSA models require 4.8k and 762 learnable parameters, respectively, making them extremely compact.

4.3.5 Model training

Two separate models with the reported DMHSA architecture have been trained using CSI reports as input data for one of them and location reports for the other. The models have been trained by simulating scheduling instances at random samples of a satellite pass, using the same parameters reported in Table 8. At each instance i, $N_{UE,sched,i} \sim U(8, N_B)$ users are randomly selected for scheduling, their CSI vectors (or location vectors) are pre-processed and collected as training input data sample $\mathbf{F}$ (Section 4.3.3), and their SINR is assessed by means of the MMSE equation and stored as label (*i.e.*, true SINR value) for the corresponding input data sample. Once the batch is filled with $N_{batc} = 8192$ samples, the input data $\mathbf{X}^{(DMHSA)} \in \mathbb{R}^{N_B \times \delta \times N_{batc}}$ and the corresponding padding masks $\mathbf{M}^{(DMHSA)} \in \mathbb{R}^{N_B \times N_{batch}}$ are fed to the model, and the corresponding SINR estimates $\widehat{\mathbf{Y}}^{(DMHSA)} \in \mathbb{R}^{N_B \times N_{batch}}$ are obtained; then, the Mean Squared Error (MSE) loss function is evaluated between the SINR estimates at the output of the model and the corresponding ground truth labels $\mathbf{SINR} \in \mathbb{R}^{1 \times N_{beams} \times N_{batch}}$:

$$MSE = \frac{\sum_{i=1}^{N_{batc}} \sum_{u=1}^{N_B} \mathbf{M}_{u,i}^{(DMHSA)} \cdot \left(\widehat{\mathbf{Y}}_{u,i}^{(DMHSA)} - \mathbf{SINR}_{u,i}\right)^2}{\sum_{i=1}^{N_{batch}} \sum_{u=1}^{N_B} \mathbf{M}_{u,i}^{(DMHSA)}}. \quad (47)$$

Table 18: Simulation parameters for DMHSA training.

Parameter	Value	Comments
Sub-Carrier Spacing (SCS)	120 kHz	In this case, 132 PRBs are available
Operating Frequency Range	FR2 (Ka-band)	See 3GPP TR 38.821
Carrier frequency	20 GHz	See 3GPP TR 38.821
User bandwidth B	190.08 MHz	$132 \cdot 120 \cdot 12$ kHz
Scenario	Digital Divide	See Section 4.2.1.2 in [2]
Number of radiating elements $N_{R,sat}$	512	See Section 2.3.1 of [41]
Power per radiating element $P_{av,el}$	65 mW	See Section 2.3.1 of [41]
Total power on-board P_{av}	15.2218 dBW	$10\log_{10}(0.0065 \cdot 512)$
Minimum user elevation angle ε_t	30°	See Section 4.2.3 of [41]
Service area elevation angle ε_S	30°	See Section 4.2.3 of [41]
Channel type	clear-sky	See Section 2.4.2.1 of [41] and 3GPP TR 38.811
NTN node altitude	1000 km	See Section 2.1 of [41]
Service area centre	(45°N, 5°E)	See Section 4.2.3 of [41]
UE terminal	VSAT	See Section 2.3.2 of [41] and 3GPP TR 38.811

Through the backpropagation algorithm and the Adam optimizer, [53], the trainable weights of the model are updated, and a new batch of data starts being generated. To improve the training process, the following techniques have been employed:

- **L2 regularization**. This technique ensures that trainable weights do not diverge too much from 0, thus reducing overfitting. This is achieved by introducing a loss function penalization factor equal to the sum of the squared weights scaled down by a constant ϵ_R. Although overfitting is not an issue *per se* due to the continuous generation of data, it was observed that the model would still benefit from a low ϵ_R value rather than not implementing L2 regularization at all.

- **Learning rate warm-up**. Typical learning rate schedules foresee starting training with a large learning rate to quickly move from the initialization state towards a local minimum of the loss function. However, it has been observed that sometimes the training process benefits from starting with a very low learning rate λ_{min} and linearly ramping up to a large learning rate λ_{max} in a few training epochs T_w. This technique, called "learning rate warm-up", often helps in starting the gradient descent in a better direction: the general intuition behind this is that the direction for gradient descent may not be optimal at the beginning of the training process due to random weight initialization, thus leading to a great descent in a suboptimal direction when a large learning rate is implemented. On the opposite, a low starting learning rate can help in gaining momentum towards a more optimal direction over multiple training steps or epochs.

- **Learning rate cosine annealing with warm restarts**. This schedule lets the learning rate oscillate from a maximum value λ_{max} to a minimum value λ_{min} over the span of T_c epochs following a cosine trend as in the following equation, [54]:

$$\lambda(i) = \lambda_{min} + \frac{1}{2}(\lambda_{max} - \lambda_{min})\left(1 + cos\left(\frac{T_{cur}}{T_c}\pi\right)\right),$$

where $\lambda(i)$ represents the value of the learning rate at the i-th training epoch and T_{cur} represents the current number of training epochs since the last warm restart. Conversely from cold restarts, once $T_{cur} = T_c$ the trainable weights are not reinitialized to small random values, possibly leading to other nearby local minima being reached during the following cycle of T_c epochs.

- **Early stopping**. Typically, training is interrupted after T_s epochs without a loss function improvement, with the model providing the best loss function value being restored. Given the cosine annealing schedule, which tends to provide loss function improvements at the end of a cycle (*i.e.*, when low learning rates are being used), we set T_s in terms of number of completed full cosine annealing cycles since the lowest loss function value was achieved.

Table 19 reports the parameters employed to train both of the DMHSA models. As a new batch of data is generated at each epoch and samples contained in previous batches are not reused in the following epochs, the models cannot overfit to the dataset: for this reason, a validation set is not generated.

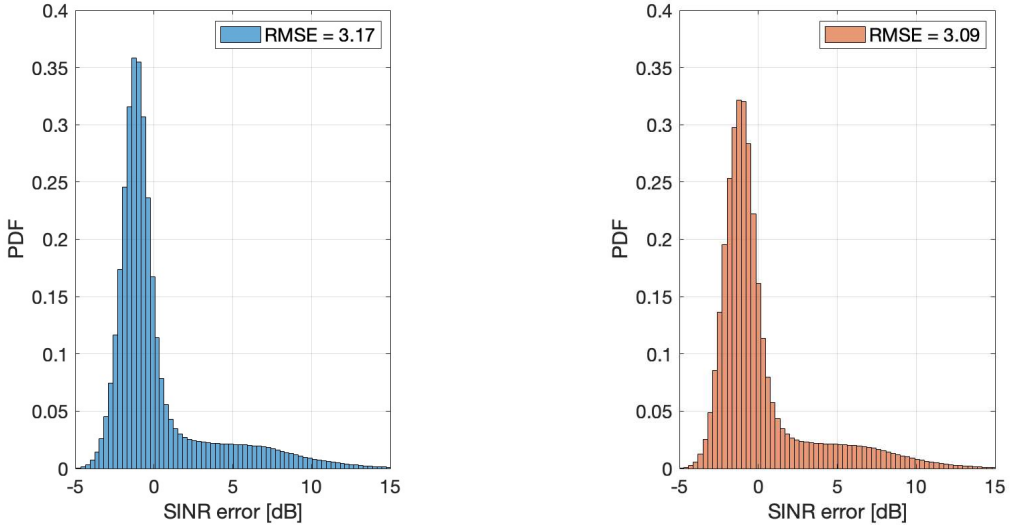
Table 19: Training parameters for DMHSA.

Training parameter	Value
Training batch size	$N_{batch} = 8192$
Maximum number of training epochs	15000 epochs
L2 regularization factor	$\epsilon_R = 10^{-6}$
Learning rate warm-up period	$T_w = 40$ epochs
Learning rate cosine annealing period	$T_c = 100$ epochs
Early stopping patience	$T_s = 4$ full cosine annealing cycles
Minimum learning rate	$\lambda_{min} = 10^{-4}$
Maximum learning rate	$\lambda_{min} = 5 \cdot 10^{-3}$

4.3.6 Performance evaluation

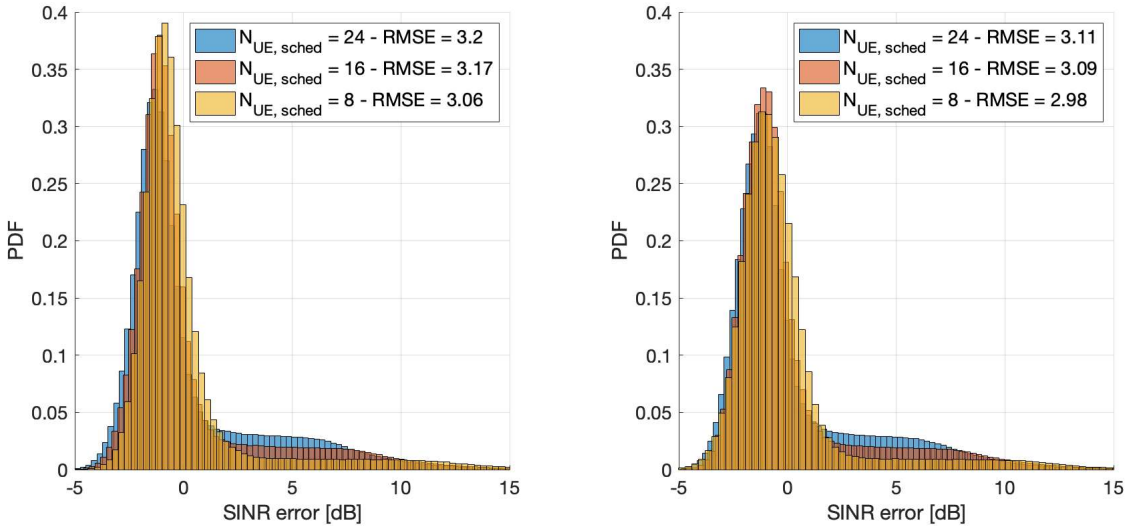
4.3.6.1 DMHSA with random scheduler and variable number of users

The objective of the first evaluation is to determine the SINR estimation error under general conditions: *e.g.*, training on randomly scheduled users ensures that the models observe both cases of optimal scheduling and ill-conditioned beamforming matrices. For this reason, the models were first evaluated on a test set of $2.56 \cdot 10^6$ test samples under the same conditions of the training set, *i.e.*, each sample contains a random number of randomly scheduled users' reports obtained in random instants of the satellite pass, for a total of $\approx 4 \cdot 10^7$ SINR estimates. As for the training data, the simulation parameters used to generate test data are the same reported in Table 18.



a) CSI-DMHSA *b) GEO-DMHSA*

Figure 43: SINR estimation error distribution with random scheduler.



a) CSI-DMHSA *b) GEO-DMHSA*

Figure 44: SINR estimation error distribution with random scheduler for different $N_{UE,sc}$ values.

Figure 43 reports the histogram of the error distribution for both CSI-DMHSA and GEO-DMHSA, computed as $\mathbf{E} = \hat{\mathbf{Y}}_{\mathrm{DMHSA}} - \mathbf{SINR}$, with Probability Distribution Function (PDF) normalization. The distribution is skewed in both cases, suggesting that the models cannot provide accurate estimates in certain conditions: this is the case of ill-conditioned beamforming matrices resulting from poor user scheduling (*i.e.*, the random scheduler often selects users

that are geographically close together, resulting in highly correlated CSI vectors). This consideration is made clearer with Figure 44, where the same assessment is carried out by separating the test datasets based on the number of scheduled users. As expected, for both models, the RMSE increases with the number of users scheduled, and so does the frequency of high-error estimations. Nonetheless, the lowest-complexity GEO-DMHSA model also achieves a lower SINR estimation RMSE with respect to the CSI-DMHSA model. While the error tails reach high SINR values, it should be noted that they are caused by poor scheduling choices made by the random scheduler; thus, the accuracy of the models should be evaluated considering a proper scheduling technique.

4.3.6.2 DMHSA with PQS scheduler

In the second evaluation step, the PQS scheduler presented in [2] is implemented. This corresponds to a more realistic case in which a scheduling algorithm selects groups of users to be served during the same time slot for which the SINR should be assessed (*e.g.*, to determine the expected rate for each user, or to evaluate the sum-rate achievable by different groups in order to select which group to schedule). The PQS scheduler parameters are the same as the one reported in Table 9, while the simulation parameters are reported in Table 18 and Table 20; the CSI and GEO scheduling constraint threshold was selected from the values reported in Table 17 and Table 18 of [2], respectively, based on the minimum and maximum requested capacity C_{min} and C_{max}. It should be noted that the models are assumed to be deployed as is, *i.e.*, without retraining on user reports with PQS scheduling. For this reason, due to the different SINR statistics with respect to the training dataset, the obtained estimator is biased; such bias is assessed and subtracted by the models' output.

Table 20: Additional parameters for DMHSA testing with PQS scheduling.

Parameter	Value	Comments
Minimum capacity request, C_{min}	5,20 Mbps	See Section 4.2.1.2 in [2]
Maximum capacity request, C_{max}	100,500 Mbps	See Section 4.2.1.2 in [2]

Figure 45 reports the CDFs of the absolute error $\mathbf{E}_{abs} = \left|\hat{\mathbf{Y}}_{\mathrm{DMHSA}} - \mathbf{SINR}\right|$ achieved by the CSI-DMHSA and GEO-DMHSA models under the minimum and maximum capacity requests C_{min} and C_{max}, respectively, reported in the legend as "C_{min}/C_{max}". The plot shows that both models are able to achieve a median absolute error (*i.e.*, corresponding to the 50% CDF) under 0.7 dB for all cases. Overall, the performance of the DMHSA models is similar, with the location-based model providing slightly improved high percentiles and the CSI-based model excelling in the lower ones (except in the 5/100 case, where the opposite is true). The plot also shows that the cases with the largest C_{min} are the ones providing the best absolute error: as reported in Section 4.3.1.4 of [2], a low C_{min} value leads the PQS scheduler tends to schedule less users than the number of beams, thus resulting in consistently high SINR values, a case that was not well represented in the training data considering the implementation of a random scheduler; on the opposite, $C_{min} = 20$ Mbps results in a better aligned SINR distribution. Similarly, an increase of C_{max} from 100 Mbps to 500 Mbps with $C_{min} = 5$ Mbps leads to a significant improvement of the absolute error due to the more stringent constraint on the CSI vectors correlation and the inter-user distance.

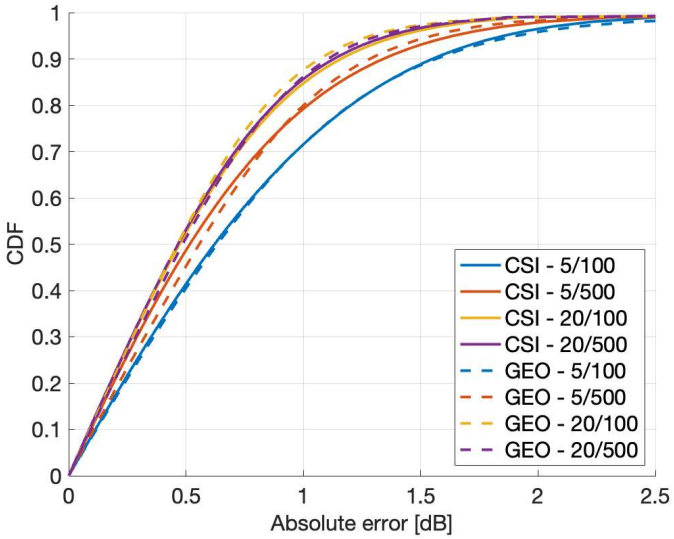


Figure 45: CDF of the SINR estimation absolute error with PQS scheduling.

Table 21: SINR estimation RMSE with PQS scheduling.

Scheduler configuration	C_{min}	CSI-DMHSA		GEO-DMHSA	
		C_{max}			
		100 Mbps	500 Mbps	100 Mbps	500 Mbps
$\sigma_{vis} = 50$ ms $\sigma_{\text{cap}} = 2$	5 Mbps	0.96 dB	0.84 dB	1.18 dB	0.90 dB
	20 Mbps	0.71 dB	0.70 dB	0.68 dB	0.70 dB

Table 21 reports the SINR estimation RMSE achieved by the DMHSA models under the considered capacity requests. Notably, the RMSE remains under the 1dB threshold in most cases (except for the 5/100 case with GEO-DMHSA). The two models perform similarly at high minimum capacity request; on the opposite, the CSI-DMHSA model has an advantage over its location-based counterpart at low minimum capacity request. Clearly, the observations reported for Figure 45 are also reflected in the RMSE.

5 ONBOARD PROCESSING

The development of 5G, and long-term 6G, NTN systems makes on-board processing with fully regenerative payload a major trend that tends to be generalized, especially for telecom constellation systems.

When it comes to design processing capabilities, two sides shall be jointly considered:

- Pros and opportunities brought by processing: performances (e.g. gain on signal-to-noise ratio (SNR), system capacity, etc.) and provided features (internal system features that help system design, or value-added service directly perceived by users and customers)

- Constraints and impacts: such as requirement related to processing, CPU, memory, power consumption, implementation, mass and sizing impact, resistance to radiations, and maybe thermal impacts. From these perspectives, any processing resources are always sized at the right level in operational systems and processing margins are sized to be limited for this reason. At the same time, in system with 5-10 years of satellite lifetime, the exact need in terms of telecom service may be hard to anticipate and size, particularly when thinking to the generalisation of software processing related to the processing of telecom services. Ideally, expectations from operation and service providers would be to integrate satellite constellations in Cloud infrastructures with the same level of flexibility and number of resources as found in ground.

Such trade-off is hard to achieve, and generally speaking, the exact mission and system objectives will allow to determine what are the limits to consider - hence on a per system basis. In short-term application, and first generation of 5G constellation system, the target cost to consider for wide-scale constellation is in Billions of dollars. System offered capacity (total Gbps) and achieved bit rate per users (peak/average) are among the first KPIs of interest. The offered Gbps capacity per satellite can also be derived. It means that this amount shall be:

- Supported in every external communication interface
    - Feeder link (satellite-to-GW), possibly with multiple GWs at the same time
    - On ISL, if the system is equipped
    - On the user link. This represents on the user link ~50 to 500 MHz of spectrum band to process, and frequency reuse may apply among different spot beams raising this even higher.

- Supported by the OBP router / switch (sum of all data rates)

- Supported on all internal communication interfaces (for example between radio access layer and infrastructure-level network layer)

Other drivers are:

- The development of 5G/6G NTN standard remains without any large-scale deployment today, letting aside non-standard 4G based solutions, and/or in-orbit experimentations. The 5G/6G found its origin in terrestrial networks where the resources are “unlimited”. Do the requirements and 3GPP specifications (at least in short-term) will actually be feasible in operational system where satellites are used as (partial) 5G base stations? For processing requirements, the short-term critical questions relate to:

- The question of high frequency update (more CPU) demanded by 3GPP process (user plane, control plane) and by the supporting means such as active antenna for near real-time beamforming
- The memory requirement can be significant (upon the exact role of the satellite and split of on-board function; could be several Gbytes to tens or hundreds of Gbytes)

- Application on the virtualization concepts and technologies on the execution platforms:
  - Strong incentives to support and segregate distinct missions, and/or services from distinct operators; they can manage services on their own, on the other hand in a context of limited resource it is difficult to ensure an optimal allocation in particular taking into account variations (*dynamic and elastic partitioning*) with generally some pre-empting usages.

- Achieved QoS and QoE:
  - In addition to data rates 5G/6G users will expect low and stable end-to-end delay, while access to the majority of 5G services known in TN.
  - Continuity of service: transparent handover, with no/few packet losses.

Last, impact on 3GPP standard may also have to be considered. For example, some implementation feasibility could lead to introduce features in the scope of desired Release. Even if deviations can always be implemented on a per system basis, they should be minimized to guarantee the long-term support of the feature.

BEAM MANAGEMENT

When it comes to digital processing on-board topic, the antenna beamforming strategy is among first topic to consider for the antenna beam laws computation and application. At system level, the question of Earth moving beams vs Earth fixed beams require first a trade-off analysis. Beam size determination can also be jointly considered, if not already known in customer specifications.

Table 22: Earth moving beams vs Earth fixed beams comparison

	PROS	CONS
Earth moving cell	Simpler satellite beam management (for antenna and its Beam Forming Network) & Radio Parameters configuration	**QoS and Performance:** • Frequent radio HOs. • Impact on UE Throughput (overhead due to heavy HO signalling)
Earth-fixed cell	**Smooth impact on 5G 3GPP standard and against existing Terrestrial Network** **QoS and Performance:** • Less Frequent radio HOs. • Less Impact on UE Throughput	More complex satellite beam generation. Potentially requires additional change/adaptation with respect to radio parameters configuration.

More in detail:

- From QoS and performance perspective, moving cells may suffer from recurrent handovers (HOs): In these cells, the signalling overhead due to HOs may range from tens of kbps at the nadir to hundreds of kbps at satellite coverage border.

- For earth fixed cells the HOs happen in bursts. The peak in number of handovers can be lowered by extending the overlapping period (this is implementation dependent, e.g. constellation design).

- Due to the asynchronous nature of the NR Release 15 handover functionality with random access at every cell change, there is an undesirable temporary data interruption gap at every handover. This is impacting mainly the moving cell, the connected UE is continuously handed over between cells. This may degrade the end-user QoE.

- Another main KPI to be considered when comparing fixed and moving cells is the handover success rate. This KPI is likely to be degraded on moving cell due to fast mobility and high volume of Handovers (more than 250000 HOs/h in case of moving cell of 60km size). Furthermore, in case of a handover failure, the UE cannot return to old cell (due to the motion of the source moving cell) which means the call will be simply dropped. Because the moving cell is suffering largely from the fast mobility and high volume of handovers to be handled, we can expect that: the overall HO success rate for the moving cells will be lower compared to fixed cells, while the call drop rate will be greater compared to fixed cells.

- When it comes to tracking area management, for Earth fixed cell deployment: Earth Fixed timing advance is used. Thus, no enhancement will be needed in the standard. For moving cell: Earth Fixed timing advance or moving timing advance can be used. This will need some modification in the 3GPP standard.

Overall, the earth fixed cells are the preferred solution with respect to QoS and performance and smooth impact of 3GPP standard.

5.1 PROCESSING AND STORAGE CAPABILITIES OF SATELLITES

In recent years, satellite technologies have experienced rapid evolution, not only in terms of payload and communication capabilities but also with respect to onboard processing and storage. Traditionally, satellites have relied on ground stations for most data processing tasks. However, as demands for latency-sensitive services, data autonomy, and efficient bandwidth usage grow, satellites are increasingly equipped with greater onboard computing and memory resources.

Onboard processing allows satellites to make real-time decisions, optimize communication resources, and support intelligent applications such as beamforming, routing, or even AI inference. Onboard storage complements this by allowing satellites to buffer and prioritize data, compress and encrypt content before downlink, or cache information for intermittent contact periods. Nevertheless, satellite computing systems face strict constraints such as limited power supply, exposure to radiation, heat dissipation issues, and the need for highly reliable and fault-tolerant hardware.

This section categorizes satellites into three types:

(1) traditional telecommunication satellites
(2) hybrid satellites supporting telecommunication and edge computing,
(3) satellites primarily designed for edge computing.

For each, we explore current state-of-the-art CPU, RAM, and storage configurations, supported with real-world examples.

5.1.1 Traditional telecommunication satellites

Traditional telecommunication satellites are of Geostationary class orbit, serving large area (e.g. regions of multiple countries) and potentially many users. These satellites are primarily designed to relay data between users and ground stations, with low or no onboard processing acting on the user signal. Their main function is to amplify and transmit signals, often using fully transparent bent-pipe architectures with fixed digital signal processing chains. A few amounts of those satellites can implement so-called regenerative payload with on-board processing acting on the signal. Those satellites can address specific needs: on-board modem functions (achieving better link budgets); on-board switching (support of flexible mesh connectivity at packet-level); etc.

- **CPU:** These satellites typically use radiation-hardened processors such as LEON3/LEON4 (SPARC-based), or earlier RISC designs. Clock speeds range from 50 MHz to 250 MHz. These processors are designed for reliability rather than performance, and can be used as HW solution for running the SW tasks required for the satellite platform management. Note that instead CPU processor, FPGA or ASIC processing platforms can also be used if modem and/or switching capabilities only are needed.

- **RAM:** RAM capacities range between 256 MB and 1 GB of SDRAM or DDR2 memory. Error-correcting code (ECC) is standard to mitigate the risk of radiation-induced bit flips.

- **Storage:** Onboard storage is limited, generally ranging from 4 GB to 64 GB of NAND flash or similar radiation-tolerant memory. Storage is used for basic buffering and telemetry.

- **Example:** The Amazonas-3 GEO satellite operated by Hispasat includes fixed transponders with minimal onboard processing. Viasat-2, another GEO satellite, supports 300 Gbps throughput using fixed frequency planning and limited onboard computing [55].

5.1.2 Hybrid telecommunication and edge computing satellites

This new generation of satellites supports traditional telecommunication functions while also embedding edge computing features to process data locally. These satellites employ digital payloads and software-defined transponders capable of dynamic reconfiguration.

- **CPU:** These systems use more capable SoCs, such as ARM Cortex-A53 or A72 (64-bit), sometimes in multi-core configurations. Clock speeds range from 500 MHz to 1.5 GHz. Some platforms use FPGAs or custom ASICs for specific AI or signal processing tasks such as in Software Defined Radios solution (SDR).

- **RAM:** Capacities vary between 2 GB and 16 GB of DDR3/DDR4 or LPDDR4 memory, with ECC and radiation shielding. This maximum value is expected to scale by factor 1-10 in upcoming years (2030).

- **Storage:** Onboard storage can reach 128 GB to 2 TB, typically implemented using radiation-tolerant NAND flash. Storage is used for buffering, caching, local data analysis, and redundancy.

- **Example:** OneWeb's Gen-1 LEO satellites use advanced digital payloads and perform some local data routing and beam steering [56]. SES-17, a GEO HTS (High-Throughput

Satellite), includes Thales' SpaceFlex digital processor, allowing in-orbit reconfigurability and resource allocation [57].

5.1.3 Edge computing-first satellites

Designed specifically for onboard intelligence and autonomous decision-making, these satellites support real-time analytics, AI inference, and data fusion. They are typically deployed for Earth observation, defence, or advanced IoT use cases.

- **CPU:** Edge-first satellites often incorporate heterogeneous computing architectures combining ARM CPUs, GPUs, and FPGAs. Platforms like NVIDIA Jetson (TX2, Xavier) or Xilinx Zynq Ultrascale+ are increasingly used in experimental missions. Clock speeds can reach 2 GHz, with multi-core processors.
- **RAM:** These platforms support 8 GB to 32 GB of high-speed LPDDR4 or GDDR memory, necessary for running AI models and handling large sensor inputs.
- **Storage:** Onboard memory may range from 512 GB to 4 TB, especially in Earth observation or optical relay satellites where high-resolution imagery is processed before transmission.
- **Example:** ESA's OPS-SAT mission is a testbed for AI and edge computing in orbit, using a Xilinx Zynq-7000 SoC with 512 MB RAM and 24 GB storage [58]. Starlink V2 Mini satellites reportedly feature custom SoCs and high-capacity flash storage to support inter-satellite laser links and intelligent routing [59].

Satellite processing and storage capabilities are undergoing a paradigm shift, moving from simple signal relay systems to intelligent, flexible platforms that support complex in-orbit operations. Traditional telecommunication satellites still dominate in number, but hybrid and edge-computing-first satellites are gaining prominence as the industry pivots toward autonomy, reduced latency, and more dynamic mission profiles.

Table 23 summarizes the typical CPU, RAM, and storage specifications across the three satellite categories discussed.

Table 23: Typical CPU, RAM, and Storage Specifications

Category	CPU	RAM	Storage
Traditional Telecom Satellites	50 – 250 MHz LEON3/LEON4	256 MB – 1 GB	4 GB – 64 GB
Hybrid Telecom + Edge Computing Satellites	500 – 1500 MHz ARM Cortex-A53/A72	2 GB – 16 GB	128 GB – 2 TB
Edge Computing-First Satellites	2000 MHz Jetson TX2/Xavier, Zynq SoC	8 GB – 32 GB	512 GB – 4 TB

These improvements in onboard processing and storage capabilities not only support better communication efficiency but also open up new applications in real-time analytics, AI, and autonomous space systems. As launch costs decrease and satellite hardware becomes more modular, we can expect a growing shift toward intelligent constellations operating closer to the edge of data generation.

Looking ahead, future generations of satellites are expected to feature even more advanced processing and storage capabilities, driven by trends in miniaturization, AI integration, and modular satellite platforms. Some of the projected characteristics include:

Table 24: Projected characteristics in terms of CPU, RAM and Storage of next generation satellites

Future Category	Expected CPU	Expected RAM	Expected Storage
Next-Gen Hybrid Satellites	> 2 GHz Multi-core RISC-V, ARM Neoverse	16 GB – 64 GB	1 TB – 8 TB
AI-Dedicated Edge Satellites	AI accelerators, GPUs (e.g., Jetson Orin, EPYC)	32 GB – 128 GB	4 TB – 16 TB
Quantum/Swarm Satellites (R&D)	Custom ASICs, optical/radio photonic chips	TBD (high speed cache)	TBD

These advances will enable satellites to handle petabyte-scale sensing, perform autonomous AI model training and inference, and conduct adaptive mission planning without constant ground intervention. As regulatory and commercial ecosystems evolve, future constellations will likely operate as a distributed computing layer in low Earth orbit, forming the backbone of space-based edge intelligence.

5.2 ON-BOARD ARCHITECTURES

In the following we consider 3 incremental on-board architectures, each requiring more demanding resources, but also offering more features. The analysis conducted in this section complements the descriptions provided in in Sections 2.2, 2.3, 2.4 from the OBP standpoint.

We could associate:

- CU/DU splitting (split 2) more relevant for the short-term application (current constellation design)
- Full gNB on-board to mid-term (3-5 next years)
- Full gNB + user plane function (UPF) for even long-term (>5 years)

5.2.1 CU/DU splitting

The first option is to consider gNB CU/DU split, and according to the on-board and ground architecture presented in Figure 46.

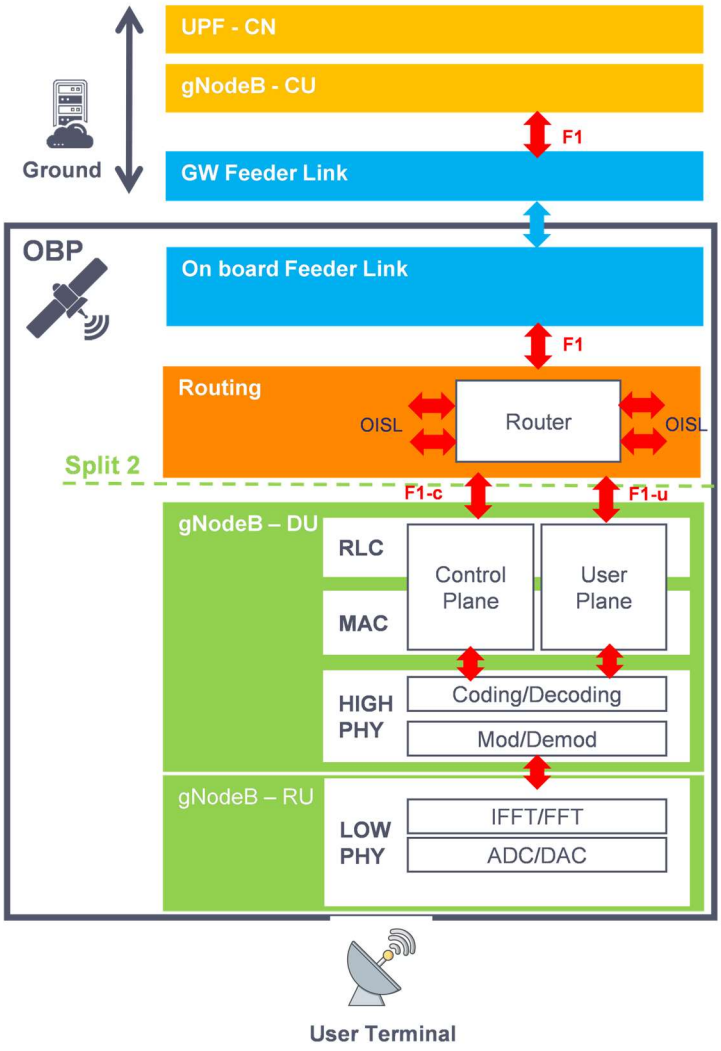


Figure 46: Payload and ground architecture - split 2 (DU/RU on board, CU on ground)

The physical layer is responsible for coding, modulation, beamforming, and mapping of the signal to the appropriate physical time-frequency resources. It provides services to the MAC layer in the form of transport channels and the overall transport channel processing is mostly the same in UL and DL.

A cyclic redundancy check (CRC) for error-detecting purposes is added to each transport block, followed by error-correcting coding using low density parity check (LDPC) codes. Rate matching adapts the number of coded bits to the scheduled resources, the code bits are scrambled and fed to a modulator, and finally the symbols are mapped to the physical resources, including the spatial domain. For the UL there is also a possibility of an FFT-precoding to reduce the peak-to-average-power ratio (PAPR) of the signal at the amplifier input.

One key aspect of this processing chain is the multi-antenna precoding, i.e. beamforming. The purpose of the precoding is to map the different transmission layers to a set of antenna ports using a precoder matrix to provide directivity and focus the overall transmitted power in a certain direction. In the context of a digital beam forming network, this operation may present a challenge in terms of implementation complexity, especially in a fixed beam scenario where the beamforming laws must be updated frequently.

At access layer, one important function to perform is the real-time L2 scheduling. Implementation of the scheduler shall be carefully investigated: firstly, its performance is detrimental to delivered QoS, and shall be accommodated with the relevant transmission channels (FR1 or FR2, according to the type of satellite channel). The typical dynamicity (few ms) to consider is different on the satellite (even LEO) case than for terrestrial case and could be relaxed on a ~10 ms basis, or more with Gaussian channels (FR2 case in particular). A positive impact for alleviating processing, if a starting implementation was used from a TN software stack would be expected.

5.2.2 Full gNB on-board

In a second architecture, the full gNB is envisaged on-board (see Figure 47).

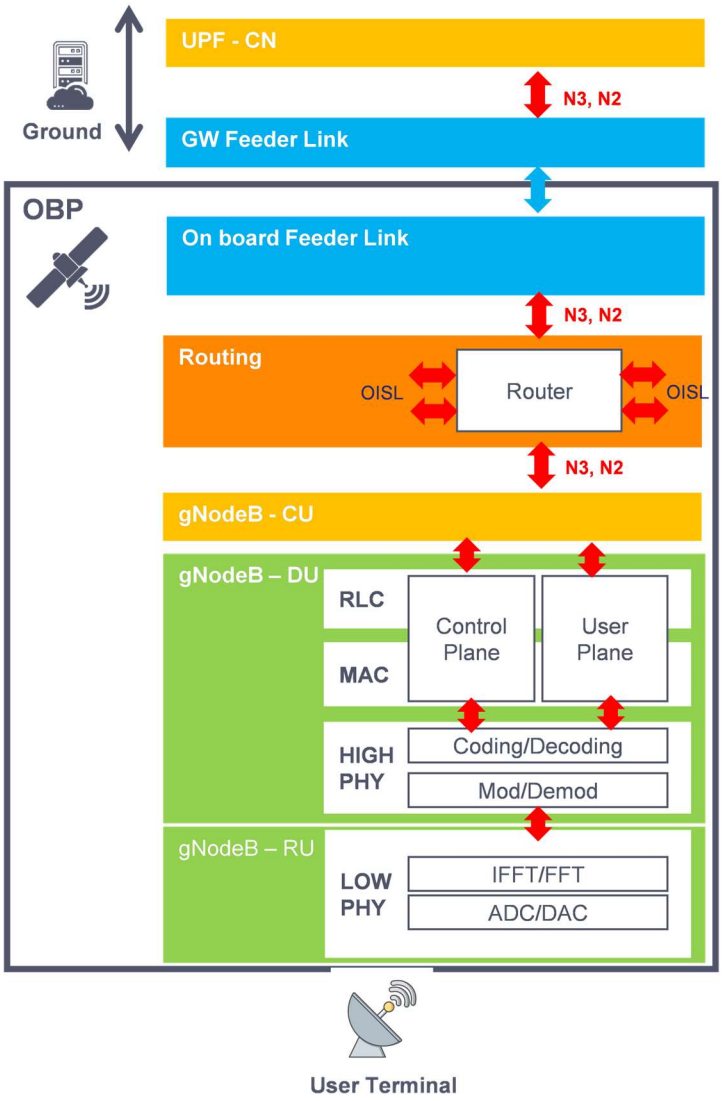


Figure 47: Payload and ground architecture - gNB on board and UPF/CN on ground

Note that split can have various impact on system levels implication, but this analysis exceeds the scope of this discussion.

The full gNB implementation will add processing and network load to consider, in particular for the short-term.

Architecture with full gNB on-board makes easy the implementation of RAN-Core interconnection conceptually, while validation of communication interface shall still be carefully investigated in the context of NTN. Industrially speaking, it is also of interest. However, no application layer nor caching functions can be implemented on board in this architecture as the N3 interface tunnels the traffic between the gNB and the UPF. Full gNB can be an intermediary step to the last architecture.

5.2.3 Full gNB on-board + UPF

The third architecture envisages the full gNB on-board with UPF functions (see Figure 48).

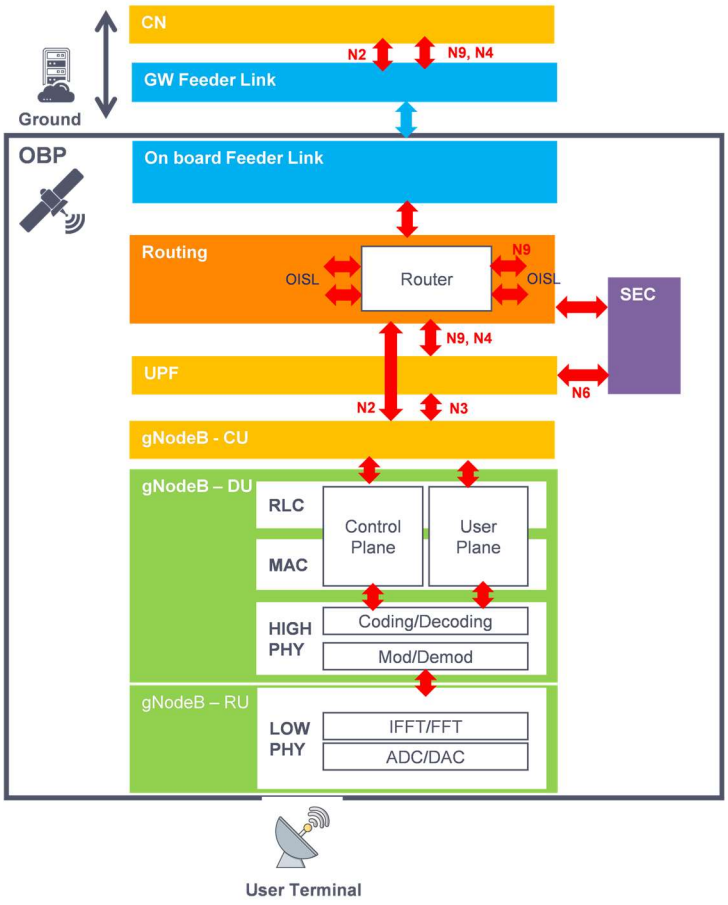


Figure 48: Payload and ground architecture - gNB + UPF on board and core network on ground

This third architecture is compatible with mesh connectivity through the N9 interface at the OISLs. It can provide the ability for a user terminal to reach back without the need for anchor relay. The consideration of limited power and computing resources further highlight the need for efficient processing of traffic at gNB + UPF side, and algorithms that optimize resource usage while minimizing energy consumption. Potentially only a subset of the traffic could be fully processed on-board enabling SEC and Mesh only for this part of the traffic.

This architecture is also compatible with SEC as the N6 interface of the UPF enables accessing to the PDU layer which makes it then compatible with application layer and caching functions.

SEC[3] refers to the deployment of computational resources and data processing capabilities closer to the data source -in this case, on satellites payload. This approach aims to reduce latency, optimize bandwidth usage, and enhance the efficiency of satellite communications by processing data at the "edge" of the network rather than transmitting it back to centralized data centres on Earth.

However, to provide these functions the satellite needs to be equipped with additional OBP units, such as CPUs, GPUs, or specialized accelerators, or mass storage. These nodes can perform data analysis, filtering, caching and decision-making tasks in orbit.

This is one of the main challenges of SEC as satellite have limited power resources, size and mass, and the processing of the traffic is already demanding which may leave few processing units for other tasks or value-added application. It's a trade-off (economical and technical) that should be done between the processed traffic and the capacity to perform application layer or caching function that can provide added value.

In some use cases processing data locally on the satellite can help that only relevant or summarized information is transmitted back to Earth, to significantly reduce the volume of data that needs to be sent over potentially limited bandwidth links. Edge computing can also minimize the time it takes to process and respond to application layer request, as it avoids the delays associated with transmitting data to and from ground stations.

However, one of the big challenges in SEC for LEO constellations is the fact that the satellite is moving and then the edge for the UE is frequently changing, making needs for SEC HO and context exchanges.

Further, the SEC approach will be a concrete case to instantiate concepts of payload (or OBP) resources. OS virtualization (Hypervisor type 1 or 2) or container-based approach (Linux Docker) are very popular solutions. Not speaking about the services (e.g., data analysis, filtering, caching and decision-making, etc.) the top-level implementation issues to consider in the design choices include:

- How the computing resource arbitration is managed among the service (soft / elastic partitioning)? or hard partitioning? is the ground involved in the decision process, what is the dynamicity to be expected?

- What about segregation / isolation constraints between the OS/containers from the Cyber-security perspectives? is the qualification level of cyber-security of the hypervisor relevant to the specific services and missions envisaged?

5.3 RESOURCE CONSUMPTION MEASUREMENT TOOLS AND METHODOLOGIES

As satellite systems become increasingly software-defined and compute-capable, understanding the resource consumption of applications and processes is vital for efficient system design and mission assurance. Accurately measuring CPU, memory (RAM), and storage utilization helps engineers assess application feasibility, optimize performance, and detect bottlenecks. In this section, we review state-of-the-art tools and methodologies for

[3] Space edge computing is also addressed as part of T5.5 in WP5, in the context of 'Onboard networking capabilities' and later documented in deliverable D5.5 due at M30 of the project.

profiling application-level resource usage, with a specific focus on their integration within our 5G NTN testbed.

5.3.1 Tools for measuring resource consumption

Various tools exist to monitor the usage of CPU, RAM, and storage at different levels of abstraction, from process-specific to system-wide metrics. These tools are typically categorized as follows:

- System Monitors: Tools such as *top, htop*, and *vmstat* provide real-time snapshots of system resource usage. While useful for coarse analysis, they offer limited insight into specific application behaviour [60].
- Performance Profilers: More advanced tools like *perf, sysstat, strace,* and *iotop* allow fine-grained profiling of CPU cycles, memory accesses, and I/O operations. These tools are especially useful for benchmarking computationally intensive or I/O-bound tasks [61] [62].
- Container-Level Tools: In containerized environments, tools like *Docker Stats, cAdvisor,* and *Kubernetes* resource metrics expose per-container CPU, memory, and storage usage. This is particularly relevant in satellite ground emulation systems running microservices [63].
- Application-Level Profilers: Framework-specific profilers such as *Python's memory_profiler* or *TensorFlow's Profiler* provide deep integration for tracking function-level resource usage [64].
- Storage and Filesystem Monitoring: Utilities such as *du, df, iotop,* and *iostat* can be combined with logging frameworks to trace and record data footprint, access latency, and throughput [65].
- Custom Instrumentation: For embedded or real-time systems, developers often embed hooks and counters directly into the application source code, using APIs such as *getrusage()* or *clock_gettime()* to track system calls and execution time [66].

5.3.2 Methodology for resource profiling

Resource consumption profiling typically follows a structured methodology:

- Baseline Measurement: Initial measurements are taken during system idle and baseline operation.
- Incremental Profiling: Components are enabled one by one to isolate their individual resource usage.
- Stress Testing: Workloads are artificially scaled (e.g., synthetic traffic, concurrent tasks) to test peak performance and resource limits.
- Scenario Testing: End-to-end use cases (e.g., video streaming, sensor data relay) are executed in controlled environments to measure application-level behaviour.
- Comparative Analysis: Multiple configurations (e.g., algorithms, encoding settings, caching policies) are compared in terms of CPU/RAM/storage impact.

Metrics are collected using centralized logging systems (e.g., Prometheus, ELK stack) and visualized through dashboards (e.g., Grafana) [67].

5.3.3 Application to the 5G NTN testbed

The methodology described in the previous section will be implemented on the 5G NTN testbed developed in WP6 to evaluate resource consumption (CPU, RAM, and storage) associated with the execution of various applications and services. This analysis will help better understand the system requirements of the targeted use cases and possibly identify potential bottlenecks or areas for optimization if it assessed critical for the project.

At this stage, no specific tool has been selected from the various options identified. The selection will be guided by general criteria such as:

- Compatibility with the software environments, operating systems, and kernels of the target platforms
- Maturity, stability, and quality of available documentation
- Ease of use and integration into our workflows

At first glance, we do not anticipate the need for advanced professional tools capable of analyzing threads or individual functions in detail, as our objective is not focused on fine-grained optimization or industrialization.

Since the testbed is not yet fully operational at this stage of the project, concrete measurements can only be carried out at a later phase. The results from these test campaigns will be included in deliverable D6.5, which is dedicated to presenting the PoC integration and validation test results.

6 RAN SOFTWARIZATION

This section describes the general stack modifications that are needed to the PoC implementation. It will be integrated within WP6.

6.1 O-RAN ARCHITECTURE AND FUNCTIONAL SPLITS

SRS' CU/DU implementation is compliant with both 3GPP (rel. 17) and O-RAN Alliance specifications. It has been designed to be fully portable, scalable and support disaggregated deployments. As part of the 5G-STARTDUST tasks, the architecture and features of the CU/DU have been extended and refined to fulfil the project's demonstration requirements, as it will be detailed below.

6.1.1 Disaggregation status and supported splits

SRS' RAN implementation has been developed to support all splits of relevance to 5G-STARDUST, as shown in Figure 49.

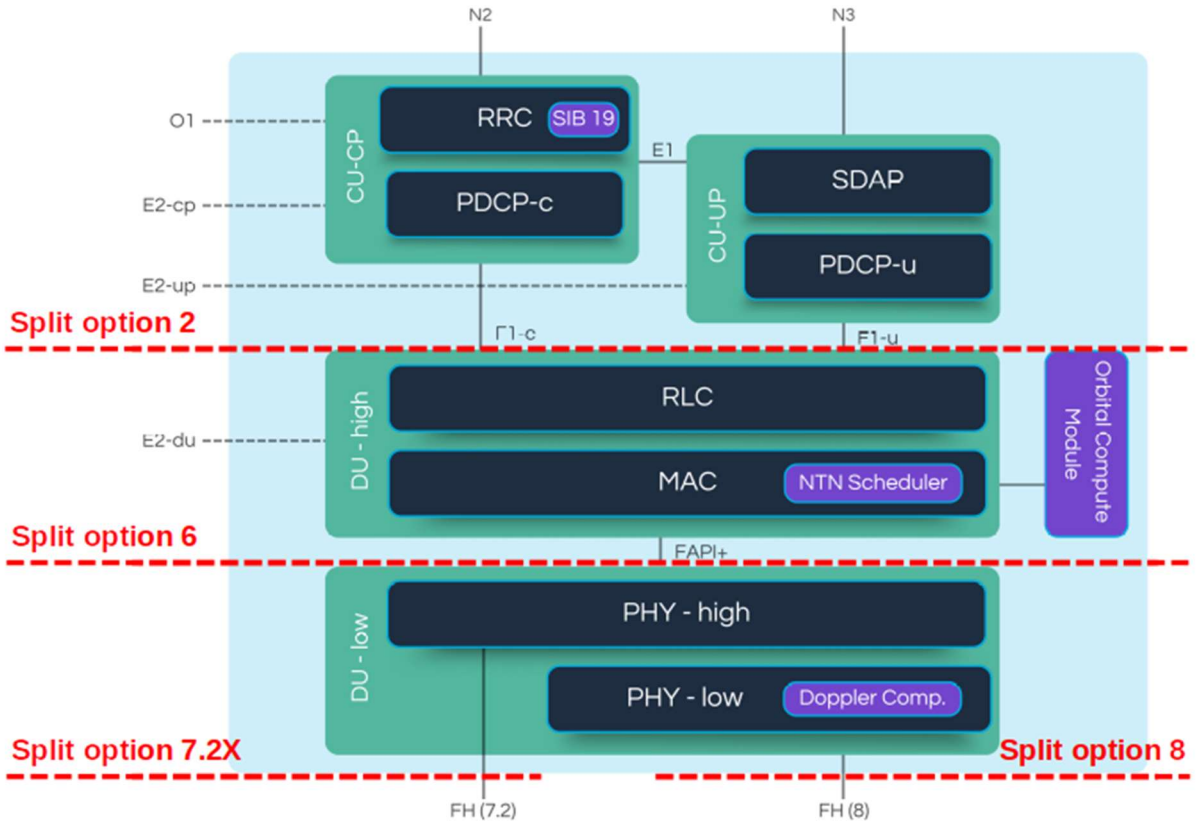


Figure 49: Functional split options supported by SRS CU/DU implementation.

Split option 2

Split option 2 is fully supported by SRS gNB, enabling the disaggregated deployment of the CU and the DU components in different machines. Furthermore, the CU is further divided into the CU-UP and CU-CP. In this extra step, the CU-UP handles the SDAP and the user segment of the PDCP, whereas the CU-CP manages the RRC procedures and control segment of the

PDCP. Following the O-RAN Alliance specifications, the E1 interface is used to communicate the CU-CP and the CU-UP, whereas the F1-C and the F1-U are be used to communicate with the DU respectively. The separation of the CU into independently deployable CU-UP and CU-CP components is currently ongoing, which will provide extra flexibility and efficiency to manage the distribution of RAN functions.

Split option 6

The SRS DU implementation supports the separation of the MAC and PHY layers, which will use the FAPI interface to communicate, following a set of 3GPP-defined specifications. The utilization of third-party PHY-layer implementations is also supported through this deployment option.

Split option 7.x

The SRS DU supports the integration to O-RUs by means of an in-house implementation of the OFH interface, as defined by the O-RAN Alliance. In this case, the DU will host up to the (more complex and computing-intensive) high PHY-layer functions, whereas the O-RU will handle the low PHY-layer and the RF processing. Integration with a wide range of commercial O-RUs has been validated by SRS both in laboratory and real-world deployments.

Split option 8

Support of software defined radio (SDR) platforms, which will implement the RF processing, is also provided by SRS DU. In this case, the full PHY-layer will be hosted in the DU, which will forward I/Q samples to the SDR platform by using a high-speed link over an open radio transport protocol (e.g., VITA Radio Transport – VRT – or compressed VITA – CVITA –). This split is commonly used in laboratory setups.

6.1.2 Discussion on gNB deployment options in NTN scenarios

As introduced in Section 2, having the full gNB (CU and DU) flying seem the most reasonable and straightforward deployment options to be used in NTN scenarios, when taking into account the current O-RAN Alliance and 3GPP specifications, as well as the typical latencies of satellite links. Yet, a brief discussion of the different options is provided next.

Specific NTN features and modifications have been added to the SRS CU/DU implementation to enable support for regenerative payloads, with the full gNB deployed in space, to account for long propagation delays or Doppler effects, as highlighted (in purple) in Figure 49. Yet, hosting the complete stack on the satellite increases its computing and power consumption requirements.

Regarding Split option 2, with a latency budget of up to 10 ms between PDCP (CU) and RLC (DU), and relatively low bandwidth requirements (exchange of control and user plane signalling data), deploying the CU on the ground and putting the DU in orbit can be an option for LEO satellite use cases, with some limitations (e.g., maximum altitude of the satellite, maximum data-rate, larger buffer requirements to accommodate the increased F1 interface latency [10]). As in the previous case, the computational and power requirements of the satellite are kept up, being the DU the most computing intensive component of the gNB.

The disaggregation of the CU and the DU, resulting from adopting Split option 2, can also provide some interesting deployment options when both are located in space. For instance, a deployment with a single CU controlling various DUs (e.g., each implementing one beam) can

be interesting, e.g., when considering energy-efficient realizations (such as dynamically powering-off DUs if the number of connected users in the DU's cell is low).

When considering partially regenerative payloads, with parts of the DU hosted in the satellite and the remaining functions deployed on ground, the adoption of split options 6 (i.e., PHY and RF processing in space, requiring to have the FAPI over the feeder link), 7.2 (i.e., low-PHY and RF processing in the satellite, putting the OFH over the feeder link) or 8 result impractical when taking into account the typical round-trip latencies of LEO, MEO and GEO, and the bandwidth requirements, as detailed in Section 2. Nevertheless, the adoption of those splits in fully regenerative architectures can allow leveraging some of the benefits observed in terrestrial deployments (e.g., greater scalability, lower-cost radio hardware design) or building more complex space network topologies, optimized for specific use cases [6] (e.g., energy-efficiency, DU scalability in terms of computing vs performance).

6.2 NTN-SPECIFIC ENHANCEMENTS IN THE GNB

The gNB stack from SRS has been modified and extended to add new radio functions to enable NTN functionality within an O-RAN-compatible architecture. A general overview of the developments done in the framework of 5G-STARDUST is provided below (the full details of these modifications are covered in D4.9 [68]).

SIB19 support

The gNB supports SIB19 to provide satellite assistance information. Two different options are implemented for Ephemeris information: i) position and velocity vectors, as shown in Figure 50, and ii) orbital parameters.

`ntn:`			
	`cell_specific_koffset: 239`		`# Cell-specific k-offset.`
	`ta_common: 0`		`# TA common offset.`
	`ephemeris_info_ecef:`		`# Satellite ephemeris in position and velocity state vector format.`
		`pos_x: -28105880`	
		`pos_y: 31509747`	
		`pos_z: -1691895`	
		`vel_x: 34`	
		`vel_y: 9`	
		`vel_z: -385`	

Figure 50: Excerpt of the SRS gNB configuration file, focusing on NTN parameters [69].

An external SIB19 generator has also been implemented, with a rapid updating mechanism in the gNB. This enables the generation of timestamped SIB19 content using satellite TLE data, the ground location of the UE (access link), and a ground station (feeder link). Furthermore, a SIB19 updater mechanism using JSON over Websocket has been added as well, which simplifies the integration of the gNB with OAM services. Finally, improved SIB19 content precision is attained through an efficient mechanism for tracking SFN-Timestamp mapping.

HARQ support

Two different NTN HARQ management approaches have been implemented in the gNB:

i. *Disabled HARQ:* support for the straightforward NTN HARQ overhead-reducing solution has been added to SRS gNB in two different forms:

   - As dictated by the rel. 17 standard, by implementing the *DownlinkHARQ-FeedbackDisabled-r17* signalling in the RRC, to inform the UEs to disable their HARQ reporting (i.e., no ACK/NACK will be sent by the UEs for the DL transmissions).

   - Using a semi-custom solution, in coordination with the NTN UE developments done by CTTC, which will keep sending the HARQ reports for the DL, whereas the gNB updates its MAC scheduling to accommodate the extended delays and implements a history of flushed HARQ messages.

ii. *Increased HARQ process support:* the MAC scheduler of the gNB has been enhanced to support up to 32 HARQ processes, which enables the full use of HARQ in NTN use cases (i.e., normal utilization of ACK/NACK messages for DL transmissions). With this, the gNB will retransmit DL data as required, increasing the chances of correct decoding under adverse channel conditions.

Handling of feeder link

The DU from SRS has been enhanced in two main fronts for feeder link support. First, broadcasting of timing advance (TA) information has been added as part of the SIB19, as specified in the rel. 17 standard, including the common TA (TA-common) offset to account for the propagation delay, and the drift rate of the common TA (TA-Common-Drift) and its variation (TA-Common-Drift-Variant).

Second, compensation of the Doppler shift has been included to both DL and UL channels. In more detail, in the case of the DL transmission, the gNB applies a Doppler pre-compensation with the objective to help mitigating it, whereas in the case of the UL channel, a post-compensation is applied in the DU.

NTN cell bandwidth

The SRS' gNB has been natively designed to be flexible and scalable. From the stack point of view, the CU/DU supports all bandwidths from 5 to 100 MHz. Given that the NTN functionality added under the 5G-STARDUST umbrella is built on top of the terrestrial gNB implementation, support for a 100 MHz bandwidth is natively provided by SRS implementation when targeting satellite use cases as well. This being said, release 18 of 3GPP's specifications is introducing support for up to 20 MHz channel bandwidth in the NTN bands (i.e., n254, n255) [70]. Amongst others, this implies that currently available NR NTN UE implementations are not yet supporting bandwidths above the 3GPP specifications. For this reason, only limited evaluation of bandwidths greater than 20 MHz has been performed in the laboratory (as detailed in D6.3 [71]).

Other extensions and enhancements

The accommodation of the previously detailed NTN-specific features has required the enhancement and extension of other parts of the stack. Most relevantly:

- Complimentarily to the SIB19, the *Kcell_offset* parameter, as defined in TS 38.214 [72], has been introduced in the MAC scheduler implementation, to account for the extended RTT in satellite use-cases. This enhancement affects a number of queues and control timers, to enable the correct operation of the gNB (starting from the initial attach procedure).

- Extension of buffers, control timers and ASN1-related procedures in the RLC and RRC layers, amongst others to enable the retransmission of wrongly decoded data messages, complimentarily to the HARQ modifications.

Furthermore, the whole NTN function-library of the gNB has been fully refactored to improve its modularity, ease of extension, maintenance and (automated) testing.

6.3 RIC AND O&M SERVICE INTEGRATION

The non-RT RIC is meant to provide orchestration and management means to the CU/DU, by executing rApps tailored for large timescale (up to minutes) RAN optimization actions (e.g., policy computation, radio resource management). Complimentarily, the near-RT RIC focuses on RAN control and monitoring in much lower timescales (between 10 ms and 1 second), through xApps meant to provide fine-grain adaptability in response to the dynamic network variations. Furthermore, the latter can be steered by the outputs of the non-RT RIC to attain a fully optimized RAN at all times.

The gNB implements the E2 interface that enables the interconnection of the CU and the DU with near-RT RICs. In more detail, the E2 implementation of SRS supports the E2 service model (E2SM)-key performance metric (KPM) to provide a wide variety of run-time metrics to monitor the status of the RAN from the point of view of the different CU/DU stack layers (e.g., RLC, RRC, PHY). By monitoring the RAN, its operation can be optimized by avoiding undesired situations or adapting its configuration in response to the observed metrics. In this sense, the CU/DU implementation also supports the E2SM-RAN control (RC) to trigger RAN adaptations and take the required control actions. During the development of 5G-STARDUST, the control action ID 1, of the RIC style type 3 connected mode mobility control was added to the gNB [73], which is especially relevant to the satellite use case as it enables handover control via xApps (e.g., based on the observed UE throughput/latencies or network load). This feature is a first step towards the conditional handover (e.g., in a TN/NTN coexistence case).

6.3.1 O1 interface introduction

A baseline implementation of the O1 interface has also been added recently to the SRS' gNB implementation. Through O1 it is supported the remote & automated configuration of the gNB, which is the corner stone of orchestration. That is, the most relevant gNB configuration parameters [74] can be defined via O1 and override the default values (e.g., rApp coordinating the deployment of various cells with an optimum coexistence configuration).

7 CONCLUSIONS

Architectural Split and Radio Intelligence Controller Placement

Disaggregating gNB components in NTNs implies novel architectural and operational challenges due to the mobility of hosting platforms like LEO satellites and the strict latency and synchronization constraints of terrestrial RAN interfaces. While functional split architectures (presented Options 1 to 3 in Sections 2.2, 2.3 and 2.4) offer deployment flexibility, they require re-evaluating all core interfaces and control functions in the 5G stack.

In Option 1, the DU and RU are co-located onboard the satellite, while the CU remains on the ground. This reflects a Split 2 architecture, where lower-layer functions (e.g., PHY, MAC, HARQ) are processed in space, reducing latency for time-sensitive operations. However, the F1 interface must traverse the feeder link between satellite and Earth. This link is sensitive to propagation delay, intermittency, and jitter, which may compromise control-plane responsiveness and degrade user-plane throughput under variable link quality. In contrast, N2 and N3 interfaces (connecting the CU to the 5G CN) and the E2 interface (between CU and near-RT RIC) remain fully terrestrial and are unaffected by the satellite hop. Still, the end-to-end responsiveness of the RAN is limited by the F1 bottleneck, particularly during handovers or real-time reconfigurations initiated from the CU.

Option 2a places the RU, DU, and CU on a single satellite. This removes the need for internal F1 and E1 links and avoids fronthaul timing issues, but introduces feeder link constraints for N2/N3. If the RIC stays grounded, the E2 interface suffers from propagation delay, impacting time-sensitive functions like beamforming or mobility control. Deploying near-RT RICs onboard may mitigate these delays but brings added complexity for lifecycle management and reliability under satellite resource constraints. Option 2b spreads RU, DU, and CU across different satellites connected by ISLs. This modular approach supports scalability and redundancy, but stresses protocol limits, for instance in F1-C interface. This limits CU placement and requires strict control over constellation topology. E1 separation between CU-CP and CU-UP faces similar issues with added complexity for session state consistency across moving platforms. The Xn interface, essential for inter-gNB coordination, is poorly suited for dynamic satellite environments and needs redesign for beam-aware handovers and fast UE context forwarding. E2 becomes critical in this distributed setup. Near-RT control suffers under multi-hop signalling delays and losses. Solutions include embedding near-RT RICs with each DU or CU or clustering RICs across local satellites. These strategies require efficient synchronization and orchestration. xApps must be space-aware, using telemetry, visibility maps, and link conditions to manage HARQ, beamforming, and load balancing with limited compute resources. As satellites move or become unavailable, DU and CU nodes may need to migrate. Existing context transfer protocols are insufficient for maintaining continuity across F1 and E1 under dynamic topologies. Synchronization, vital for RAN performance, is also challenged by variable ISL delays. New timing methods or predictive mechanisms are needed to avoid HARQ and scheduling issues from skew and misalignment.

Option 3 adds onboard UPF, turning satellites into edge data nodes with local breakout over N6. This benefits edge computing use cases and eases backhaul load but creates new problems for routing, security, and state persistence. First, hosting gNB and UPF functions on a satellite imposes significant processing demands on the spacecraft. The satellite must perform real-time baseband processing, packet routing, and possibly edge computations within strict size, weight, power, and radiation constraints. The ESA 5G Space-Based Infrastructure paper [18] estimates the power needed: in a mid-term LEO scenario, just the NR baseband processing (full gNB) plus feeder link modem could draw on the order of 130+ watts onboard where the user link (onboard NR baseband processing/full gNB) consumes approximately 78.6

W and the feeder link modem (DVB-S2X) draws about 55.9W. The computing load is significantly higher -for a fully orbital gNB compared to split architectures (55-70 % more demands) [24]. This means that more powerful onboard CPUs/DSPs or dedicated accelerators are required. In fact, researchers are also actively researching specialised hardware for satellite edge processing. The article in [27], for example, presents “GPU@SAT”, a GPU-like high-performance accelerator implemented on an FPGA and designed for onboard processing of intensive tasks in space. These efforts emphasise the need for space-grade computing hardware (multi-core CPUs, FPGAs/GPUs, AI accelerators) for the 5G RAN and the CN functions in orbit. At the same time, UPF failure or satellite loss also demand fast recovery and session relocation. For this reason, RICs must now account for N6 optimization and service placement, increasing their role in system orchestration.

Among the three principal architectural options studied, Option 2a-with the full gNB stack (RU, DU, CU) integrated onboard a single satellite- emerges as the most practical and balanced choice in terms of protocols and hardware standpoint under current technology constraints. It avoids fronthaul and midhaul timing bottlenecks, reduces signalling latency, and enables low-complexity RAN management while preserving compatibility with 5G CN interfaces. Its self-contained structure simplifies orchestration, enhances scalability, and lowers dependency on persistent feeder links or high-throughput ISLs. These qualities make Option 2a particularly suited for medium-class LEO/MEO constellations targeting broadband NTN deployments in the near term. Looking ahead, O-RAN will be instrumental in enabling intelligent, flexible, and modular NTN systems, serving as the architectural backbone for 6G networks that span terrestrial and non-terrestrial domains.

As NTN integration progresses toward commercialization in 5G-Advanced and 6G ecosystems, several future directions must be pursued to mature the architectural and functional split models proposed in this work.

1. AI-Native Orchestration and Autonomy

Beyond static placements of functions, future systems should support advanced reconfigurable splits that can adapt to factors like link quality, mobility, and computational load. This level of dynamic control requires highly sophisticated orchestration platforms capable of real-time telemetry ingestion and predictive resource management across space and ground domains. This intelligent orchestration will optimize the interplay between Near-RT and Non-RT RICs. For instance, federated learning and semantic policy distillation can be used to manage these disaggregated control loops, enabling efficient global policy enforcement and local adaptation. Intent-based networking will also play a central role in this process. By moving beyond rigid configurations and toward a more autonomous, AI-driven architecture, NTN systems can achieve greater resilience, performance, and efficiency.

2. Implications for Standardization: 3GPP and O-RAN Alliance

The architectural and RIC placement strategies proposed in the project have significant implications for the evolution of both 3GPP and O-RAN Alliance standards. While 3GPP Release 17 and 18 have introduced support for NTN, these specifications primarily address transparent and regenerative payloads, and do not fully account for the architectural disaggregation enabled by O-RAN. Our findings highlight key areas where current standards fall short and require enhancement to achieve efficient TN-NTN convergence.

3GPP Implications: The mobility of non-geostationary satellites poses challenges to the existing 3GPP protocols, which assume stable connectivity. The NG-Flex configuration in TS 38.410 is a good starting point for managing gateway reassignment and session re-anchoring in full gNB deployments onboard (Option 2). However, further work is needed to standardise

robust mechanisms for IP mobility, bearer context synchronization and QoS management under variable connection conditions. The study presented in TSG SA WG2 to support satellite edge computing via UPF onboard is also highly relevant, and confirms the feasibility of Option 3. In this regard, 3GPP could strengthen these efforts to define standard procedures for UPF onboard deployment, N9-based inter-satellite routing, and fast recovery from satellite failures.

O-RAN Alliance Implications: The Open Fronthaul and F1/E1 interfaces are not designed for the long round-trip delays and Doppler shifts that occur in space links. A new working group could be established to develop space-adapted fronthaul and midhaul standards that can cope with these limitations. This includes defining new transport layers and modifying HARQ and scheduling cycles. In addition, the O-RAN Alliance needs to standardise inter-RIC protocols, such as suggested in Section 2.5.2, to enable seamless coordination and state transfer between Near-RT-RIC instances distributed across a constellation. These new standards should also address the dynamic assignment of E2 nodes and the migration of xApps and rApps to support a scalable and resilient control plane in NTNs. Finally, O-RAN's RICs should be enhanced to take into account ISL conditions and satellite telemetry for smarter and predictive resource management.

Probabilistic forecasting with P-KANs for RRM

Beyond single-point prediction, this deliverable also addressed the problem of resource forecasting under uncertainty by introducing probabilistic extensions of KANs (P-KANs) for PRB demand prediction. Unlike conventional models that provide only a single expected value, P-KANs output full predictive distributions, enabling the system to explicitly account for uncertainty in network dynamics. This capability is critical in non-terrestrial networks (NTNs), where traffic patterns are highly variable due to satellite mobility, user heterogeneity, and limited feeder link capacity. By quantifying the range of possible traffic outcomes, probabilistic forecasting supports more risk-aware resource management policies, such as adaptive bandwidth reservation, congestion avoidance, and resilience against unexpected demand spikes.

Experimental results based on real satellite traffic traces show that P-KANs outperform baseline probabilistic MLPs in terms of both accuracy and reliability. Specifically, improvements are observed in key probabilistic metrics such as the CRPS and the FIC, confirming the ability of P-KANs to deliver sharper and better-calibrated forecasts. The exploration of different output distributions, including Gaussian and T-Student, highlights trade-offs between flexibility and robustness: Gaussian P-KANs offer stable and well-calibrated intervals, while T-Student variants capture heavier tails and rare events more effectively. Importantly, these probabilistic models reduce systematic over-provisioning of resources, leading to more efficient spectrum usage while still safeguarding quality of service (QoS).

From an operational perspective, probabilistic forecasting with P-KANs represents a step toward AI-native and uncertainty-aware RRM in integrated terrestrial–non-terrestrial networks. By allowing operators to incorporate predictive confidence into decision-making, these models enhance adaptability to real-world dynamics where deterministic predictions fall short. This are steps for self-optimizing NTN systems capable of balancing efficiency, reliability, and robustness under highly dynamic conditions.

Fed-KAN

In the proposed approach, KAN is used within satellites as FL client to enable efficient traffic prediction. The proposed Fed-KAN model has exhibited several significant advantages over the conventional Fed-MLP models in the context of NTN traffic prediction based on real satellite network operator data. The use of FL clients (beams) for local model training can improve

scalability, data privacy and adaptability. More specifically, experimental results indicate that Fed-KAN achieves a 77.39% reduction in average test loss compared to Fed-MLP, demonstrating improved model prediction performance. Hence, the capability of Fed-KAN to model non-linear relationships in network traffic data and its robustness to fluctuations can improve decision-making in resource allocation and optimization in satellite connectivity. Finally, efficient model updates can improve adaptability to real-world operational constraints, ensuring robust and uninterrupted service delivery. Future work can concentrate on enhancing the capabilities of Fed-KAN for AI-native integrated networks including heterogeneous domains such as non-terrestrial and terrestrial networks (with xHaul, Open RAN, non-public, edge and cloud connectivity). In addition, the integration of the Fed-KAN framework with digital twinning applications can enable real-time performance and proactive adjustments to improve resilience and service delivery in various domains of integrated networks.

Multivariate probabilistic forecasting

Multivariate probabilistic forecasting has been proposed to support bandwidth allocation strategies. For the numerical evaluation, dynamic allocation schemes are considered. In this framework, bandwidth assignments are adjusted based on traffic demand predictions generated by model-driven forecasting techniques, enabling a more responsive and potentially more efficient use of resources compared to the baseline static allocation approach. Within the dynamic allocation setting, the performance of multivariate probabilistic forecasting techniques for multi-beam satellites is benchmarked against univariate methods. The numerical results show that AI-based multivariate probabilistic models significantly reduce over-provisioning while also achieving a substantial decrease in under-provisioning rates. Overall, these techniques offer an attractive trade-off, delivering notable bandwidth savings (around 45%) and maintaining a very low under-provisioning rate (below 2%).

Near-RT RIC solutions

The objective of this study was that of exploiting AI/ML solutions to estimate the SINR of a group of users scheduled for transmission in a user-centric beamforming system, reducing the computational complexity with respect to the assessment of the SINR through the beamforming matrix while maintaining satisfying estimation accuracy.

The considered architecture, based on the MHSA mechanism, achieves a reduction of the Big-O complexity of a factor 3 in the CSI-based case and of two orders of magnitude in the location-based case. This last result is a consequence of the lower dimensionality of the user reports for location-based user-centric beamforming systems, which avoid scaling the computational complexity by the number of antenna elements N_R. Furthermore, both models achieve satisfying SINR estimation performance considering users with various capacity requirements being scheduled with the PQS algorithm, with the RMSE not passing the 1dB threshold in most of the cases.

With the estimation performance being evaluated, the DMHSA models can be deployed for inference in different procedures. Scheduling algorithms typically provide a single group of users to be served during the same time slot. While mechanisms, *e.g.*, the priority queue and the CSI correlation / inter-user distance check of the PQS algorithm, are in place to improve the capacity offered in a time slot, the decision is often sub-optimal in this regard. Thus, the capacity to a group of potentially scheduled users may be evaluated through the SINR by means of DMHSA-based estimation. This opens the way to schedulers that propose multiple candidate groups and prioritize that providing the highest average SINR, *i.e.*, the highest offered capacity.

On-board processing

Three incremental on-board architectures are considered, each requiring more demanding resources, but also offering more features: CU/DU splitting, full gNB on-board and full gNB with some UPF functions on-board. Each option can be associated with a certain time horizon (short, mid, long term). These architectures impose a non-trivial burden on the satellite OBP, which must support high-rate baseband signal processing and real-time scheduling decisions. Depending on the supported bandwidth, number of antenna ports and expected user load, the satellite must be provisioned with high-performance CPUs, possibly accompanied by FPGAs or GPUs for acceleration of PHY layer operations. As satellite hardware becomes more modular, we can expect a growing shift toward intelligent constellations operating closer to the edge of data generation. These advances will enable satellites to handle petabyte-scale sensing, perform autonomous AI model training and inference, and conduct adaptive mission planning without constant ground intervention. As regulatory and commercial ecosystems evolve, future constellations will likely operate as a distributed computing layer in LEO, forming the backbone of space-based edge intelligence.

RAN softwarisation

This deliverable outlines the extensions necessary to integrate the RAN solution into the PoC. Implementation efforts have focused on supporting different split options, to implement E2 and O1 interfaces and to modify radio functions to enable the NTN functionality within an O-RAN compatible architecture. As discussed in Section 2, deploying the full gNB on board the satellite appears to be the most reasonable and straightforward option. However, hosting the complete stack in space significantly increases computing and power consumption requirements. In this context, disaggregating the CU and DU can offer alternative deployment scenarios, particularly when both are located on the satellite. For example, a single CU could control multiple DUs, enabling energy-efficient strategies such as dynamically powering down DUs when the number of connected users within a given cell is low.

8 REFERENCES

[1] 5G-STARDUST, «D4.4: Preliminary report on AI-based radio resource management, RAN softwarisation and onboard processing,» 2024.

[2] SNS JU Project 5G-STARDUST, D4.6 "Final report on the Unified Air Interface, User-Centric and Beamforming Solutions", 2025.

[3] J. Baranda, M. Caus, L. Blanco, C. J. Vaca-Rubio, E. Zeydan, K. Dev, Z. Li y T. d. Cola, «On the architectural split and radio intelligence controller placement in integrated O-RAN-enabled non-terrestrial networks,» *arXiv preprint arXiv:2507.02680,* 2025.

[4] O-RAN Alliance, «O-RAN Alliance,» 2025.

[5] L. M. P. Larsen, A. Checko y H. L. Christiansen, «A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks,» *IEEE Communications Surveys & Tutorials,* vol. 21, nº 1, pp. 146-172, 2019.

[6] E. Baena, P. Testolina, M. Polese, D. Koutsonikolas, J. Jornet y T. Melodia, «Space-O-RAN: Enabling Intelligent, Open, and Interoperable Non Terrestrial Networks in 6G,» *arXiv preprint arXiv:2502.15936,* 2025.

[7] 3GPP TS 38.401, «NG-RAN; Architecture description (Release 18),» 2025.

[8] 3GPP TS 38.470, «NG-RAN; F1 general aspects and principles (Release 18),» 2024.

[9] Ericsson AB; Huawei Technologies Co. Ltd.; NEC Corporation; Nokia, «Common Public Radio Interface: Requirements for the eCPRI Transport Network V1.2,» 2018.

[10] 3GPP TR 38.821, «Solutions for NR to support Non-Terrestrial Networks (NTN)».

[11] G. Masini et al., "5G meets satellite: Non-terrestrial network architecture and 3GPP,," *International Journal of Satellite Communications and Networking,* vol. 41, no. 3, p. 249–261, 2023.

[12] X. Lin, S. Rommer, S. Euler, E. A. Yavuz y R. S. Karlsson, «5G from Space: An Overview of 3GPP Non-Terrestrial Networks,» *IEEE Communications Standards Magazine,* vol. 5, nº 4, p. 147–153, 2021.

[13] F. Rossato, «Simulation Study, Design and Implementation of 5G NR Layer-2 Protocols for Non-Terrestrial Networks,» *Università degli studi di Padova,* 2024.

[14] H. B. Tsegaye, «Towards Resilient and Secure Beyond-5G Non-Terrestrial Networks (B5G-NTNs): An End-to-End Cloud-Native Framework,» *Università degli studi di Trento,* 2024.

[15] 3GPP TS 38.410, «NG-RAN; NG general aspects and principles,» 2023.

[16] 3GPP TS 23.501, «System Architecture for the 5G System (5GS), including AMF Region and AMF Sets,» 2023.

[17] O-RAN Alliance,, «Deployments of O-RAN-based Non-Terrestrial Networks,» 2025.

[18] European Space Agency (ESA), A. Gavras, and H. Zaglauer,, "White Paper on 5G Space-based Infrastructure (5G-IS)," European Space Agency, White Paper ESA ARTES Contract No. 4000134982/21/NL/CLP, 2024.

[19] European Space Agency (ESA) 5G SPL Team, «Demonstration of a 5G g-NodeB in Space: Webinar Slides (Public Version),» European Space Agency, Webinar Slides, 2021.

[20] 3GPP , «Key Issue #2: Support of Satellite Edge Computing via UPF on board,» 3GPP Presentation Slide, 2022.

[21] 3GPP TR 23.700-27, «Study on 5G System with Satellite Backhaul,» 2022.

[22] Y. Jiang, W. He, W. Liu, S. Wu, X. Wei y Q. Mo, «A B5G Non-Terrestrial-network (NTN) and Hybrid Constellation Based Data Collection System (DCS),» *MDPI Aerospace,* vol. 10, nº 4, 2023.

[23] Y. Liu, L. Wang, Z. Lu y G. Shou, «A QoS Guaranteed Efficient Integration of UPF and LEO Satellite Networks,» *IEEE Transactions on Network and Service Management,* vol. 21, nº 4, p. 3727–3739, 2024.

[24] S. S. S. G. Seeram, L. Feltrin, M. Ozger, S. Zhang y C. Cavdar, «Handover Challenges in Disaggregated Open RAN for LEO Satellites: Tradeoff between Handover Delay, and Onboard Processing,» *Frontiers in Space Technologies,* vol. 6, 2025.

[25] SNS JU 5G-STARDUST, «D5.5: Final report on AI-data driven,» 2025.

[26] SNS JU Project 5G-STARDUST, D3.2 "End-to-end ground and space integrated architecture", Feburary 2024.

[27] G. Benelli, G. Todaro, M. Monopoli, G. Giuffrida, M. Donati y L. Fanucci, «GPU@ SAT DevKit: Empowering Edge Computing Development Onboard Satellites in the Space-IoT Era,» *MDPI Electronics,* vol. 13, nº 19, p. 2024.

[28] Z. Liu et al.,, «Kan: Kolmogorov-arnold networks,» *arXiv preprint arXiv:2404.19756,* 2024.

[29] A. N. Kolmogorov, «On the representation of continuous functions of several variables by superposition of continuous functions of a smaller number of variables,» *American Mathematical Society ,* 1961.

[30] C. J. Vaca-Rubio, L. Blanco, R. Pereira y M. Caus, «Kolmogorov-Arnold Networks (KANs) for time series analysis,» *arXiv preprint,* 2024.

[31] L. C. K. Xu y S. Wang, «Kolmogorov-Arnold Networks for time series: Bridging predictive power and interpretability,» *arXiv,* 2024.

[32] C. J. V.-R. e. al., «Probabilistic Forecasting for Network Resource Analysis in integrated Terrestrial and Non-Terrestrial Networks,» *IEEE Communications Standards Magazine,* 2025.

[33] E. Zeydan, C. J. Vaca-Rubio, L. Blanco, R. Pereira, M. Caus y A. Aydeger, «F-KANs: Federated kolmogorov-arnold networks,» de *Proceedings of the 1st workshop on distributed AI for enhanced wireless networks (DAINET) 2025, in conjunction with IEEE consumer communications & networking conference (CCNC)*, 2025.

[34] E. Zeydan, C. J. Vaca-Rubio, L. Blanco, R. Pereira, M. Caus y K. Dev, «Fed-KAN: Federated Learning with Kolmogorov-Arnold Networks for Traic Prediction.,» *arXiv preprint arXiv:2503.00154,* 2025.

[35] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," *Artificial intelligence and statistics, PMLR,* p. 1273–1282, 2017.

[36] Q. Duan, J. Huang, S. Hu, Z. L. R. Deng y S. Yu, «Combining federated learning and edge computing toward ubiquitous intelligence in 6G network: Challenges, recent advances, and future directions,» *IEEE Communications Surveys & Tutorials,* 2023.

[37] H. Chen, M. Xiao y Z. Pang, «Satellite-based computing networks with federated learning,» *IEEE Wireless Communications,* vol. 29, nº 1, p. 78–84, 2022.

[38] Y. Jing, J. Wang, C. Jiang y Y. Zhan, «Satellite MEC with federated learning: Architectures, technologies and challenges,» *IEEE Network,* vol. 36, nº 5, p. 106–112, 2022.

[39] J. Schmidt-Hieber, «The kolmogorov–arnold representation theorem revisited,» *Neural networks,* vol. 137, p. 119–126, 2021.

[40] 5G-STARDUST, «D4.1: Open data sets for ML-based RRM,» *2024.*

[41] S. H. a. J. Schmidhube, «Long Short-Term Memory,» *Neural Computation,* vol. 9, nº 8, pp. 1735-1780, 1997.

[42] P. B. M. O. a. Y. B. M. Ravanelli, «Light Gated Recurrent Units for Speech Recognition,» *in IEEE Transactions on Emerging Topics in Computational Intelligence,* vol. 2, nº 2, pp. 92-102, 2018.

[43] B. N. Oreshkin, «Neural basis expansion analysis for interpretable time series forecasting,» de *International Conference of Learning Representations (ICLR)*, 2020.

[44] C. Challu, K. G. Olivares, B. N. Oreskin y F. R. Garza, «NHITS: Neural Hierarchical Interpolation for Time Series Forecasting,» de *AAAI Conference on Artificial Intelligence*, 2023.

[45] B. Lim, S. Ö. Arik, N. Loeff y T. Pfister, «Temporal fusion transformers for interpretable multi-horizon time,» *International Journal of Forecasting,* vol. 37, nº 4, p. 1748–1764, 2021.

[46] 3GPP, TR 38.811 "Study on New Radio (NR) to support non-terrestrial networks (Release 15)", September 2020.

[47] 5G-STARDUST, «D4.3: Report on the Unified Air Interface, User-Centric and Beamforming Solutions,» April 2024.

[48] 3GPP, TR 38.803 "Study on new radio access technology: Radio Frequency (RF) and co-existence aspects (Release 14)", September 2019.

[49] SNS JU Project 5G-STARDUST, D4.5 "Report on the Unified Air Interface, User-Centric and Beamforming Solutions", 2025.

[50] Y. I. Demir, M. S. J. Solaija y H. Arslan, On the Performance of Handover Mechanisms for Non-Terrestrial Networks, 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), August 2022.

[51] D. Bahdanau, K. Cho y Y. Bengio, Neural machine translation by jointly learning to align and translate, 3rd International Conference on Learning Representations (ICLR) 2015, 2015.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser y I. Polosukhin, Attention is all you need, Advances in neural information processing systems (NeurIPS), 2017.

[53] D. P. Kingma y J. Ba, Adam: a method for stochastic optimization, rd International Conference on Learning Representations (ICLR), 2015.

[54] I. Loshchilov y F. Hutter, SGDR: stochastic gradient descent with warm restarts, 5th International Conference on Learning Representations (ICLR), 2017.

[55] Viasat, «Viasat-2 Satellite Overview,» 2022. [En línea]. Available: https://www.viasat.com/.

[56] OneWeb, «OneWeb Satellite Constellation Factsheet,» 2023. [En línea]. Available: https://www.oneweb.world/.

[57] SES, «SES-17 and O3b mPOWER White Paper,» 2023. [En línea]. Available: https://www.ses.com/.

[58] European Space Agency, «OPS-SAT: ESA's flying laboratory,» 2022. [En línea]. Available: https://www.esa.int/.

[59] SpaceX, «Starlink Engineering Upadates,» 2023. [En línea]. Available: https://www.spacex.com/starlink.

[60] Linux Foundation, «Linux Performance Monitoring Tools,» 2023. [En línea]. Available: https://linux.die.net/.

[61] B. Gregg, «Linux Performance Tools,» *ACM Queue,* vol. 14, nº 1, pp. 48-72, 2016.

[62] perf Wiki, «perf: Linux profiling with performance counters,» 2023. [En línea]. Available: https://perf.wiki.kernel.org/.

[63] Google, «cAdvisor: Container Advisor,» 2023. [En línea]. Available: https://github.com/google/cadvisor.

[64] Python Software Foundation, «memory_profiler,» 2023. [En línea]. Available: https://pypi.org/project/memory-profiler/.

[65] man7.org, «iotop(1) - Linux man page,» 2023. [En línea]. Available: https://linux.die.net/man/1/iotop.

[66] IEEE, «POSIX System Application Programming Interface,» *IEEE Std 1003.1,* 2017.

[67] Prometheus Authors, «Prometheus Monitoring System and Time Series Database,» 2023. [En línea]. Available: https://prometheus.io/.

[68] 5G-STARDUST, «D4.9: Development and Test Results,» 2025.

[69] srsRAN gNB for 5G NTN, «srsRAN Project Documentation».

[70] J. Krause, «Non-Terrestrial Networks (NTN) overview, 3GPP MCC».

[71] 5G-STARDUST, «D6.3: PoC Preliminary integration and validation test results,» 2025.

[72] 3GPP TS 38.214, «NR; Physical layer procedures for data (Release 18),» 2023.

[73] ORAN SC RIC, «docker with example xApps and srsRAN configurations».

[74] Configuration Reference,, «srsRAN Project Documentation».

[75] 5G-STARDUST, «D4.6: Final report on the Unified Air Interface, User-Centric and Beamforming Solutions,» 2025.