Deliverable 4.2

Initial Evaluation of BeGREEN O-RAN Intelligence Plane, and AI/ML Algorithms for NFV User-Plane And Edge Service Control Energy Efficiency Optimization

September 2024

Pending European Commission Approval

Co-funded by
the European Union

6GSNS

Contractual Date of Delivery:	**October 2, 2024**
Actual Date of Delivery:	**July 31, 2024**
Editor(s):	**Miguel Catalán-Cid (i2CAT)**
Author(s)/Contributor(s):	**Miguel Catalan-Cid, Esteban Municio, Miguel Fuentes, David Reiss (i2CAT)** **Juan Sánchez-González, Jordi Pérez-Romero, Oriol Sallent, Anna Umbert (UPC)** **German Castellanos, Revaz Berozashvili, Simon Pryor (ACC)** **J. Xavier Salvat, Jose A. Ayala-Romero, Lanfranco Zanzi (NEC)** **Joss Armstrong (LMI)** **Keith Briggs (BT)** **Israel Koffman (REL)** **Jesús Gutiérrez (IHP)** **Mir Ghoraishi (GIGASYS)**
Work Package	**WP4**
Target Dissemination Level	**Public**

This work is supported by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101097083, BeGREEN project. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or SNS-JU. Neither the European Union nor the granting authority can be held responsible for them.

Revision History

Revision	Date	Editor / Commentator	Description
0.1	2024-03-01	Miguel Catalan-Cid (i2CAT)	Initial version and initial ToC
0.2	2024-04-02	Miguel Catalan-Cid (i2CAT), All	ToC and expected contributions closed
0.3	2024-04-23	All	Draft contributions (bullet points, sentence outline)
0.4	2024-05-15	I2CAT, LMI	Chapter 2 initial contributions (AI Engine)
0.5	2024-06-01	I2CAT, NEC, UPC	Chapter 2 contributions (SMO, O-Cloud, Relays, RIS)
0.6	2024-06-18	I2CAT, LMI, UPC, NEC, RunEL	Chapter 2 (Edge, SMO, RunEL) and Chapter 3 (Dimensionality, vRAN, Relays, Edge) contributions
0.8	2024-07-23	I2CAT, LMI, ACC, UPC	Chapter 2 revised (ACC, LMI, i2CAT), Chapter 3 contributions (i2CAT, UPC)
0.9	2024-09-03	All	Chapter 2 comments addressed (all); Chapter 3 revised (UPC, NEC, i2CAT)
0.91	2024-09-10	All	Chapter 2 revised (i2CAT), Chapter 3 comments addressed (all), Chapter 1 and 4 provided (i2CAT)
0.92	2024-09-17	Miguel Catalan-Cid (i2CAT) Jesús Gutiérrez (IHP)	Final review
1.0	2024-09-30	Mir Ghoraishi (GIGASYS) Simon Pryor (ACC)	Proof reading and submission to the EC

Table of Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project
5GC	5G Core
5QI	5G QoS Identifier
5GNR	5G New Radio
AAL	Acceleration Abstraction Layer
AI	Artificial Intelligence
AIA	AI Engine Assist
AIF	AI Functions
AIMM	AI-enabled Massive MIMO
AMF	Access and mobility Management Function
AP	Average Precision
API	Application Programming Interface
ARFCN	Absolute Radio Frequency Channel Number
ASM	Advanced Sleep Mode
ASN	Abstract Syntax Notation
AWS	Amazon Web Services
B5G	Beyond 5G
BaaS	Backend-as-a-Service
BLER	Block Error Rate
BS	Base Station
BSC	Base Station Controller
BYOD	Bring Your Own Device
CCC	Cell Configuration and Control
CHD	Coverage Hole Detection
CN	Core Network
CNN	Convolutional Neural Network
COTS	Commercial Off-The-Shelf
CP	Control Plane
CPU	Central Processing Unit
CQI	Channel Quality Indicator
CU	Central Unit
CU-CP	Central Unit Control Plane
CUPS	Control User Plane Separation
CU-UP	Central Unit User Plane
CVE	Common Vulnerability and Exposure
D2D	Device to Device
DBSCAN	Density Based Spatial Clustering of Applications with Noise
DCAF	Data Collection Application Function
DCCF	Data Collection Coordination Function
D-DQN	Decaying DQN
DL	Downlink
DME	Data Management and Exposure
DMS	Deployment Management Service
DNN	Deep Neural Network
DPD	Digital Pre-Distortion
DPDK	Data Plane Development Kit

DQN	Deep Q-Network
DRB	Data Radio Bearer
DRL	Deep Reinforcement Learning
DSCH	Downlink Shared Channel
DSS	Dynamic Spectrum Sharing
DT	Decision Tree
DU	Distributed Unit
DUT	Devices Under Test
DV	Data Volume
E2AP	E2 Application Protocol
E2SM	E2 Service Model
EC	Energy Consumption
EI	Enrichment Information
EMF	Electromagnetic Field
eNB	eNodeB
EPC	Evolved Packet Core
ET	Envelope Tracking
FaaS	Function as a Service
FAPI	Function Application Platform Interface
FCAPS	Fault, Configuration, Accounting, Performance, Security
FEC	Forward Error Correction
FDD	Frequency Division Duplex
FNN	Feedforward Neural Network
FOCOM	Federated O-Cloud Orchestration and Management
FR	Frequency Range
gNB	gNodeB
GNSS	Global Navigation Satellite System
GP	Gaussian Processes
GPP	General-Purpose Processor
GPR	Gaussian Process Regression
GPU	Graphics Processing Unit
GTP	General Packet Radio Service Tunneling Protocol
GUI	Graphic User Interface
HA	High Availability
HARQ	Hybrid Automatic Repeat request
HM	Handover Manager
HOF	Handover Failure
HPO	Horizontal Pod Autoscaling
HTTP	HyperText Transfer Protocol
HW	Hardware
IAB	Integrated Access and Backhaul
ICS	Information Coordination Service
IMS	Infrastructure Management Service
IoT	Internet of Things
IoU	Intersection over Union
IPC	Instructions Per Cycle

ISAC	Integrated Sensing and Communication
ITU	International Telecommunication Union
JSAC	Joint Sensing and Communications
KPI	Key Performance Indicator
KPM	Key Performance Metrics
L2	Layer 2
LCB	Lower Confidence Bound
LCM	Life Cycle Management
LPU	Logical Processing Unit
LSTM	Long Short-Term Memory
LTE	Long Term Evolution
MAC	Medium Access layer
mAP	Mean Average Precision
MAPE-K	Monitor-Analyze-Plan-Execute over a shared Knowledge
MCS	Modulation and Coding Scheme
MDT	Minimization of Drive Test
MEC	Multi-Access Edge Computing
MFAF	Messaging Framework Adapter Function
MISE	Memory-Interference Induced Slowdown Estimation
ML	Machine Learning
MLOps	Machine Learning Operations
MLP	Multi-Layered Perceptron
MME	Model Management and Exposure
mMIMO	Massive Multiple Input Multiple Output
mmWave	Millimetre Wave
MNO	Mobile Network Operator
MOLA-ADNA	Mobility and Load Aware proActive haNDover Algorithm
MPKI	Misses per 1000 instruction
MSE	Mean Squared Error
MVA	Mobile Video Analytics
MVNO	Mobile Virtual Network Operator
NAS	Non-Access Stratum
Near-RT	near Real-Time
NF	Network Function
NFO	Network Function Orchestrator
NFV	Network Function Virtualisation
NGAP	Next Generation Application Protocol
NGMN	Next Generation Mobile Networks
NI	Network Intelligence
NIC	Network Interface Card
NIF	Network Intelligent Function
NIO	Network Intelligence Orchestrator
NIP	Network Intelligence Plane
NIS	Network Intelligent Service
N-MAPE-K	Network MAPE-K
non-RT	Non-Real-Time
NR	New Radio

ns-3	Network Simulator 3
NSA	Non-Stand Alone
NSAP	Network and Service Automation Platform
NSSMF	Network Slice Subnet Management Function
NUC	Next Unit of Computing
NWDAF	Network Data Analytics Function
OAI	Open Air Interface
OAIC	Open AI Cellular
OAM	Operation and Maintenance
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frecuency-Division Multiple Access
O-FH	O-RAN Front-Haul
OMEC	Open Mobile Evolved Core
ONAP	Open Network Automation Platform
ONF	Open Networking Foundation
ONOS	Open Network Operating System
O-RAN	Open RAN
OS	Operating System
OSA	OAI Software Alliance
OSC	O-RAN Software Community
OTIC	Open Testing & Integration Centre
PAWR	Platforms for Advanced Wireless Research
PCI	Physical layer Cell Identifier
PDCCH	Physical Downlink Control Channel
PDCP	Packet Data Convergence Protocol
PDSCH	Physical Downlink Shared Channel
PDU	Packet Data Unit
PGW	Packet Data Network Gateway
PoE	Power over Ethernet
PHY	Physical layer
PLMN	Public Land Mobile Network
PM	Performance Management
PMD	Poll Mode Driver
PoC	Proof of Concept
PoE	Power over Ethernet
PRB	Physical Resource Block
PUCCH	Physical Uplink Control Channel
PUSCH	Physical Uplink Shared Channel
QoS	Quality of Service
RACH	Random Access Channel
RAD	Relay Activation/Deactivation
rApp	Radio access network Applications
RAN	Radio Access Network
RAT	Radio Access Technology
RB	Resource Blocks
ReLU	Rectified Linear Unit

RFM	Relay Function Management
RIC	RAN Intelligent Controller
RIS	Reconfigurable Intelligent Surface
RISA	RSI Actuator
RISC	RIS Controller
RISO	RIS Orchestrator
RL	Reinforcement Learning
RLC	Radio Link Control
RLF	Radio Link Failure
RLP	RU Load Predictor
RN	Relation Network
RNC	Radio Network Controller
RNN	Recurrent Neural Network
RNTI	Radio Network Temporary Identifier
RRC	Radio Resource Control
RRM	Radio Resource Management
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RU	Radio Unit
RUE	Relay UE
SA	Standalone
SCTP	Stream Control Transmission Protocol
SCTP-C	Stream Control Transmission Protocol Client
SCTP-S	Stream Control Transmission Protocol Server
SDAP	Service Data Adaptation Protocol
SD-CN	Software-Defined Core Network
SDK	Software Development Kit
SDR	Software Defined Radio
SD-RAN	Software-Defined Radio Access Networking
SEP	Service Enablement Platform
SHAP	Shapley Additive Explanation values
SINR	Signal to Interference and Noise Ratio
SIRA	Single Instance Resource Allocation
SISO	Single Input Single Output
SLA	Service Level Agreement
SM	Service Model
SME	Service Management and Exposure
SMF	Session Management Function
SMO	Service Management and Orchestration
SNR	Signal to Noise Ratio
SQL	Structured Query Language
SRB	Signalling Radio Bearer
SSM	Smart Surface Monitoring
SVM	Support Vector Machine
TAI	Tracking Area Identity
TCO	Total Cost of Ownership

TD	Time Difference
TDD	Time Division Duplex
TDoA	Time Difference of Arrival
TGW	Telemetry Gateway
TIP	Telecom Infra Project
TRL	Technology Readiness Level
TSDB	Time Series Database.
UE	User Equipment
UL	Uplink
ULP	UPF Load Predictor
UMAP	Uniform Manifold Approximation and Projection
UML	Unified Modeling Language
UP	User Plane
UPF	User Plane Function
vBS	Virtual Base Station
vCU	Virtual Centralized Unit
vDU	Virtual Distributed Unit
VNF	Virtual Network Function
VPN	Virtual Private Networks
VPP	Vector Packet Processor
vRAN	Virtual RAN
WLAN	Wireless Local Area Network
WP	Work Package
xApp	Cross-Functional Application
XAI	eXplainable AI
XDP	eXpress Data Path
XGBoost	Extreme Gradient Boost

Executive Summary

This document, BeGREEN D4.2 deliverable, presents the development status and the initial validation of the BeGREEN Intelligence Plane and the methods assisted by proposed Artificial Intelligence (AI) and Machine Learning (ML) to enhance the energy efficiency of the Radio Access Network (RAN) and Edge domains. This work builds on the concepts established in BeGREEN D4.1 [1], by introducing and refining the architecture, mechanisms, and use cases designed to reduce energy consumption without impairing network performance. The deliverable also reports an initial evaluation of the proposed solutions.

BeGREEN D4.2 is structured in two main chapters. The first one deals with the architecture of the BeGREEN Intelligence Plane, a cross-domain framework designed to integrate AI/ML processes within the O-RAN architecture with the objective of enhancing the decision-making process of rApps and xApps. As the main novelty, the Intelligence Plane incorporates the AI Engine, which hosts ML models and associated services, offloading them from the RAN Intelligent Controllers (RICs). By decoupling the AI/ML services from the O-RAN control loops, the BeGREEN Intelligence Plane offers a modular and reusable framework that allows for independent model development and deployment. Additionally, the deliverable discusses extensions to allow the integration of Edge and Core domains, and of RAN technologies which are currently beyond the scope of traditional O-RAN implementations like Relays, Reconfigurable Intelligent Surfaces (RIS), and Integrated Sensing and Communication (ISAC). Finally, it is examined how to manage and mitigate conflicts across the RICs between contradictory optimisation policies.

The second main chapter is dedicated to the evaluation of AI/ML-based solutions that enhance energy efficiency, including the Intelligence Plane itself. First, it presents dimensionality reduction techniques to minimize data inputs for ML models without impacting the model accuracy. This approach reduces data processing overhead and improves the energy efficiency of ML models. The compute resource allocation in virtualized RAN (vRAN) scenarios is also analysed, and Reinforcement Learning (RL) algorithms to dynamically decide on resource allocation according to network load and power consumption patterns are proposed. Another proposed strategy is the carrier on/off switching technique, which uses ML to predict traffic patterns and switch off unnecessary 5G carriers during low-demand periods, significantly reducing energy usage by offloading traffic to 4G carriers. The deliverable also details AI/ML approaches controlling fixed and UE-based relays to ensure that network resources are utilized more effectively, leading to both improved coverage and reduced energy consumption. In particular, the proposed mechanisms address coverage hole detection (CHD), fixed relay placement, candidate Relay UE (RUE) identification and dynamic relay activation and deactivation. Regarding the Edge domain, two main uses cases and methods are considered. The first one proposes a dynamic allocation of the compute resources dedicated to the User Plane Function (UPF) according to the forecasted traffic demand. The second one proposes a Bayesian online learning algorithm addressing the joint orchestration of vRANs and Edge AI services, aiming at minimizing overall power consumption while meeting the service's performance constraints.

Initial evaluations of these AI/ML-assisted solutions show promising results in terms of energy savings and performance optimization in both the RAN and Edge domain. The deliverable includes use case studies, technology characterization and experimental validations demonstrating how these algorithms can achieve substantial energy reductions in both the RAN and Edge domains. In the case of the Intelligence Plane, the reported validation assesses the baseline architecture and operations, focusing on the AI Engine and the Non-Real-Time RIC. The final evaluation of the Intelligence Plane and the AI/ML-assisted mechanisms will be reported in the upcoming BeGREEN D4.3.

1 Introduction

Despite being more energy efficient than predecessor generations, the transition to 5G has raised the energy consumption of the network due to the features required to support new advanced services. In this context, reducing energy consumption has emerged as a key challenge for network operators, both in terms of environmental impact and operational costs. As highlighted by ITU in the IMT-2030 report on future trends [2], one of the main goals of 6G will be the reduction of network-wide energy consumption. To address this problem, the O-RAN Alliance is defining control mechanisms that allow managing energy saving features in a multi-vendor O-RAN environments [3]. However, specific energy saving algorithms, are left open for vendor differentiation. Additionally, the integration of AI/ML[1] will be key to learn from historical data, proactively adapt to evolving network dynamics, and drive automated control decisions.

To address these challenges, BeGREEN proposes an Intelligence Plane, which works as a cross-domain management entity, integrating control and monitoring functions across RAN, Core and Edge domains, and fostering the creation of advanced ML models. The proposed framework incorporates an AI Engine to the O-RAN architecture. This component decouples the provision of AI/ML services, including the serving of ML models, from the Service Management and Orchestration (SMO) and the RAN Intelligent Controllers (RICs) elements. This allows independent ML model and control algorithms development, facilitating the reusability of ML models like predictors by different rApps and xApps. In addition, new interfaces and control components are incorporated to the O-RAN architecture to manage and monitor Edge and Core domains, and to support RAN technologies currently not being considered by the specification: Relays and Relay User Equipment (RUE), Reconfigurable Intelligent Surface (RIS) and Integrated Sensing and Communication (ISAC). The Intelligence Plane also includes specific BeGREEN metrics, the Energy Score and the Energy Rating, which are used to determine the absolute and relative performance of the network entities in terms of energy efficiency.

Once incorporated to the O-RAN architecture, these proposed components and technologies can be exploited by novel AI/ML-driven control mechanisms to reduce the energy consumption of the RAN and Edge domains. In the RAN, the proposed energy-saving optimisations focus on optimal compute resource allocation of virtual RANs (vRANs), dynamic 5G carrier deactivation and traffic offloading, and the utilisation of Relay and RIS technologies to enhance energy efficiency. In the Edge, resource allocation strategies are also applied to optimise the energy consumption of User Plane Function (UPF) and of Edge AI services. Additionally, methods to reduce the data dimensionality of ML models are also discussed, aiming at minimizing energy consumption without impacting model accuracy.

BeGREEN D4.2 describes the progress on the development of the proposed solutions and presents their initial validation. The deliverable is structured is as follows:

- Chapter 2 describes the architecture of the Intelligence Plane. Building on the foundations established in BeGREEN D4.1 [1], the introduced components and interfaces are further detailed and developed towards specifying the final BeGREEN architecture that will enable the proposed solutions to enhance energy efficiency at the RAN and Edge domains. Three main entities are considered:
    - AI Engine: The key element of the Intelligence Plane, providing the framework to implement and expose AI/ML services. This chapter describes how it is implemented using available open-source frameworks and integrated with the RICs through the AI Engine Assist (AIA) rApps and xApps. Additionally, the current definition and implementation of the Energy Score and Energy Rating functions is detailed.

[1] AI and ML terms denote related and overlapping concepts. In fact, ML can be seen as a subset of AI. In this document, the term AI/ML will be used to denote AI and/or ML techniques.

- SMO and non-RT RIC: Developments are focused on the integration with the AI Engine to expose ML models to rApps implementing optimisation control-loops and on the integration with the near-RT RIC through energy saving-based A1 policies. To this end, the data models of the AIA rApps and of the A1 policies are detailed.
- Near-RT RIC: Two xApps are presented. On the one hand, the Energy Saving xApp manages the operation status of the cells according to the energy saving policies. On the other hand, the Handover Manager xApp optimizes the handover process allowing to apply load balancing strategies. The join operation of both xApps will be required when applying energy saving strategies such cell on/off switching. Additionally, in this section a novel conflict mitigation and management strategy among contradictory policies and RAN control actions is discussed.

Once introduced its main architecture, Chapter 2 presents the integration of the Intelligence Plane and the RAN and Edge domains. First, it briefly presents their relationship with O-RAN O-gNB and O-Cloud, focusing on the application of O-RAN aligned energy saving strategies. Secondly, it describes the designed approach to integrate RAN technologies currently not being considered by O-RAN, like RIS, ISAC and Relays. In this case, the principal aim is to define the interfaces, components and procedures required to apply the energy saving strategies exploiting these technologies being considered in BeGREEN. Finally, Chapter 2 finalizes describing the approach to integrate the Edge domain to enable the monitoring and control of edge resources.

- Chapter 3 presents the initial evaluation of AI/ML-assisted procedures being developed in BeGREEN, including the Intelligence Plane itself. In relation to the methods introduced in BeGREEN D4.1 [1], this deliverable refines and extends the description of the solutions and of their application scenarios[2]. Then, results related to performed evaluations are presented. The chapter is structured as follows:
  - Dimensionality reduction: This method proposes a solution to systematically reduce the input data required to train and retrain models such as predictors. The objective is to enhance the energy efficiency of the models without impacting the required model accuracy. Initial results are based on a real dataset provided by a Mobile Network Operator (MNO).
  - Compute resource allocation in vRAN: Addresses the problem of compute resource allocation in virtualized RAN under shared computing infrastructure. According to an initial experimental characterization, proposes Reinforcement Learning (RL) based solution which adapts the compute resources allocated to virtual Base Stations (vBSs) according to network demands, avoiding over- and under-provisioning issues. The method is evaluated experimentally in a testbed.
  - AI/ML and data-driven strategies for energy-efficient 5G carrier on/off switching: This solution considers scenarios with capacity and coverage cells, as is the case of current 5G Non-Stand Alone (NSA) deployments, in order to propose on/off switching and traffic offloading strategies. According to the real data from a MNO, available energy saving opportunities are studied, evaluating how different heuristics and ML-driven strategies could be applied to match them.
  - AI/ML-based algorithmic solutions for relay-enhanced RAN control: Proposes the utilisation of Relays and RUEs to enhance energy efficiency of areas with high traffic demands and poor propagation conditions, avoiding the installation of new cells or the increase of cell transmission power. Several AI/ML-based methods are defined to provide a complete

[2] This deliverable does not include an evaluation of the RIS integration into O-RAN, which will be reported in D4.3.

solution, including CHD, fixed relay placement decision, identification of candidate RUs, and relay activation and deactivation. Initial evaluation is based on simulations according to a realistic scenario in a University Campus.

- Traffic-aware compute resource management to enhance UPF energy efficiency: This method deals with the dynamic management of the compute resources allocated to a UPF according to the traffic demand. According to the experimental characterization of a high-performance open-source UPF implementation, energy efficient strategies are proposed and evaluated. Proactive management of the strategies will be provided by ML models forecasting the traffic demand.
- Joint orchestration of vRANs and Edge AI services: Addresses the problem of optimizing the allocation of resources to vRANs and Edge AI services, considering the intertwined relationships and trade-offs between RAN and Edge configurations and their impact on performance. To solve it, and according to the results from an experimental characterization, an online learning algorithm formulated as a contextual bandit is proposed.
- Intelligence Plane validation: Presents the initial evaluation of the Intelligence Plane, focusing on the integration of the AI Engine and Non-RT RIC components through AIA rApps. The validation is based on the demonstration performed at the 2024 EuCNC & 6G Summit.

- Finally, Chapter 4 presents the summary and conclusions, highlighting the key findings reported in the validation section and outlining the main directions for future work, which will be detailed in the next deliverable, BeGREEN D4.3.

2 BeGREEN Intelligence Plane

This chapter is devoted to the description of the BeGREEN Intelligence Plane architecture. As was introduced in BeGREEN D4.1 [1], the Intelligence Plane aims at providing the required AI/ML control and management plane functions to the O-RAN architecture [4] with the main objective of enhancing the ability to perform RAN optimisations. In the case of BeGREEN, these capabilities are exploited to reduce the overall energy consumption of the RAN and Edge infrastructure. To this end, in addition to the Service Management and Orchestration (SMO), the Non-Real-Time RAN Intelligent Controller (Non-RT RIC) and the Near-Real-Time Intelligent Controller (Near-RT RIC), the Intelligence Plane incorporates the AI Engine, which provides a serverless execution environment hosting the AI/ML models, offering inference and training services to the rApps/xApps by following a loosely coupled approach. To enhance reusability and efficiency, the AI Engine can also host functions, such as the BeGREEN Energy Score and Energy Rating [1][5]. These functions may be used to orchestrate specific rApps or xApps, or to configure them according to the areas or components that require optimisations.

The chapter is structured as follows. Section 2.1 describes in detail the architecture of the BeGREEN Intelligence Plane, including the AI Engine and the RICs, and extending the description provided in BeGREEN D4.1 [1]. According to this architecture, Sections 2.2 and 2.3, respectively introduce the required integrations with RAN and Edge domains to provide the energy saving optimisations being proposed within the scope of the BeGREEN project. Hence, the main focus of this section is to analyse the alignment with the O-RAN specification, and, when necessary, to specify the novel components, interfaces or procedures required to develop BeGREEN proposed optimisations and to integrate them into the Intelligence Plane framework.

BeGREEN focuses on energy-efficient optimizations within the RAN and Edge infrastructure domains. Hence, the Intelligence Plane works as a cross-domain management entity, integrating control and monitoring functions across RAN and Edge domains. This integration facilitates the creation of advanced ML models that can be utilized by analytics consumers, like rApps and xApps, to implement energy-efficient automated control loops. Therefore, the architecture of the BeGREEN Intelligence Plane, as depicted in Figure 2-1, consists of the O-RAN SMO and the RICs, which are extended with additional control and management capabilities, plus the AI Engine, which hosts the ML models and implements the required AI/ML services. Figure 2-1 also illustrates how the Intelligence Plane interacts with the RAN and Edge domains, what in some cases requires of new/extended interfaces, as will be detailed in the following paragraphs.

Regarding the RAN domain, as introduced in Section 2.2, in addition to the energy-efficient management of O-Cloud and O-gNB components through O-RAN compliant components and interfaces, the Intelligence Plane aims to incorporate control over RIS, fixed relays, or Relay User Equipment (UEs with relaying capabilities), which are currently beyond the scope of O-RAN. Thus, BeGREEN proposes new interfaces or extensions termed O1+ and E2+ for monitoring and controlling these elements and integrating them within the Intelligence Plane at the non-RT and near-RT domains, respectively. Additional information on these interfaces is provided in the sections devoted to these solutions.

As depicted in Figure 2-1, the SMO incorporates Edge control functions in addition to other common O-RAN functions, such as O1 and O2 terminations. As introduced in Section 2.3, this allows the definition of optimisation strategies targeting energy efficiency, which can exploit the individual or joint management of RAN and Edge resources and the ML models available in the AI Engine. As shown in Figure 2-1, the required interface to enable the integration of SMO and Edge Resource Controller is termed as O2+, since it leverages functionalities of O-RAN's O2 interface.

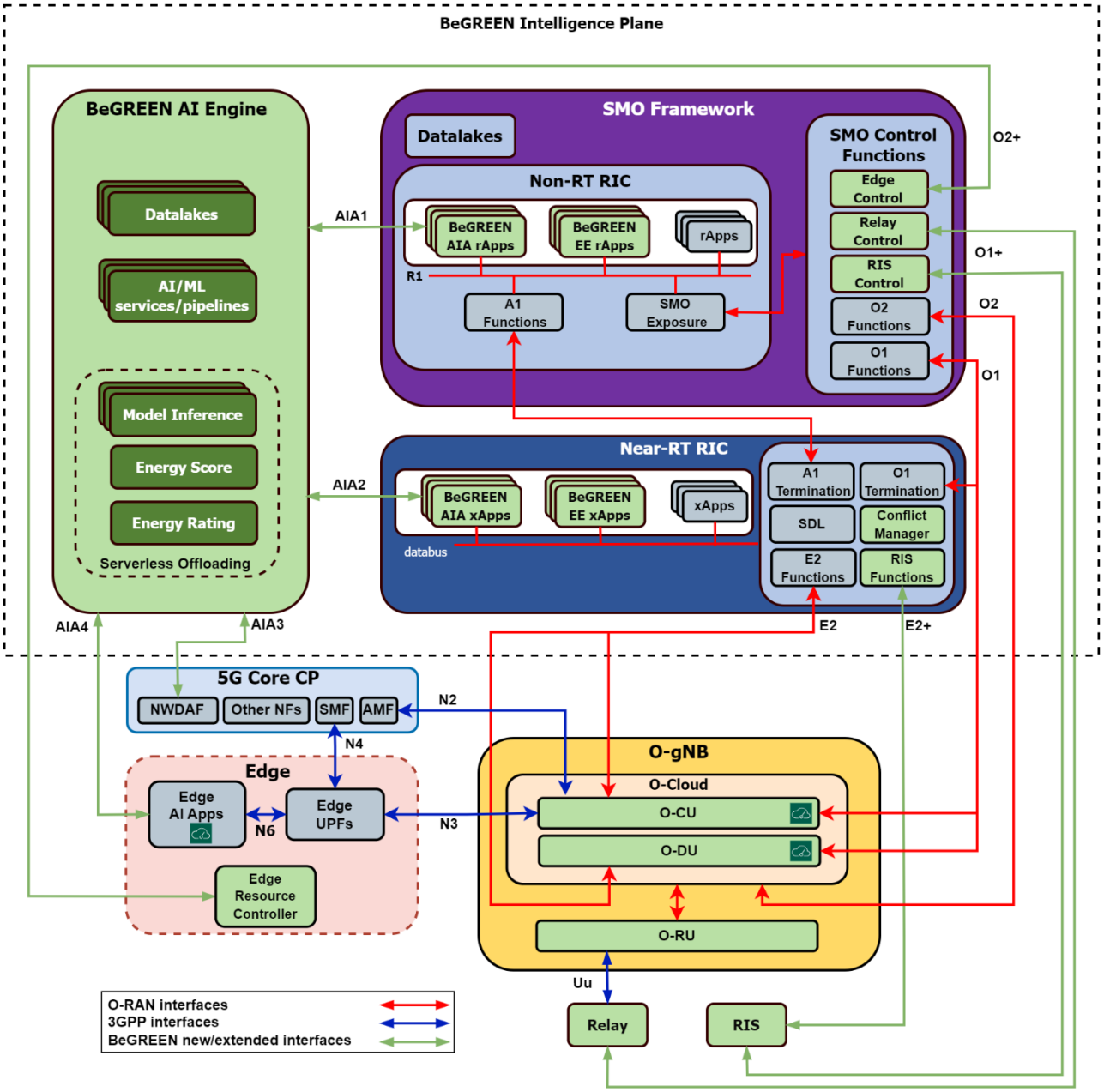


Figure 2-1: BeGREEN Intelligence Plane architecture

In the case of the SMO plus Non-RT RIC and the Near-RT RIC, BeGREEN considers two specific implementations. On the one hand, the **SMO plus the Non-RT RIC** leverage the O-RAN Software Community (OSC) implementation[3], by focusing on exposing the AI/ML services available at the AI Engine through the R1 interface, and on the management of energy-efficiency policies through the A1 interface. On the other hand, the **Near-RT RIC** is based on a commercial cloud-native solution, dRAX[4], developed by Accelleran. Note that, in addition to these frameworks, which will be integrated and used to demonstrate the Intelligence Plane in the WP4 and WP5 validations and demonstrations, other specific implementations may be used to validate technologies developed within WP4 but with a lower Technology Readiness Level (TRL).

Finally, completing the Intelligence Plane architecture, the **AI Engine** hosts ML models to offload inference tasks from the RICs and implement the necessary AI/ML workflows or services. As shown in Figure 2-1, the

[3] https://lf-o-ran-sc.atlassian.net/wiki/spaces/RICNR/overview
[4] https://accelleran.com/ran-intelligent-controller/

AI Engine implements AI/ML services or pipelines including model management, monitoring, training, serving, and a datalake with prepared data. The models are served in a serverless way, which enables efficient scaling of workloads in production. Besides ML models, the AI Engine will host other functions heavily used by rApps/xApps, such as the BeGREEN Energy Score and Rating calculations. These Key Performance Indicators (KPIs) will be used to assess the energy efficiency of the network and its components and applied optimizations, helping to identify areas or components with low efficiency and triggering the required optimizations. These models and functions are exposed to the control rApps/xApps implementing energy efficiency optimisations though AIA1 and AIA2 BeGREEN interfaces plus associated AI Engine Assist rApps/xApps, what allows to decouple the implementation of control-loops from the management of ML models. As will be detailed in the next subsection, this facilitates model reusability by different control rApps/xApps and allows the integration of AI/ML workflows through the AI Engine independent of the RICs implementation. A similar approach could be adopted for Edge and Core domains, using AIA3 and AIA4 BeGREEN interfaces to expose AI Engine AI/ML services to Edge applications or Network Data Analytics Function (NWDAF) analytics. Nevertheless, this concept is no further elaborated in BeGREEN, and these interfaces are only considered for monitoring purposes.

Next subsections detail the implementations of each of these main components.

2.1 AI Engine

The AI Engine is the key component of the Intelligence Plane, providing the framework to implement and expose AI/ML services. Figure 2-2 depicts its main components and interfaces, and how it integrates with the RICs. Conceptually, it entails three main design decisions: 1) loosely coupled approach, 2) model-based AI/ML services, and 3) serverless inference, whose characteristics are described as follows:

1) Loosely coupled approach: The models are hosted in the AI Engine and exposed to the rApps/xApps rather than being embedded in the control rApps/xApps that require their outputs. Consequently, any control rApp/xApp can access the outputs of the ML model, which are exposed as data types (e.g., offering load or energy predictions for specific cells), promoting model reuse. This solution allows ML model developers to focus on the model implementation and optimisation, while rApp/xApp developers can work on the control logic independently of how the model will be trained and served.
2) Model-based AI/ML services: As depicted in Figure 2-2, each ML model will expose its own data processing, training, monitoring and inference services. These services will be managed by dedicated rApps/xApps that, in the BeGREEN architecture, they are denoted as AI Engine Assist rApps/xApps (AIA rApp/xApp). Mainly, these AIA Apps are responsible for exposing the ML model outputs to the control Apps by communicating with the inference service of each model in the AI Engine; in the case of the Non-RT RIC the exposure will be done through the R1 interface, while in the Near-RT RIC it will be done through the message infrastructure or databus. AIA apps may also implement operations to feed the model dataset with new RAN data, to monitor or to enable the monitoring of model accuracy and drift, and to trigger model retraining. Nevertheless, pre-trained or offline trained models will be also supported by incorporating them to the AI Engine and exposing their inference service.
3) Serverless inference: BeGREEN AI Engine implements serverless inference, allowing the deployment of ML models on either servers or clusters separate from the RICs, thus enabling offloading through serverless computing and hardware acceleration. Nevertheless, in the case of near-RT inference, specific nodes could be specified to the required near-RT decision-making (e.g. the same server hosting the Near-RT RIC or collocated servers in the same edge).

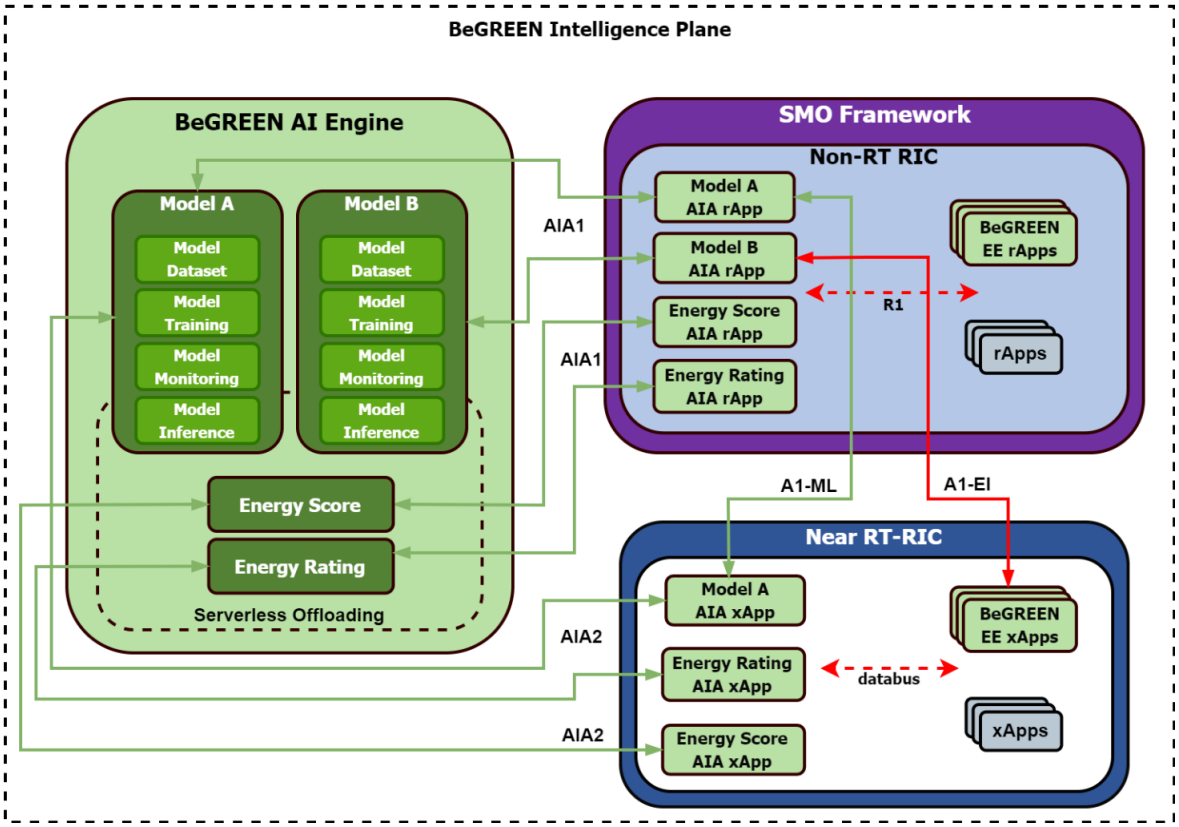


Figure 2-2: AI Engine – Architecture

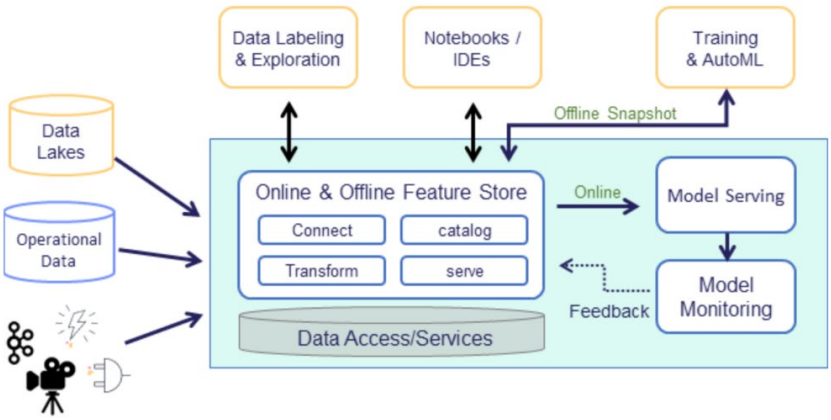


Figure 2-3: AI Engine - MLRun MLOps pipeline

According to these requirements and design decisions, the implementation of the BeGREEN AI Engine is based on the MLRun framework[5]. MLRun is designed to streamline the ML lifecycle on Kubernetes and covers the whole ML pipeline. As depicted in Figure 2-3, it includes among others: (1) data ingestion and processing, (2) model development and training, (3) model serving and (4) model monitoring. MLRun uses a MinIO[6] service as shared storage for artefacts and accesses it using the S3 protocol. MinIO is a high-performance, S3 compatible object store. It is built for large scale AI/ML, datalake and database workloads. It is software-defined and runs on any cloud or on-premises infrastructure. Additionally, MLRun also allows to incorporate pre-trained models into the serving and monitoring pipelines.

[5] https://www.mlrun.org/
[6] https://min.io/

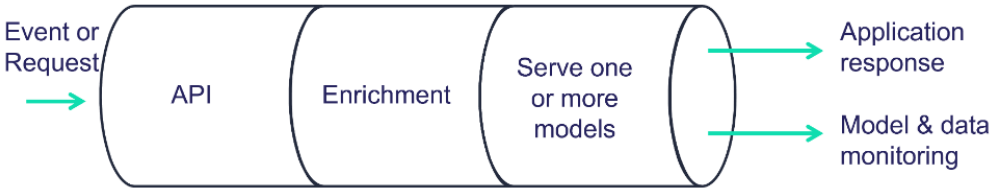


Figure 2-4: AI Engine - MLRun real-time serving pipelines

Finally, MLRun integrates Nuclio [7] for serverless model serving. Nuclio is an open-source serverless computing platform designed for high-performance applications, particularly in data processing, real-time analytics, and event-driven architectures. It abstracts away the complexities of infrastructure management, allowing developers to focus solely on writing and deploying functions or microservices. Using Nuclio, MLRun can deploy real-time pipelines which are served through an Application Programming Interface (API), as depicted in Figure 2-4. When used within a Kubernetes cluster, Nuclio can exploit Kubernetes features that allow to specify needed resources (e.g., CPU, memory and GPU)[6] and node affinity[7]. These features could be exploited to train and serve models that require hardware acceleration or a specific deployment in nodes located at the edge, such as the exposition of Near-RT inference to the xApps.

As previously introduced, the exposure of ML models hosted in the AI Engine to the RICs is done by the AI Engine Assist rApps or xApps. The objective of these Assist Apps is to decouple the management of models from the RICs and from the control rApps/xApps exploiting them, also allowing the specific handling of each specific model. For instance, pre-trained models that are deployed at the AI Engine just for inference, may not require a training and/or monitoring service to be implemented in the associated AIA rApp/xApp. Similarly, the needs of data processing could be different for each model (e.g., real-time or batch data, exposure through REST API, stream systems or databases, etc.). Therefore, BeGREEN AI Engine plus the AIA rApps/xApps allow the ML Developer to decide how to implement the required AI/ML services for a specific model with no impact on control rApps/xApps. Also, ML developers can decide which modules should be implemented in the AIA rApps and/or the xApps associated to the same model. Figure 2-5 depicts the reference model for the BeGREEN AIA xApps/rApps, which is described as follows:

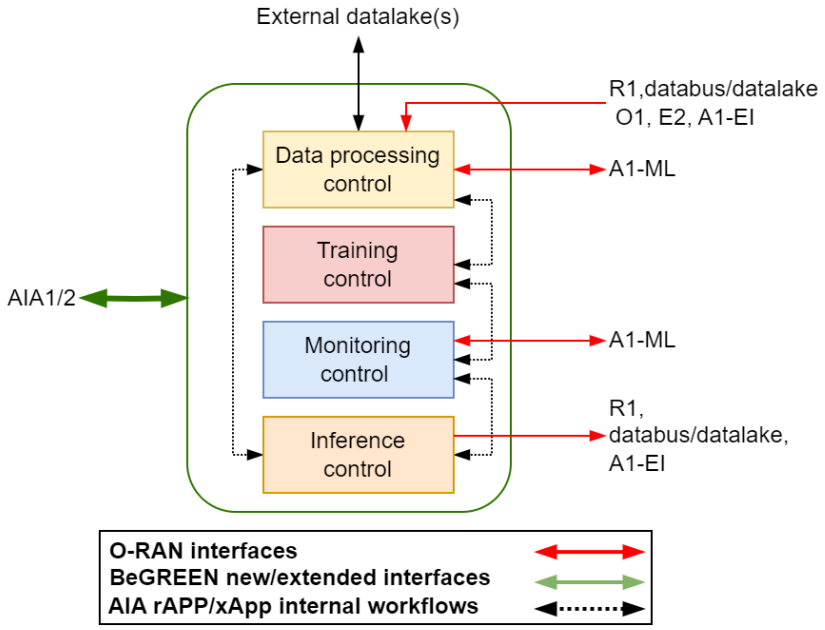


Figure 2-5: AI Engine - AIA rApps/xApps reference model

[7] https://nuclio.io/

- Data processing control: Its main function is to consume and process data from O-RAN or external domains, preparing it for training or inference services. The data, which is processed as real-time data or as batch data, can be exposed directly to the inference control module, to the AI Engine through AIA1 or AIA2 interfaces for training or monitoring, or to external datalakes (e.g., for offline training outside the AI Engine). Also, communication with the training control module may occur to manage training operations. Finally, the A1-ML interface may allow the exchange of processed data between AIA rApps and xApps of the same model.
- Training control: Triggers the (re)training of models in the AI Engine, for instance according to inputs from the monitoring control module. As aforementioned, communication with the data processing control module may occur to prepare the collection and processing of the needed data.
- Monitoring control: Its main function is to monitor the performance of the model. Monitoring may be triggered according to inference outputs or by internal logic (e.g., according to a period). A possible output may be stopping the inference and/or triggering model retraining. The A1-ML interface may be used to exchange information between the monitoring controls of AIA rApps and xApps associated to the same model. For instance, to trigger model retraining by the rApp training control module in case this module is not implemented in the xApp, or to stop the inference being performed by xApps in case the monitoring module of the rApp has additional information (e.g., obtained from different Near-RT RICs).
- Inference control: Performs inference by getting processed data from the data processing control module and triggering the model serving at the AI Engine through AIA1 or AIA2 interfaces. In the case of the Non-RT RIC, the inference results are exposed to control rApps through R1 interface (see Section 2.1.2) or to control xApps through A1-EI interface (enrichment information). In the case of the Near-RT RIC, the exposure to control xApps happens through the Near-RT RIC databus or datalake, according to the vendor implementation.

An initial validation of the AI Engine and the AIA rApps can be found in Section 3.7.

2.1.1 Energy Score and Rating

The energy scoring and rating functions in BeGREEN calculate the energy efficiency in the network at the level of any component that measures both the volume of data that it transmits and its energy consumption. The definition of energy efficiency used as the metric for energy score in the context of the BeGREEN project is taken from the Next Generation Mobile Networks (NGMN) 2015 White Paper [8], which states *"Energy efficiency is defined as the number of bits that can be transmitted per Joule of energy"*. Accordingly, the unit that energy score exposes is expressed in bits per Joule.

The energy rating is a relative measure of components of the same type in the network as regards their comparative energy efficiencies. Together, these measures can be used within the project as an indicator of which areas of the network would benefit from additional orchestration of rApps/xApps for achieving maximum energy efficiency. The following calculations are applicable to these measurements.

The Energy Score *Es* can be calculated as Data Volume *Dv* divided by Energy Consumption *Ec:*

$$\boldsymbol{Es} = \frac{\boldsymbol{Dv}}{\boldsymbol{3.6 \cdot Ec}}$$

The Data Volume can be calculated as:

$$\boldsymbol{Dv} = \boldsymbol{Dv_{DL}} + \boldsymbol{Dv_{UL}}$$

where Dv_{DL} and Dv_{UL} represent the Downlink (DL) and Uplink (UL) Data Volume.

When using a throughput counter rather than a data volume counter, it may be necessary to account for

length of period for the throughput measurement if it is measured using a time-relative measurement such as Mbps, or megabits per second.

$$\boldsymbol{Dv = (Th_{DL} + Th_{UL}) \cdot \Delta T}$$

where Th_{DL} and Th_{UL} are the average downlink and uplink throughput measured in the gNB, respectively, and ΔT is the reporting period.

Although not based on ML models, the energy score and energy rating functions will be provided within the AI Engine, making them accessible to other BeGREEN components in the architecture. Having absolute as well as relative measures of energy efficiency will aid the identification of the how much the energy saving functions of BeGREEN are contributing to the energy savings achieved in each component of the network.

Energy Score:

The energy score function in BeGREEN's AI Engine communicates with the Non-RT RIC via the AIA1 interface and with the Near-RT RIC via the AIA2 interface, as depicted Figure 2-2. The energy score function is implemented as a serverless function hosted by the Nuclio component in the MLRun framework. It is available at a HTTP endpoint. It will retrieve the energy score in bits/joule for a component on being called with the data volume and energy consumption for the relevant component.

Energy Rating:

The energy rating function is also exposed through the AIA interfaces. Implemented as a serverless function, it retrieves a relative energy rating for a network entity as a quintile (i.e., A, B, C, D or E). The energy ratings are calculated, if possible, from a dataset containing data volume and energy consumption data for all available entities of a specific type (e.g., CU, DU, Cell) in the network/slice/region by reference to recent historical data. It is recommended at least to use 24 hours of historical data in order to have a broad picture of the relative energy efficiency of the entities being compared.

The energy rating function, which is also accessible as a serverless function in Nuclio through a HTTP endpoint, stores a record of the energy ratings into the datalake of the AI Engine. This allows subsequent invocations of the function to be returned more quickly if the energy ratings for the category of component have already been calculated recently. Recalculation of the energy ratings for each component type is also possible where the data used to calculate the ratings is older than desired.

Table 2-1 corresponds to an output from the energy rating function that shows the energy score quintile for each requested cell relative to all other cells in the dataset.

Table 2-1: Energy Score and Energy Rating Examples

Cell	Data Volume (kbits)	Energy Consumption (Wh)	Energy Score (bits/J)	Energy Rating
Cell A	871446613	4960	48804	A
Cell B	890397029	9750	25367	C
Cell C	79307499	4168	5285	E

2.1.2 SMO and Non-RT RIC

The SMO and Non-RT RIC components of BeGREEN have two main functionalities, as depicted in Figure 2-6. Firstly, to host the required control rApps to create the automated non-RT control-loops, which implement and manage the cross-domain optimisations targeting energy efficiency. Secondly, to expose to these rApps AI/ML services (AI Engine), xApp policies (Near-RT RIC), and RAN and Edge services (SMO).

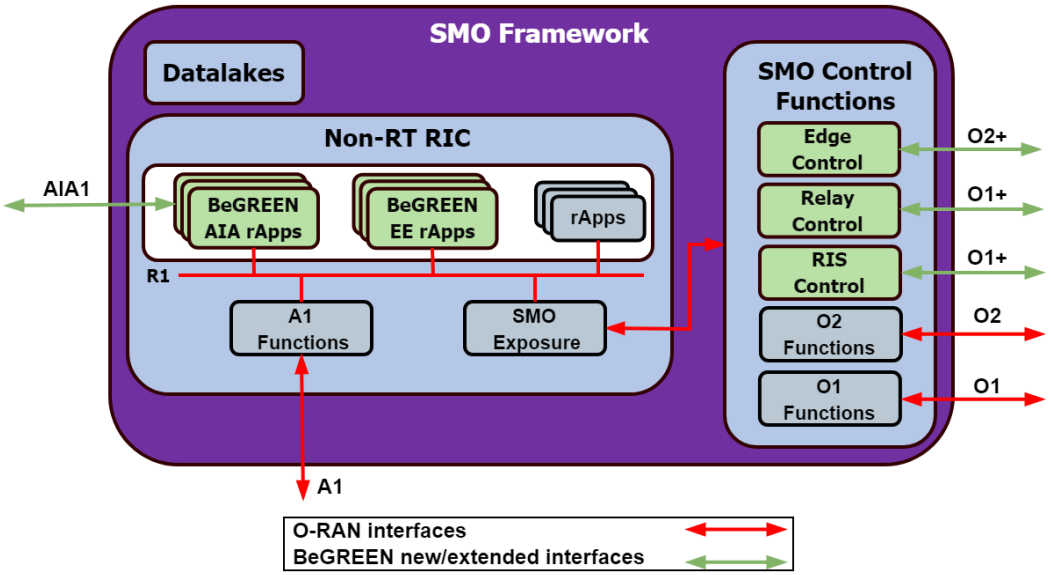


Figure 2-6: SMO and Non-RT RIC – Architecture

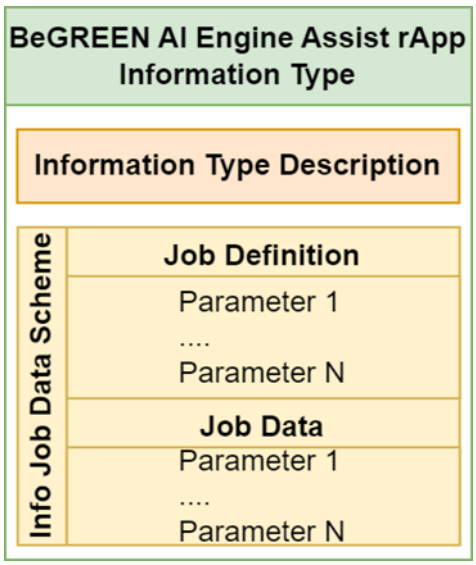


Figure 2-7: Non-RT RIC - Data model of the BeGREEN AI Engine assist rApp information types

Implementation-wise, BeGREEN is focused on the integration of the Non-RT RIC with the AI Engine and with the Near-RT RIC with the objective of validating the Intelligence Plane. Therefore, specific details about SMO functions related to O1 and O2 interfaces, and their extensions O1+ and O2+, which will have a lower TRL level, can be found in Sections 2.2 and 2.3.

As introduced in Section 2.1, the integration of the RICs and the AI Engine will be done through the AI Engine Assist rApps and xApps. While the main functionalities of the AIA rApps/xApps were previously introduced, this section will focus on of their exposure to control rApps. To this end, the Data Management and Exposure capabilities (DME) of the R1 interface are exploited [9], which simplifies the communication between data producers and data consumers. As it was presented in BeGREEN D4.1 [1], implementation-wise, BeGREEN leverages the Information Coordination Service (ICS) component, provided by the OSC [10]. This element allows the registration of specific information types, which in the case of ML models consist of the ML model outputs when doing the inference. Figure 2-7 shows the followed data model, where the Job Definition groups the required input parameters (e.g., cell ids, period) and the Job Data consists of the output parameters (e.g., cell ids, output of the model, accuracy of the prediction, etc.).

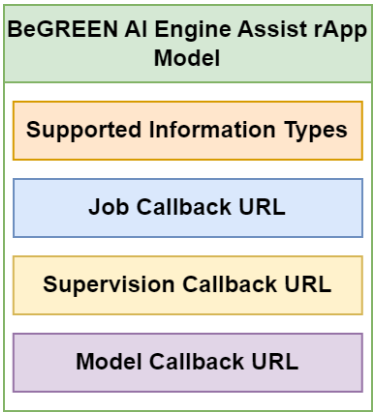


Figure 2-8: Non-RT RIC - Data model of the BeGREEN AIA rApps

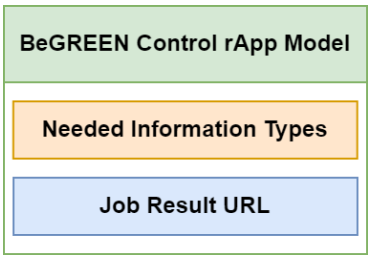


Figure 2-9: Non-RT RIC - Data model of the BeGREEN Control rApps

Once the information type is registered in the ICS through the Non-RT RIC, the deployment of the AIA rApp triggers the registration of a producer for the associated information type. The producer only needs to declare an URL for creating the subscriptions or jobs, which will follow the format defined in the information type, and another one for periodic health check from the ICS. In the case of AIA rApps, producers will be also linked to a specific ML model or function (e.g., for energy score calculation) in the AI Engine by specifying the serving URL of the model or function in Nuclio (Model Callback URL). Figure 2-8 depicts the main fields of the AIA rApp model.

The information type consumers, i.e. the control rApps, declare the requested information types and their configuration (according to the job definition), together with the URL where they expect the results to be delivered, as shown in Figure 2-9. According to this information, the ICS/R1 component searches for the needed producers and creates the required jobs or subscriptions. Finally, the producers will start sending the required data to the consumers as defined in the jobs. Additional details on the implementation of the required workflows related to the AIA rApps operations can be found in Section 3.7.

In BeGREEN, the basic interaction between the Non-RT RIC and the Near-RT RIC will happen through the A1 interface [11], mainly though A1-P to manage the policies related to energy efficiency optimisation. Additionally, other A1 functions, such as A1-EI and A1-ML, shall be implemented and validated according to the requirements of the AI Engine component, as was introduced in Section 2.1. Regarding energy saving optimisations, the last specification of O-RAN already considers the definition of specific policy and data types to manage them [12]. The main novelties of this release are:

- Energy saving policy type: Applicable to a list of Tracking Area Identities (TAIs), to a list of cells or to a specific cell.
- Energy saving policy targets: Target energy consumption (e.g., targeted RU energy consumption or energy consumption reduction).
- Energy saving resources: Defines the wanted impact on a list of cells. Used to avoid (as far as possible) or forbid impacting the operation or the coverage of specific cells while doing energy saving.

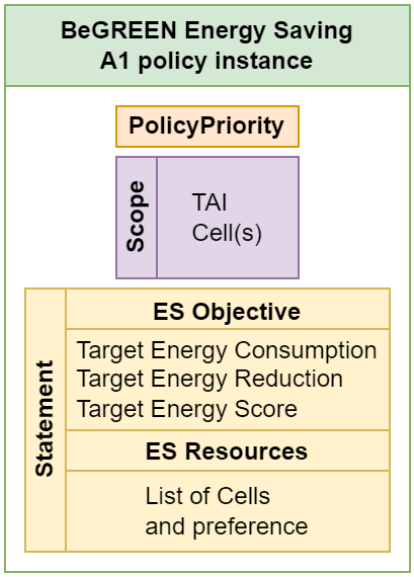


Figure 2-10: Non-RT RIC - BeGREEN Energy Saving A1 Policy

According to these fields, Energy Saving A1 policies can be enforced through the Near-RT RIC and the appropriate xApps. BeGREEN will follow this approach to define and implement the A1 policies related to the control of RUs, being compliant with the O-RAN specification. Additionally, the Energy Score (bits/J) will be also considered as a possible policy target. This way, xApps may consider energy efficiency as the optimisation objective instead of just energy consumption, increasing the options when designing xApps and their interactions with RAN functions.

Also concerning the A1 interface and its associated policies, recently the O-RAN Alliance has started addressing the management of conflicts among policies at the non-RT and near-RT levels. Several strategies related to conflict detection and management are being studied, such as implementing static and dynamic policy prioritization, or allowing partial enforcement of policies. Following the static approach, BeGREEN proposes a new field to be added in the policy instances to specify its priority. Using this priority, the near-RT will have a simple way to manage conflicts in the management of resources by xApps. More details on this and additional strategies can be found in Section 2.1.3.

According to the aforementioned fields, Figure 2-10 depicts the structure of BeGREEN's A1 policy for providing energy savings in the RUs.

2.1.3 Near-RT RIC

As described in BeGREEN D4.1 [1] and introduced in section 2 of this document, the Near-RT RIC oversees and manages the xApps of the system to provide actions over the objectives of the network. The Near-RT RIC works in a timely fashion providing solutions in the range of milliseconds (ms). To achieve this, the Near-RT RIC needs timely data from the RAN to be exposed to the xApps to support RAN control actions, while receiving A1 policies from the Non-RT RIC that guide the xApps in the actions needed to fulfil network objectives.

To this end, the dRAX Near-RT RIC from Accelleran provides a Telemetry Collector, which is designed to efficiently manage and organize telemetry data from various segments of the O-RAN infrastructure. Central to this framework is the Telemetry Gateway (TGW), which ensures interoperability across different O-RAN interfaces, such as E2, O1, F1, or A1. The TGW translates and regenerates data from the Radio Unit (RU) and Distributed Unit (DU). This way, the framework supports both fully compliant O-RAN systems and those lacking complete integration. For non-compliant systems, it translates non-standard interfaces to align with the O-RAN ecosystem. It processes raw and abstracted data from the radio environment, feeding it to the

Near-RT RIC for decision-making and RAN control. The TGW collects data from sources like the Kafka bus and publishes output metrics for use by other xApps, ensuring consistency and independence from data sources.

To validate and provide support to the BeGREEN proposal, two main xApps (described in sections 2.1.3.1 and 2.1.3.2) will be extended and included in the system evaluation and integration with the Intelligence Plane. Additionally, procedures to manage and mitigate conflicts among xApps will be implemented and evaluated. Next sections provide a detailed description of these mechanisms.

2.1.3.1 Energy Saving xApp

The Energy Saving xApp is an application developed to manage the energy consumption of various network cells. The energy-saving process relies heavily on telemetry. Telemetry involves gathering data through 3GPP supported metrics, typically sourced from the DU/RU or other equipment. However, not all required telemetry data are readily available. Therefore, an important aspect is the creation of a new metric: the energy saving percentage. This metric is defined as the ratio between the energy consumption during energy-saving periods versus that of non-saving periods.

To support this telemetry requirement, the TGW is essential. As aforementioned, the TGW is developed to integrate metrics from non-3GPP interfaces, as well as non-O-RAN compliant metrics, into the telemetry framework. This integration is crucial for applications like energy saving and Quality of Service (QoS) management within the system.

The Energy Saving xApp operates in two specific use cases:

- *Single Frequency Network:* This use case focuses on coverage optimization in scenarios where all cells operate at the same frequency. For example, as depicted in Figure 2-11, if there are six cells in an area and one cell is shut down, the remaining five cells increase their power to cover the area previously served by the shutdown cell. This approach optimizes coverage by leveraging small cells within the system.

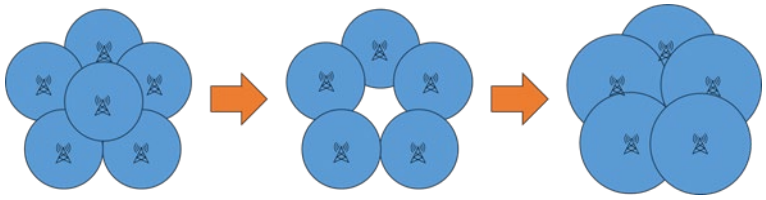


Figure 2-11: Energy Saving xApp - Single Frequency Network Use Case

- *Multi-Frequency Network System:* In this case, the network consists of two layers: a coverage layer with lower frequency cells, and a capacity layer with higher frequency cells. As depicted in Figure 2-12, when a cell in the capacity layer is shut down, users are typically handed over or forwarded to the coverage layer cells. The coverage layer cells, operating at a lower frequency, provide broader coverage, ensuring continuous service despite the shutdown of capacity cells. This is a typical scenario for MNOs.

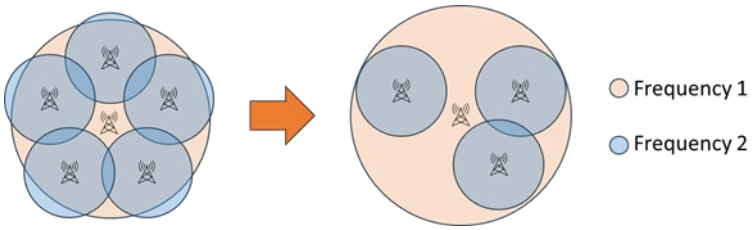


Figure 2-12: Energy Saving xApp - Multi Frequency Network Use Case.

Table 2-2: ES xApp - Required RAN metrics for the ES xApp Algorithm Decision-Making

Name	3GPP name	Unit
DL Total PRB Usage	RRU.PrbTotDl	#PRB
Number of active UE per cell	DRB.ActualActiveUe	#UE
Mean Transmission power of an NR Cell	CARR.MeanTxPwr	dBm

The first step in the Energy Saving xApp process is detecting the operational network. This requires configuring a list of network cells and identifying whether each cell belongs to the coverage layer or the capacity layer. Additionally, it involves understanding the neighbourhood relationships between cells, which can be optionally sourced from the RAN metrics. A list of UEs in the network is also necessary; if not available, this information will also be retrieved from the RAN metrics. The energy-saving algorithm and decision-making process rely on three specific metrics, which are described Table 2-2.

Apart from the metric, the xApp needs to process actions to control the RAN. The following is the list of actions and capabilities needed to provide energy savings within the Energy Saving xApp.

- Adjusting Transmission Power: When the Energy Saving xApp decides to conserve energy by turning off cells, it initiates a procedure to gradually reduce transmission power. This serves two purposes: 1) Avoiding drastic impacts on the network, and 2) allowing UEs to move from one cell to another through A3 event handovers.
- Cell Turn On and Off: When the Energy Saving xApp decides to shut down a cell, it waits until the transmission power is reduced to a minimum configurable threshold before initiating the shutdown. If UEs remain in a cell that is about to be powered down, the system triggers a handover to nearby cells—either capacity or coverage cells—depending on the received power values to ensure continuous service. Conversely, if a cell needs to be turned on to meet traffic demands, the xApp triggers the process to power up the specific cell.
- Handover: Before a cell is powered down, the system ensures that all UEs are handed over to other cells. This can either be to neighbouring capacity cells or to coverage cells, depending on the signal strength received by the users.

Regarding the xApp algorithm or control-loop, during normal network operation it measures the Physical Resource Blocks (PRBs), throughput and power usage. If a cell's PRB usage falls below a configurable threshold, the cell is marked for shutdown. The power is gradually reduced, prompting UEs to handover to other cells before the cell is shut down. If the throughput of other cells exceeds a certain threshold, this triggers the cell turn-off process. When the algorithm determines that a cell needs to be turned on, it is powered up and users can handover back to this cell, thereby increasing the network's capacity.

An important feature of the Energy Saving xApp is its capability to use cell load predictions from other xApps to optimize network performance. The cell load prediction xApp collects traffic information and uses AI/ML algorithms to predict load usage. This enables the Energy Saving xApp to foresee which cells will need to be turned on or off in the future, providing more accurate and efficient network management. In the case of the BeGREEN project, as described in section 2.1, this prediction feature will be provided by the models hosted in the AI Engine, rather than internally in the xApps.

The global interaction between different components in the Energy Saving xApp is crucial for its functionality. When information comes from the E2 node, it enters an E2 broker. This information is then passed to the dRAX databus, which connects to the TGW. The TGW interconnects information with different parts of the system, including the radio or a simulated radio like a RAN emulator or RIC Tester, as shown in Figure 2-13.

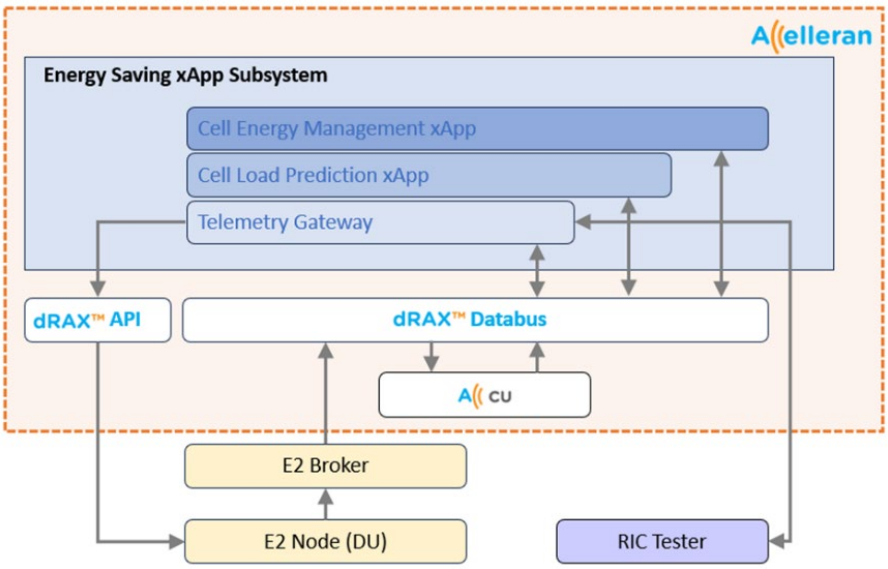


Figure 2-13: Energy Saving xApp - Global interactions for the Energy Saving xApp

Figure 2-14: Energy Saving xApp - Accelleran's dashboard to evaluate the performance of Energy Saving xApp

The Energy Saving xApp includes an analysis dashboard to visualize the effects of decision-making on users. This dashboard provides insights into energy savings achieved and offers a comprehensive analysis of the network, summarizing the RAN matrices. As shown in Figure 2-14, in some scenarios the Energy Saving xApp is able to provide savings larger than 20%, maintaining a QoS above 90%.

The extension of this xApp within the BeGREEN project will include the integration with the Intelligence Plane for traffic prediction and the conflict management capabilities. This will be reported in the next deliverables of WP4 and WP5.

2.1.3.2 Handover Manager xApp

The Handover Manager xApp is designed to optimize the handover process in 3GPP systems, offering a proactive approach to managing network resources efficiently. Traditional handover algorithms, such as the A3 algorithm, are reactive and focus on per-device optimization. In contrast, the Smart Handover-xApp employs advanced algorithms like Mobility and Load Aware proActive haNDover Algorithm (MOLA-ADNA) to achieve global optimization [13], ensuring efficient load distribution and improved overall network performance. This adaptive and QoS aware handover algorithm takes multiple metrics into account to do

network level optimization of the cell load. The Handover Manager xApp is developed on the dRAX framework, where all communication happens through the dRAX databus, both for receiving messages and sending commands. The architecture of the Handover Manager (HM) xApp, shown in Figure 2-15, includes the following components:

- Data Retriever: Retrieves messages from the dRAX databus, including Reference Signal Received Power (RSRP) and Reference Signal Received Quality (RSRQ) measurements of each UE to their respective serving and neighbouring cells, as well as the throughput of each UE.
- Data Processor: Processes incoming messages and prepares information to be stored in the internal Data Store.
- Data Store: Holds the current network overview, including each UE's serving cell, throughput, and a list of all neighbouring cells, along with their RSRP and RSRQ values and the history of each metric.
- Handover Algorithms Database: Contains different handover algorithms, including a processor for calculating additional metrics and the logic for each algorithm.
- Handover Execution Engine: Executes the handover algorithm processor, updates metrics in the Data Store, and executes the handover algorithm logic to generate a handover list.
- Action Taker: Generates appropriate handover commands based on the handover list and sends these commands to the dRAX databus for execution.

The Handover Manager xApp continuously monitors data from the RAN in the RIC, providing a global overview of the network that allows for proactive optimizations. Configuration options, such as selecting a handover algorithm, setting the periodic interval of handover execution, and the length of data history for metrics, can be adjusted in real-time.

To facilitate smart handovers, the Handover Manager xApp collects and processes various metrics through data exposure to the dRAX data bus. Key metrics include:

- RSRP and RSRQ per UE to the serving cell.
- RSRP and RSRQ per UE to all neighbouring cells.
- Downlink and uplink throughputs per UE.

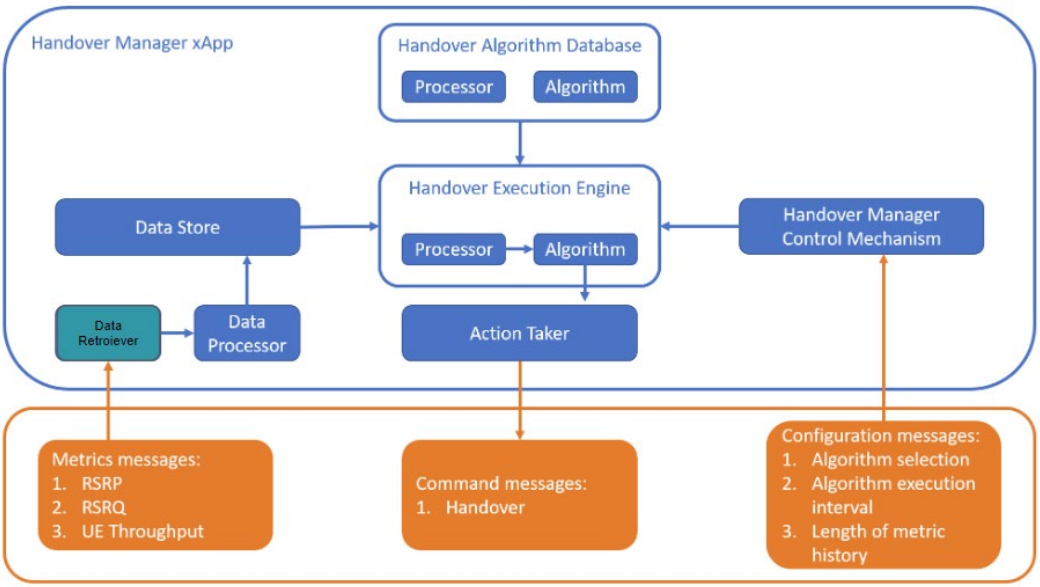


Figure 2-15: Handover Manager xApp - Internal Architecture

This information flows from the cells to the data bus and then to the xApp for analysis. Commands from the xApp back to the data bus and cells include automatically triggered handover commands, specifying the selected cell for each handover.

As aforementioned, the Handover Manager xApp is capable of working with two particular algorithms: A3 and MOLA-ADNA algorithms. The typical handover in 3GPP systems uses the A3 handover algorithm, a reactive approach. It triggers when the received power of UE crosses a certain threshold, with cell selection typically going to the one with the highest received power. The objective is per-device optimization, connecting to the neighbour cell with the highest signal indicator plus an offset over a period of time.

The MOLA-ADNA handover algorithm used by the Handover Manager xApp is proactive, meaning it actively monitors the network via the RIC and initiates handovers before issues arise. It selects cells based on multiple metrics, including RSRP, RSRQ, UE throughput and cell load. The algorithm aims for global optimization, distributing the network load across all cells. Using regression based on RSRP and RSRQ, it predicts the direction of UE movements, optimizing traffic distribution. The process involves: (1) Collecting required metrics from the data bus, (2) estimating the direction of UE movement and cell load, and (3) implementing multi-criteria decision-making to generate a handover list.

Several experiments were conducted to evaluate the proposed MOLA-ADNA handover algorithm, comparing it to the commonly used A3 handover algorithm [13]. Using dRAX with three small cells and three general-purpose UEs in an industrial setup, the experiments monitored throughput and Block Error Rate (BLER) QoS parameters. The setup included 20 MHz Time Division Duplex (TDD) small cells operating on band 42 and Raspberry Pis with band 42 LTE modems as UEs. Two static UEs were attached to cell Production-1, while a mobile UE started at cell Warehouse and moved towards cells Production-1 and Production-2. Static UEs generated 50 Mbps each, and the mobile UE generated 20 Mbps.

Key findings showed that MOLA-ADNA significantly improved UE throughput during movement, with the mobile UE's mean throughput increasing by 25% [13]. It also maintained static UEs' throughput more effectively and reduced the mean BLER by 65%, enhancing communication reliability. MOLA-ADNA's proactive optimization triggered handovers before QoS parameters degraded, unlike the reactive A3 algorithm as shown in Figure 2-16. Additionally, MOLA-ADNA demonstrated scalability, optimizing networks with up to 750 UEs within 1.2 seconds. Overall, the real-life experimentation highlighted the clear advantages of the MOLA-ADNA handover algorithm over the traditional A3 algorithm, emphasizing its proactive, multi-metric approach to network optimization. The enhancements for the BeGREEN project will include the extensions towards Geo-localization based on external information, e.g. ISAC, and extensions to support conflict management. This will be reported in the next project deliverables.

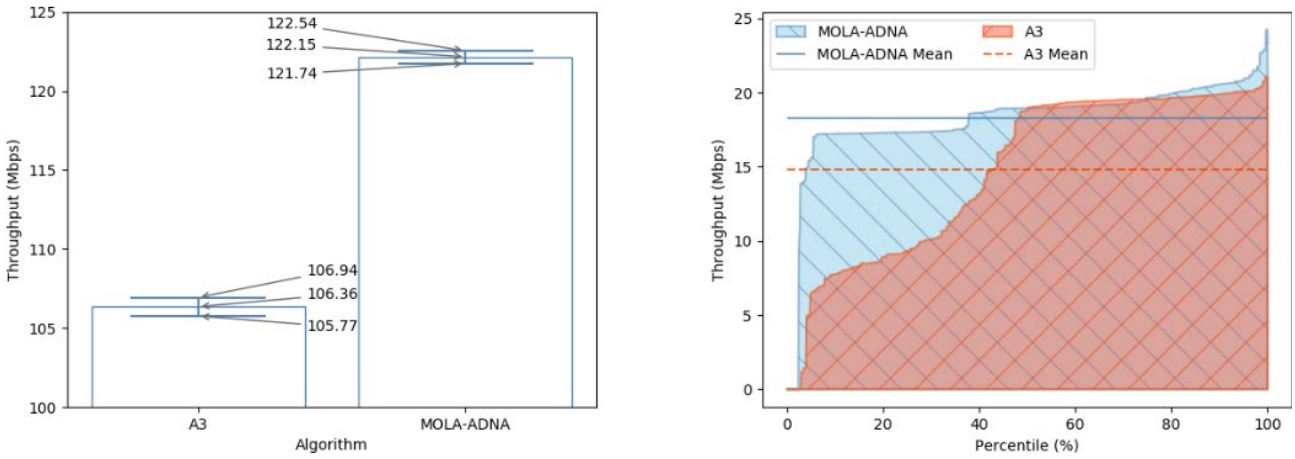


Figure 2-16: Handover Manager xApp - Preliminary MOLA-ADNA Smart Handover algorithm results.

2.1.3.3 Conflict Mitigation

Conflict mitigation involves managing situations where two applications or entities perform conflicting actions. In the context of O-RAN, conflicts occur when two actions conflict within RAN components. To manage these conflicts, policies should be considered first. Policies are rules used to control changes in the state of managed components, providing guidelines for network actions. However, these policies can sometimes create conflicting actions. Strategies to minimize the impact of these conflicts can be categorized into three main strategies: conflict detection, conflict resolution, and conflict avoidance.

- Conflict detection involve pre and post actions to identify potential and actual conflicts, whether direct, indirect, or implicit. Indirect conflicts pose a challenge as the platform may detect potential conflicts but determining whether these conflicts are harmful is not straightforward. Blocking potential conflicts involving many use cases is not a viable solution. Detecting indirect or implicit conflicts, especially with more than two conflicting applications, is a highly complex task.
- Conflict resolution, on the other hand, is typically a post-action process. Resolving ongoing conflicts is not trivial because different resolutions may harm the network or fail to meet the system's objectives. This resolution process requires careful consideration to ensure that the chosen resolution does not introduce new problems or exacerbate existing ones.
- Conflict avoidance or guidance is a pre-action strategy aimed at detecting conflicts based on previous experiences. This approach uses mechanisms like E2 guidance to avoid future conflicts, leveraging the fact that xApps often exhibit repeated behaviour. By providing guidance and information to xApps, the network can prevent many conflicts before they occur.

Regarding direct conflicts, the RIC may not always be able to decode E2SM level information. Extending E2-related API to enable the platform to use xApps for E2SM-specific processing would offer one solution to address this issue. E2SM level processing can lead to delays between the E2-related API request and the E2AP RIC Control Request message to the E2 Node. Since xApps are likely to repeat the same or similar requests for the same E2 Node, repeating conflict mitigation processes at each occurrence is wasteful and time-consuming. A longer-term guidance solution giving xApps a priori permission would avoid this issue. On the other hand, indirect and implicit conflicts are generally only detectable as post-action, after corresponding E2AP transactions have been completed. The RIC may be able to observe E2 Node KPIs to detect the impact of indirect conflicts, leading to the need for platform-initiated requests, either directly to the E2 Node or to trigger xApps to initiate data collection. This can be done using E2SM-KPM and/or E2SM-CCC. However, the analysis of collected data and messages may be difficult if the platform cannot decode E2SM level information.

In the BeGREEN project, conflict mitigation spans several scopes, including SMO, Non-Real-Time RIC, which involves rApps, Near Real-Time RIC, involving xApps, and interfaces, including the A1 and E2 interfaces. In the following points the current work in the O-RAN Alliance regarding conflict mitigation is detailed and the BeGREEN approach is described.

Current Status in O-RAN:

In the current status of O-RAN, Work Group 2 (WG2) focuses on conflicts related to A1 policies, though this topic is still not being addressed in the current A1 specification. The main scenario is related to a set of A1 policies to be requested at the Non-RT RIC, where their targets may not be predictably achievable due to the potential contradictory state of the RAN once these policies are enforced. Currently, three main conflict types are being studied: 1) Objective conflict, related to policy types with contradictory objectives (e.g. QoS and Energy Saving), 2) Resource conflict, which considers policies leading to inconsistent utilisation of RAN resources, and 3) Scope conflict, which is caused by contradictory policies targeting related scopes (e.g., Slice QoS vs UE QoS).

In the O-RAN Work Group 3 (WG3), conflict mitigation is considered in four different areas of the Near-RT RIC architecture specification [14]. The first area is conflict mitigation found in section 6.2.3 of the specification, which defines the role of the Near-RT RIC platform in conflict mitigation. Its functionality involves detecting and resolving potentially overlapping or conflicting requests from multiple xApps. Conflict mitigation also addresses interactions between different xApps, as their objectives may result in conflicting actions. However, the specifics are not fully defined, and further work in WG3 on conflict mitigation is ongoing. The second area is the A1 policy procedure, described in section 9.2.2 of [14], which outlines the procedures for setting up, updating, deleting, and querying policies. The third area is the definition of the policy management service, which is related to the A1 API procedures and is detailed in section 9.2.1.1 of [14]. And the fourth area is the E2 guidance API-related procedures, found in section 9.3.3, which defines the communication between xApps to avoid conflicts. It allows authorized xApps to obtain guidance from the Near-RT RIC platform's conflict mitigation functionality before initiating an action. The platform may signal conflicts or provide guidance to internal processes and other xApps.

Consequently, regarding the conflict management in the RICs, two conflict scopes can be considered. The first one may happen in the context of A1 policy conflicts, that typically arise when two different policies affect the same RAN element (e.g., cell) with contradictory objectives (e.g. RAN resource reservation and release). Such conflicts are ideally resolved in the Non-RT RIC but can also be managed in the Near-RT RIC. The second conflict scope may occur when different xApps generate contradicting actions over the RAN elements. Such conflicts should be managed in the Near-RT RIC. For instance, if several control messages are related to UEs (e.g., related to handover, carrier aggregation or dual connectivity control), it is possible that only the first message to arrive at the E2 Node will succeed, which can lead to a latency-sensitive critical path issue (direct control conflict). This aspect is being addressed by WG3. The conflict mitigation function in the platform could observe E2-related API messages from xApps but may not detect potential indirect conflicts between the two requests, highlighting the complexity of indirect xApp conflicts.

dRAX Near-RT RIC implementation:

To support general conflict management in the RIC, several modifications are needed for the dRAX framework, as shown in Figure 2-17. The first modification is that the Non-RT RIC needs to define an A1 policy manager to send policies to the Near-RT RIC. The second modification involves extending the Near-RT RIC to support several entities, which can be divided into three specific areas:

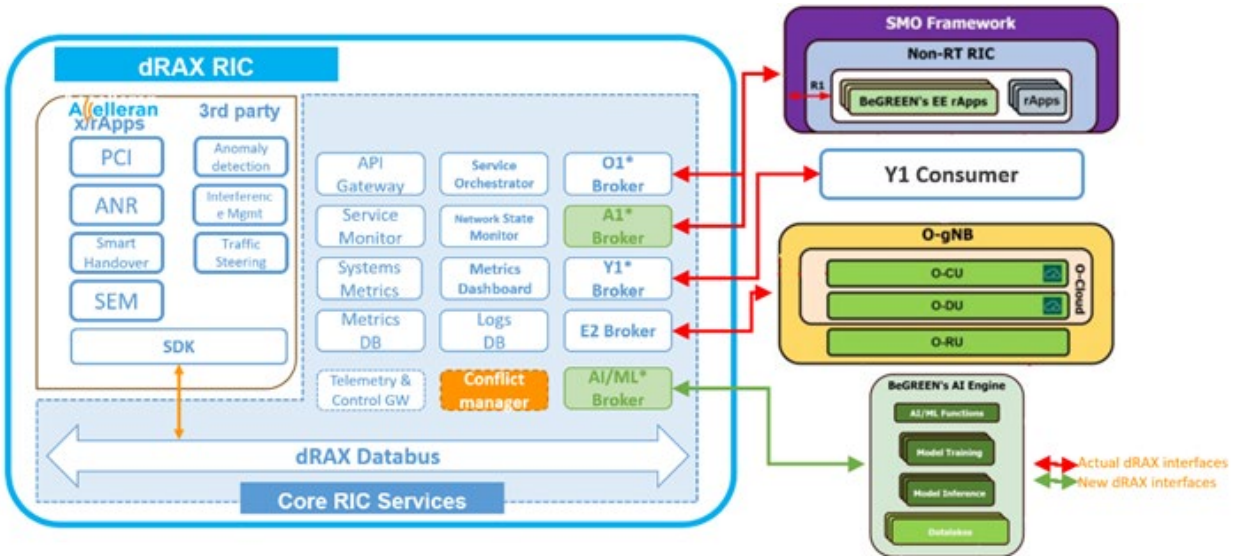


Figure 2-17: Conflict Mitigation - dRAX architecture to support conflict mitigation.

- **Subscription Manager:** This interface will receive information from the A1 policies, handle the policies, and direct them to all the xApps in the system. The Subscription Manager is a subscription

interface that will be part of the SDK of the dRAX. It serves as the interface between the dRAX and the subscription manager entity in the Non-RT RIC. The Subscription Manager is responsible for translating and managing A1 policy messages, which will be derived from future release of the A1 policy by the WG3 in O-RAN (Figure 2-18).

- **Conflict Manager Entity:** This entity will manage all the conflict processes within the Near-RT RIC. It is divided into three important components: 1) the xApps Subscription Manager, the 2) Subscription Database, and 3) the Conflict Mitigation, Detection, Resolution, and Avoidance Entity.
    - The xApps Subscription Manager handles the subscription of xApps to assist in the conflict process. It is responsible for collecting and managing the subscription database, managing and inspecting the E2SM level information, and overseeing the onboarding, securing, authorizing, and conflict authorization of the xApps.
    - The Subscription Database plays a crucial role in conflict mitigation by maintaining information about RAN elements and their relationships with policies and actions. It serves as the front-facing API for the GUI dashboard, enabling conflict mitigation resolution through exclusive or priority-based techniques. This database, or matrix, helps identify potential conflicts by defining actions for each element based on the type of element and policy type. It also provides a trace of the status of each RAN element, allowing the control manager to monitor and manage conflicts effectively.
    - The Conflict Mitigation, Detection, Resolution, and Avoidance Entity (Figure 2-19) is tasked with detecting, resolving, and providing guidance to avoid conflicts between xApps. It handles policies from the A1 interface and distributes them among the xApps. Additionally, it defines mechanisms to manage conflicts, which could be exclusive, or priority based.
- **Conflict Avoidance Handler:** The conflict avoidance handler is an entity that resides within each xApp to respond to policies and conflicts within the system. The xApp Conflict Manager is responsible for creating communication channels between all xApps to prevent conflicts. Each xApp needs a handler dedicated to specific conflict avoidance tasks. The dRAX Near-RT RIC will provide mechanisms for xApps to publish and receive alerts, particularly through the O1 VEST alert. It is up to the xApp developers to decide which types of alerts to publish and which to react to. This marks the first step towards collaborative conflict avoidance. Additionally, this mechanism can be used by rApps to manage A1 policy conflicts and distribute responsibilities via the O1 interface.

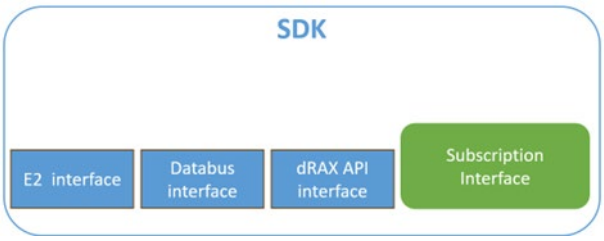


Figure 2-18: Conflict Mitigation - dRAX Subscription Manager

Figure 2-19: Conflict Mitigation - dRAX Conflict manager entity

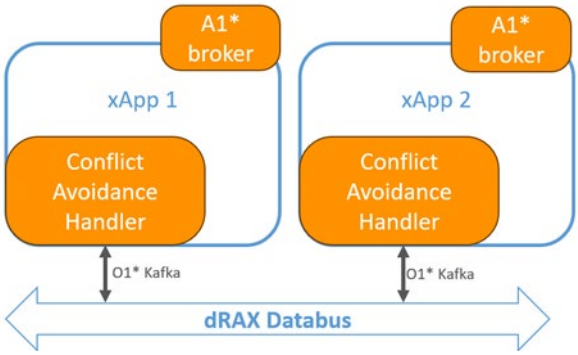


Figure 2-20: Conflict Mitigation - dRAX conflict avoidance handler inside xApps

Conflict Management collaborative solution:

Figure 2-21 presents the general implementation of the collaborative solution for the RIC, describing the elements and entities involved in conflict management. In the upper part, the SMO holds the Non-RT RIC, where rApps work together. Below that, there is the dRAX Near-RT RIC and, at the bottom, the RAN elements are depicted. To support conflict mitigation several modifications to the infrastructure are made. First, the A1 policy management resides inside the Non-RT RIC. This component sends A1 policies to the Near-RT RIC. These policies are then distributed to all xApps and are listened to by the policy listener or A1 broker within each xApp. As each xApp performs its tasks, any action an xApp intends to initiate is sent through the conflict avoidance handler to the dRAX databus. The conflict avoidance handler detects if another xApp is likely to issue a conflicting action. In this collaborative approach, if two xApps attempt to use the same resource, they step back and resolve the conflict using information from the dRAX databus based on information from the policy. Finally, the xApp conflict management stage is used to detect potential conflicts throughout the system, ensuring proactive conflict detection and resolution.

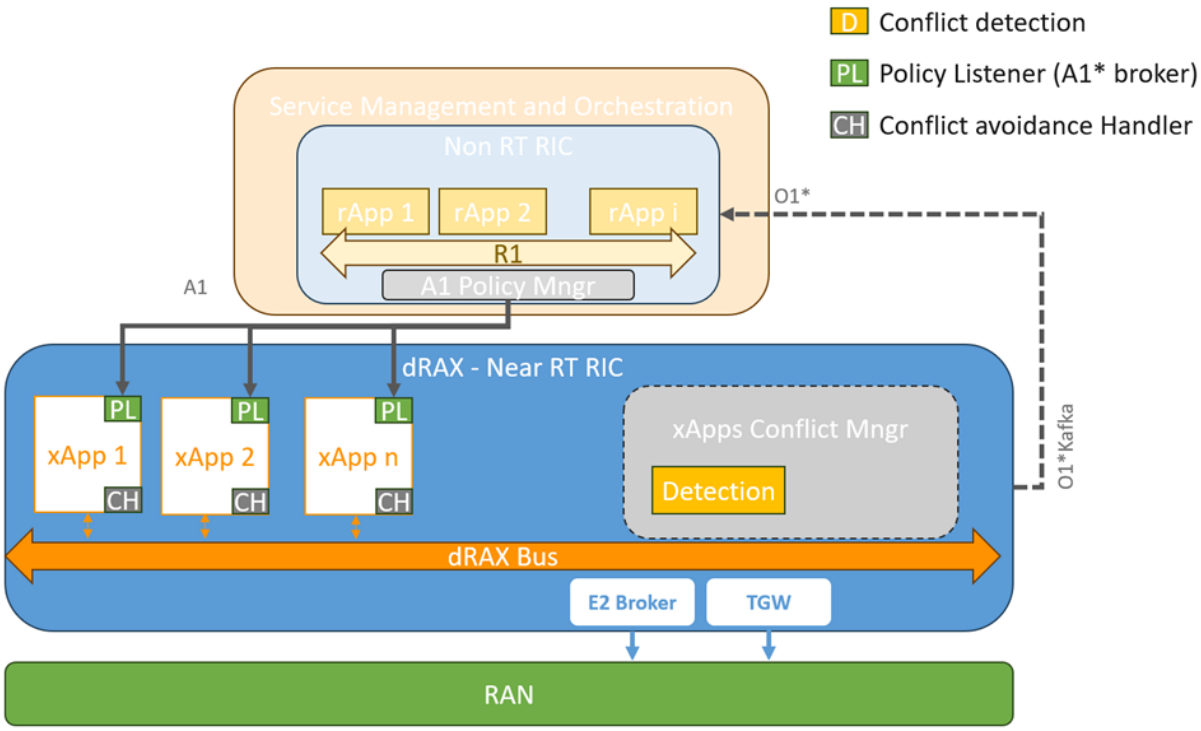


Figure 2-21: Conflict Mitigation - conflict management solution proposed by BeGREEN

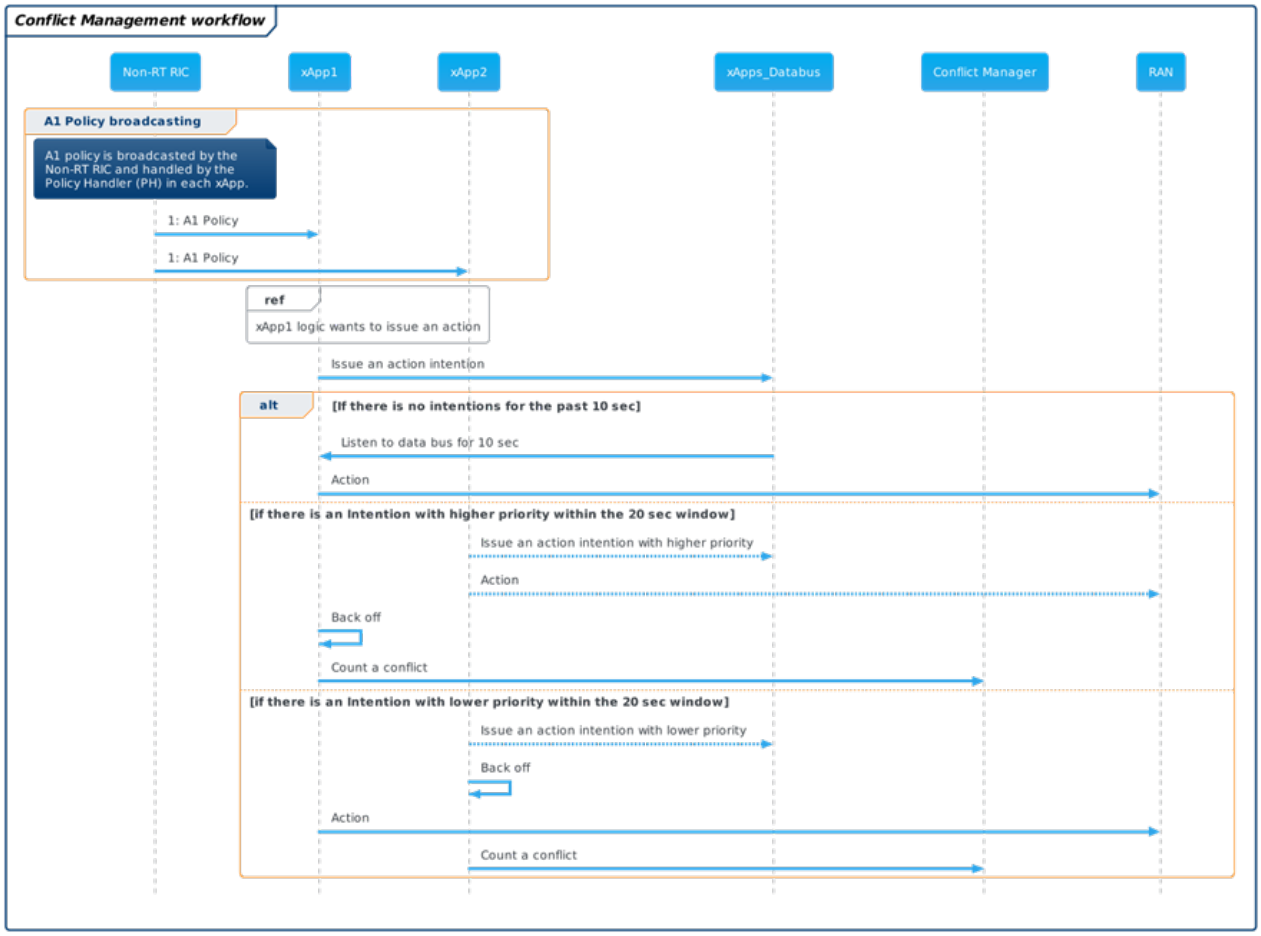


Figure 2-22: Conflict Mitigation - message workflow for collaborative conflict mitigation

The message workflow for the collaborative conflict mitigation is described in the Unified Modeling Language (UML) diagram depicted in Figure 2-21. The validation of this workflow considering scenarios involving conflicting xApps devoted to QoS and Energy Savings, will be reported in following deliverables.

2.2 Integration with BeGREEN RAN domain

This section details how the BeGREEN Intelligence Plane integrates with O-Cloud, O-gNB, RIS, ISAC and Relay technologies according to the control and monitoring needs of the proposed energy efficiency optimisations. Note that, as reported in BeGREEN D4.1 [1], in the BeGREEN project RIS, Relay and ISAC technologies are considered, which are currently beyond the scope of O-RAN. Therefore, in addition to O-RAN components, interfaces and procedures related to O-Cloud and O-gNB control and monitoring, whose utilisation within the context of energy saving uses cases was already introduced in [3], the Intelligence Plane requires of novel approaches to integrate these other technologies.

2.2.15G Base Station / O-gNB

BeGREEN research in this area was distributed in several WPs. In WP3 the research was concentrated in a specific module that turns off the RU RF Power Amplifier (PA) when no data is transmitted in the Orthogonal Frequency-Division Multiple Access (OFDMA) downlink symbol (Power Amplifier Blanking Module). In WP4 the research was focused on two AI based modules: a) The Digital Pre-Distortion (DPD), and b) the Envelope Tracking (ET) modules. For the convenience of the reader, both activities in WP3 and WP4 related to the BS Energy Consumption saving research have been reported in WP3 deliverables (e.g., BeGREEN D3.2 Section 3.2 [15]).

These solutions do not require of a Non-RT RIC or Near-RT RIC control, since decisions are taken on a real-time basis by the RU module. Nevertheless, BeGREEN WP4 is developing other strategies that will require of control-loops to optimise the energy efficiency of O-gNBs and will exploit O-RAN interfaces and components (e.g. Energy Saving xApp introduced in 2.1.3.1 and non-RT RU control described in Section 3.3). According to current O-RAN specifications, RU energy saving functionalities shall be managed by the DU component through the M-plane [16] including carrier deactivation, RF channel switch off/on, advanced sleep modes and deep-hibernate. Then, rApps may have access to these functionalities through the O1 interface [17], while xApps may use the Cell Configuration and Control (CCC) Service Model of the E2 interface (E2SM) [18]. Therefore, the proposed solutions in BeGREEN will leverage these interfaces.

Given the ISAC latency and bandwidth requirements when processing the signals at the RU, to date O-RAN does not put much emphasis on its implementation and it is not part of O-RAN Alliance focus. Therefore, no SM has been standardised for gathering data directly from the PHY layer. If that were the case, i.e. that the I/Q samples available at the RU are to be processed, it is wise to consider that any pre-processing of the signals needs to happen at early stages to reduce the datarate towards the core network (CN). This pre-processing could take place at the O-RUs, O-DUs or, in case of a centralized deployment, either at the O-CU or at the Near-RT RIC. These two last options could be the best solution when processing jointly data from different sources, i.e. different O-RUs.

BeGREEN project will study the integration of ISAC developments in the project in a tighter manner than described in the Description of Work, seeking to provide sensing information to the O-RAN Intelligence Plane, subject to the features of the considered O-RUs. These contributions will be reported in the next project deliverables of WP2 (architectural concept), WP3 (ISAC development work) and WP4 (implications on the Intelligence Plane and RAN domain integration).

2.2.2 O-Cloud

As introduced in [19], efficient CPU energy management is essential for optimizing the performance and sustainability of O-Cloud nodes in the O-RAN architecture. This section outlines the key aspects of CPU energy saving modes, data retrieval processes, and the roles of various components and interfaces within the O-RAN ecosystem.

In the O-RAN O-Cloud architecture, depicted in Figure 2-23, telemetry and inventory management is handled by the O2 interface, which is responsible for retrieving data such as supported CPU energy saving modes, CPU utilization, current operational CPU power, frequency, and voltage from the O-Cloud nodes. Additionally, RAN data, including user traffic load and RAN configurations related to network functions (O-DU & O-CU), is sourced from the O1 interface. The O2 interface enables changes in network energy configurations, allowing for dynamic adjustments to optimize energy consumption based on real-time data.

Several components are involved in this process. The Federated O-Cloud Orchestration and Management (FOCOM) manages the orchestration and lifecycle of O-Cloud resources, ensuring efficient allocation and operation. Infrastructure Management Services (IMSs) provide a logical interface for orchestrating the O-Cloud lifecycle processes, including the management of network functions and other operational procedures. Deployment Management Services (DMS) manage the lifecycle of deployments that utilize cloud resources, ensuring smooth and efficient deployment and operation of network functions.

The O2 interface is a crucial element in the O-RAN architecture, providing secure communication between the SMO and the O-Cloud. It facilitates the management of O-Cloud infrastructures and the deployment lifecycle of O-RAN cloudified network functions. Its extensible design allows for the addition of new information or functions without needing protocol or procedure changes, supporting a multi-vendor environment independent of specific implementations of the SMO and O-Cloud.

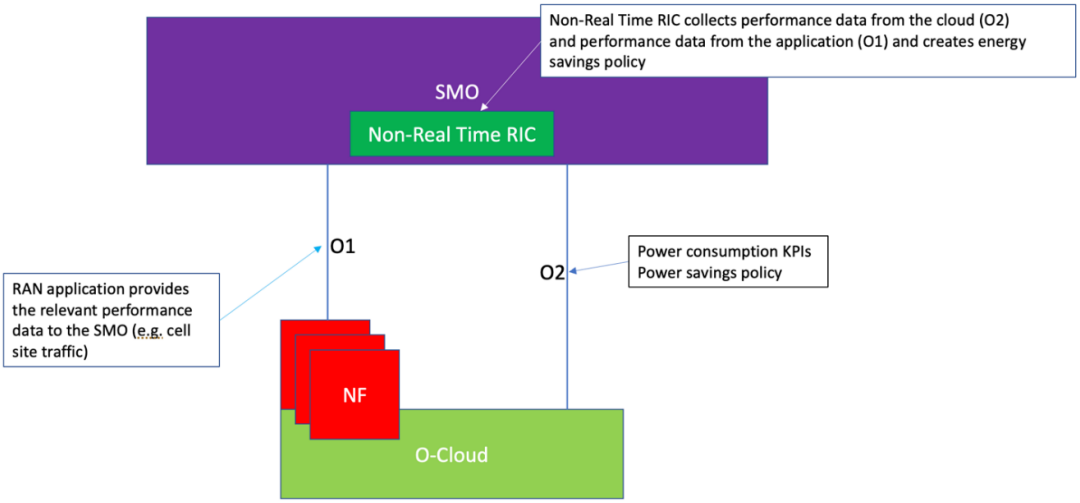


Figure 2-23: O-Cloud Energy Savings – Interfaces and operations [19]

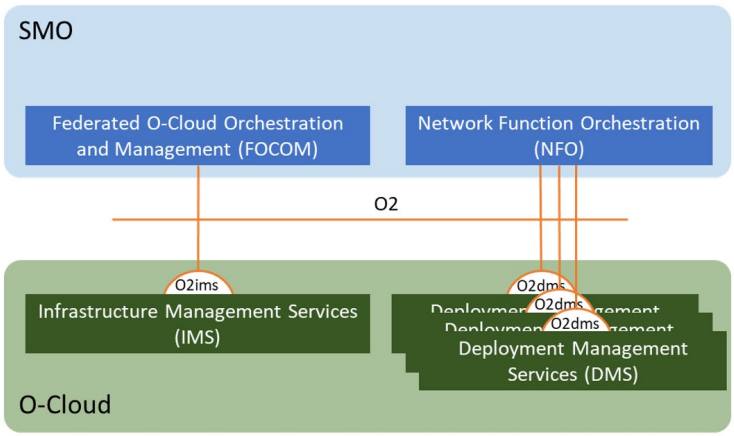


Figure 2-24: O-Cloud Energy Savings - Components and interfaces [20]

O2 IMSs are tailored for the management and provisioning of O-Cloud resources available to the SMO through FOCOM over O2ims. This includes the allocation of available O-Cloud resources (e.g., compute resources and networks) into O-Cloud node clusters and all cluster-wide operations throughout their lifecycle. O2 DMSs are dynamically created over O2ims for NF deployment placement and management of O-Cloud node clusters. They prepare O-Cloud node clusters for the network functions being deployed, with exceptions for cluster-wide operations requested by SMO through FOCOM over O2 IMS. Figure 2-24 summarizes the involved components and interfaces specified by O-RAN.

The O1 interface connects all O-RAN managed elements (and the management entities within the SMO framework. It ensures the operation and management – e.g. Fault, Configuration, Accounting, Performance, Security (FCAPS), software management, file management) of O-RAN components. This interface manages various O-RAN components, including the Near-RT RIC, O-CU, and O-DU in 5G NR. The monitoring of the RAN is conducted through the O1 interface using performance monitoring jobs, which provide aggregated counters. The O1 monitoring also interfaces with the Non-RT RIC to facilitate comprehensive performance analysis and management.

In summary, the integration of CPU energy saving modes within the O-RAN architecture involves a complex interplay of telemetry, inventory data retrieval, and dynamic network energy configuration changes. The O2 and O1 interfaces play critical roles in managing O-Cloud resources and orchestrating the deployment and operation of network functions, ensuring efficient and sustainable network performance.

2.2.3 RIS

As introduced in BeGREEN D2.2 Section 4.3.2 [21], RIS will be both controlled by the Near-RT RIC for near-RT operations (such as RIS per-element configuration) and the Non-RT RIC (such as logical split of the RIS). To this end, the RIS-enabled BeGREEN architecture is updated as depicted in Figure 2-25.

To integrate the RIS (i.e., the RIS Actuator) with the Near-RT RIC, an extension of the E2 interface is considered, denoted as E2+ in the BeGREEN architecture. To this end, two new SMs will be used:

- **E2 Service Model Smart Surface Control** (**E2SM-SSC**): This SM would modify and initiate RIS control related call processes and messages, and it will execute commands that may result in change of RIS control behaviour in a near-RT time scale. Some of the procedures will be:
    - Load pre-calculated configuration from a codebook. This procedure requires to specify the following parameters:
        - *conf_id*: configuration identifier which normally will map to a specific azimuth angle $\Theta_i$and elevation angle Φ_i.
        - *codebook_id*: codebook identifier. Several codebooks may be available at the RIS actuators. These will be normally similar to the ones presented in [22] and will translate transmitted angles ($\Theta_t\Phi_t$) to reflected angles ($\Theta_r\Phi_r$) given the configuration specified by ($\Theta_i\Phi_i$).
        - *RIS_id*: RIS identifier to select the target RIS.
    - Set phase-shift of a specific element on a RIS, requiring the following elements:
        - *element_id*: identifier of the element of the RIS that needs to be modified.
        - phase_shift: amount of phase shift to be set in the given antenna element.
        - *RIS_id*: RIS identifier to select the target RIS.
    - Load new codebook, with the parameters:
        - *codebook_conf*: codebook information that describes a particular RIS model
        - *codebook_id*: codebook identifier.
    - Delete existing codebook, containing the parameter:
        - *codebook_id*: codebook identifier.
- **E2 Service Model Smart Surface Monitoring** (**E2SM-SSM**): This SM would monitor RIS state in a near-RT time scale. Some of the procedures will be:
    - Get RIS configuration, which will retrieve the current configuration of a given RIS. It will require the parameter:
        - *RIS_id*: RIS identifier to select the target RIS.
    - Get loaded codebooks, which will retrieve the current codebooks available in each RIS actuator. No parameters are required.
    - Get phase-shift of a specific element, which will require the following parameters:
        - *element_id*: identifier of the element of the RIS from which the phase shift need be retrieved.
        - *RIS_id:* RIS identifier to select the target RIS.
    - Get energy consumption, which will obtain the energy consumed by the RIS in a given time interval. Thus, the parameter required will be:
        - *time_lapse*: time lapse to be used to calculate the energy consumed by the RIS

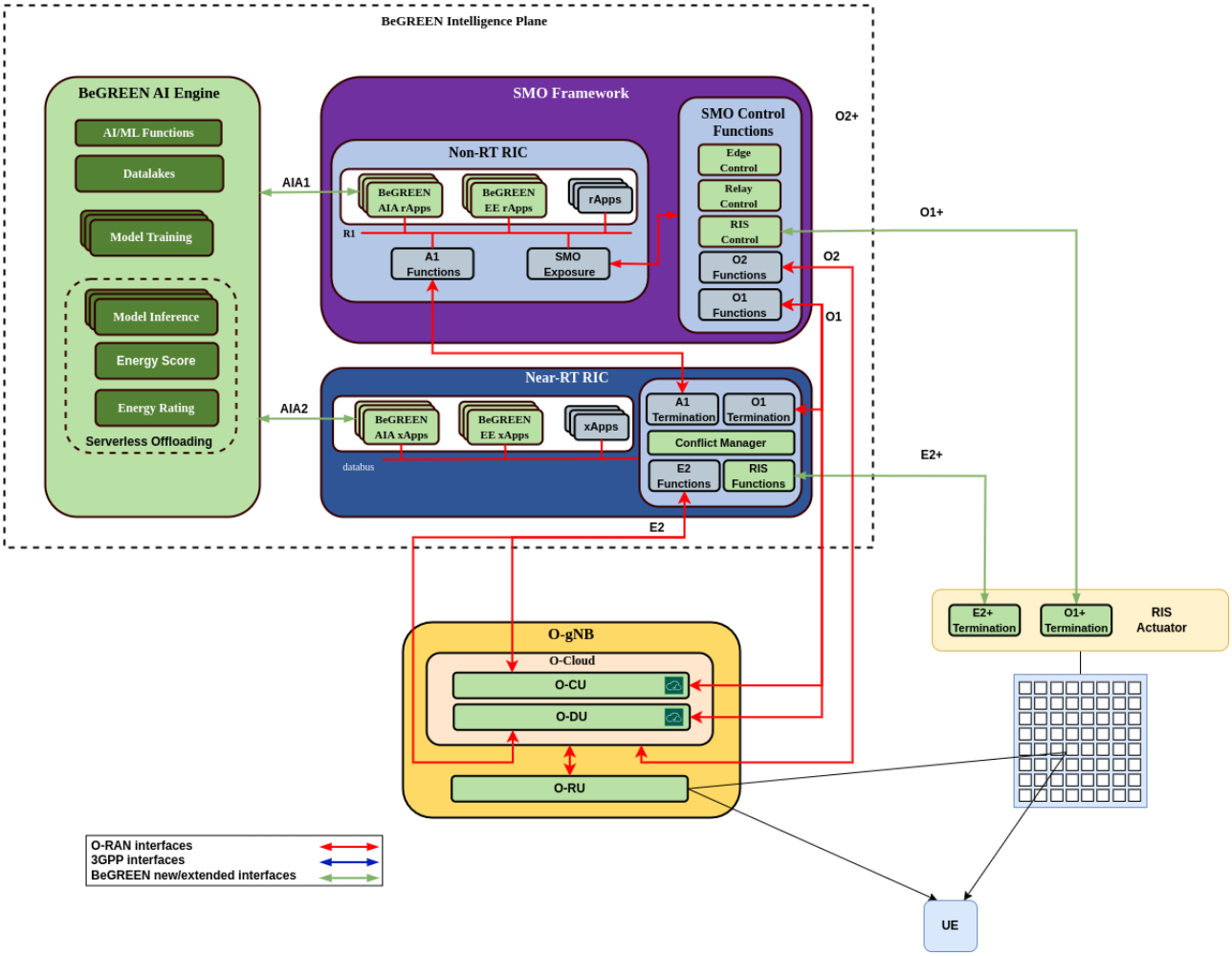


Figure 2-25: RIS - Updated RIS-enabled BeGREEN architecture

These procedures will leverage the E2 services defined by O-RAN (i.e., REPORT, INSERT, CONTROL, POLICY and QUERY) and other E2 support services. However, given that the RIS is not an E2 node, **E2SM-SSC** and **E2SM-SSM** SMs will not have some of the E2SM common IEs that other SMs have [23]. For example, IEs related to Cell Global ID will not be used, since a RIS device may not be attached to one single cell, but rather provide service to different cells simultaneously. Similarly, IEs related to Group ID, Core CP, QoS ID, Network Interface Type, Network Interface Identifier, Network Interface Message ID, Radio Resource Control (RRC) Message ID, Serving Cell Physical layer Cell Identifier (PCI), Serving Cell Absolute Radio Frequency Channel Number (ARFCN), Cell Radio Network Temporary Identifier (RNTI) and Partial UE ID will need to be updated or will not be used. For this reason, we refer to these two implemented SMs as the E2+ interface, which need to be supported both in the RIS Actuator and in the Near-RT RIC (see RIS Functions module in the Near-RT RIC). More details on the exact E2 mapping will be provided in later deliverables.

To integrate the RIS Actuator with the Non-RT RIC, the O1 interface will be used [24]. The O1 interface is based on the NETCONF protocol and uses YANG to model the configuration and data collection commands required for the interactions. We foresee that, among the main tasks to be performed by the Non-RT RIC, are the RIS logical split management and codebook optimization/training, since these are tasks be done on a non-RT time scale. This is mainly because they are usually dependant on barely dynamic aspects such as the physical location, the number of incumbent operators or the types of QoS classes. The codebook optimization can be done through the standard policy-based A1 interface, which is specifically designed to send optimized configurations from the Non-RT RIC to the Near-RT RIC. However, the logical split is a management task that is intended to be performed through the O1 interface.

To implement a RIS-enabled O1 interface, termed as O1+ in the BeGREEN architecture, both the RIS Actuator and the O1 Functions module need to implement a simplified RIS YANG model that defines the RIS logical split rationale (i.e., mapping between RIS_id's and elements_id's) and support standard NETCONF operations such as get, get-config, edit-config, lock, unlock, close-session, kill-session, etc. Again, further details of O1 RIS-enabled configurations will be provided in future deliverables.

2.2.4 Relays

The development of AI/ML-based functionalities for relay control leads to the improvement of decisions of relay deployment, relay activation/deactivation, etc., resulting in a better system performance and an energy consumption reduction. As introduced in BeGREEN D2.2 [21] and D4.1 [1] these relay functionalities cover different aspects.

First, BeGREEN considers the identification of periods of time and geographical regions with high traffic demands and poor propagation conditions (i.e. coverage holes). Another relevant relay functionality is the identification of the most adequate location for the placement of new relays and their initial configuration parameters. Moreover, the identification of UEs that can be good candidate RUEs (i.e. UEs with relaying capabilities that may serve neighbour users with poor propagation conditions with respect to the gNB [5]) is also considered. Finally, BeGREEN considers the dynamic activation/deactivation of these relays/RUEs. According to this, relays/RUEs are activated when needed to serve other UEs located in the coverage hole regions and are deactivated when they are not necessary, with the aim to minimize the energy consumption.

Next subsections detail the integration within the BeGREEN architecture of the main functions needed to control the relays and to collect the required network measurements.

2.2.4.1 Main functions involved in the relay control

For the implementation of the relay control functionalities, several components are necessary in the BeGREEN architecture. As shown in Figure 2-26, a Relay Control entity is placed at the SMO. This entity is in charge of the interaction with the relays for the collection of the network measurements and of the relay reconfigurations through an extended O1 interface., This interface is denoted as O1+ in the BeGREEN architecture, since as in the RIS case, BeGREEN will require of new YANG models to represent the relay configuration and functions. In turn, as shown in Figure 2-26, gNB measurements are sent to the SMO through the O1 interface.

The considered relay control functionalities are sustained by different rApps placed in the Non-RT RIC. In particular, the data collection rApp manages the different processes related to the collection of measurements. Moreover, the Relay Function Management (RFM) rApp, is in charge of the coordination and management of all the functionalities related to the control of the relays. This RFM rApp decides when and where each of these relay control functionalities is executed. These functionalities are sustained by means of AI/ML models hosted in the AI Engine (i.e., as shown in Figure 2-26, CHD, Fixed Relay Placement, Candidate RUE Identification and Relay Activation). These AI/ML models make use of collected information stored in the AI Engine datalakes. Additionally, as introduced in section 2.1, the BeGREEN project introduces the concept of AIA rApps. These specific rApps cover different aspects such as model inference exposure, data pre-processing, performance monitoring of the AI/ML models and determining the necessity of model updates or model retraining. The proposed solution considers a different AIA rApp for each of the proposed relay-related AI/ML models. A brief description of these processes and functionalities is presented in the following paragraphs, while the different workflows and algorithms are detailed in section 3.4.

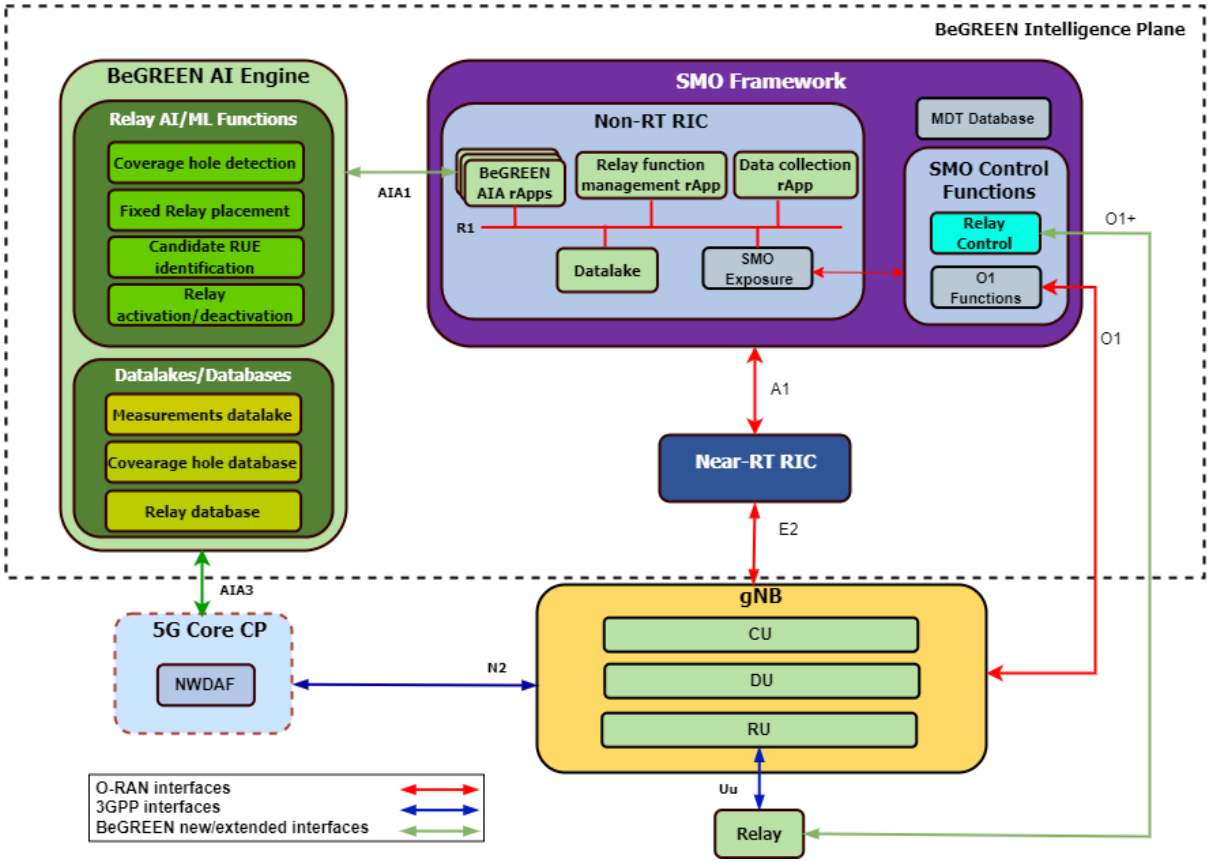


Figure 2-26: Relay Control - Main functionalities involved

On the one hand, the relay control at the SMO and several rApps in the Non-RT RIC are involved in the processes of network monitoring and measurement collection (see details of these processes in section 2.2.4.2). On the other hand, the RFM rApp in the Non-RT RIC triggers the different relay control functionalities. Concerning the CHD process, it makes use of a set of collected measurements with the aim to identify geographical regions with large traffic demands and poor coverage. For this purpose, a clustering process is executed in the AI Engine. The result of this process is a characterization of the coverage holes identified in each cell. This characterization is stored in the Coverage Hole Database as shown in Figure 2-26. More details of the CHD process can be found in section 3.4.1. After the identification of a coverage hole, the RFM rApp will decide the most adequate solution to address the problems in the detected coverage hole.

One possible solution is the use of RUEs; for this purpose, the RFM rApp may trigger the process of Candidate RUE Identification. In general, UEs with good propagation conditions with its associated BS, a static/semi-static mobility pattern, a periodical and predictable space-time location and large session durations, may be good candidates to become RUE. The list and characterization of candidate RUEs for each coverage hole is stored in the Relay Database (see Figure 2-26).

Another possible solution to address a specific coverage hole is the placement of a fixed relay. In this case, the RFM rApp may trigger the Fixed Relay Placement functionality to determine an adequate geographical location to place a new relay and its initial configuration parameters. In case a new fixed relay is deployed, the configuration parameters of this new relay are stored in the Relay Database.

With the aim of improving the system performance at the coverage holes and reduce the overall energy consumption, both fixed relays and RUEs are dynamically activated/deactivated depending on the number of users in their surroundings. To do this, the Relay Activation/Deactivation process makes use of recently collected measurements and information related to the relays status, available at the Relay Database. This information is used as input for the Relay Activation/Deactivation process that makes use of a trained model

to take adequate relay activation/deactivation decisions (see details in section 3.4.4).

2.2.4.2 Collection of network measurements

Figure 2-27 presents the process of the collection of network measurements. As shown in step 1, a continuous network monitoring process is run at each gNB to identify cells that require the activation of some of the proposed relay control functionalities. By means of performance counters, each gNB keeps track of metrics such as the number of attempted calls, number of successful connections, number of call droppings, etc. Relevant metrics for the identification of cells with bad performance are described in [25][26].

The metrics used for this purpose in the proposed approach are described in [25] as follows:

1. *Number of DRBs successfully setup (DRB.EstabSucc.5QI):* This measurement provides the number of Data Radio Bearers (DRBs) successfully established in a specific cell. Each DRB that was successfully setup to the UE increments the corresponding sub-counter by 1 per mapped 5G QoS Identifier (5QI).
2. *Number of released active DRBs (DRB.RelActNbr.5QI):* This measurement provides the number of abnormally released DRBs that were active at the time of release. DRBs with bursty flow are seen as being active if there is user data in the Packet Data Convergence Protocol (PDCP) queue in any of the directions or if any DRB data on a Data Radio Bearer has been transferred during the last 100 ms. DRBs with continuous flow are seen as active DRBs in the context of this measurement, as long as the UE is in RRC connected state. The measurement is split into sub-counters per mapped 5QI.

According to the previous metrics, the proposed approach considers the drop call rate as a metric to decide the activation of some of the relay control functionalities for a specific gNB. The drop call rate can be measured as the number of call droppings divided by the number of attempted calls in a specific gNB during certain period of time T_{drop_eval}, according to:

$$DRB_{drop\_rate} = \frac{\sum_x DRB.RelActNbr.5QI\_x}{\sum_x DRB.EstabSucc.5QI\_x},$$

where *x* represents each of the 5QI level identifier. With a periodicity T_{drop_eval}, the drop call rate is computed in all the gNBs in the network. Alternatively, cell droppings can also be measured by means of the retainability that computes the number of DRBs abnormally released divided by the aggregated DRB active session time for each mapped 5QI [26].

In the case that a drop call rate is higher than certain a given threshold in a particular gNB, the Relay control at the SMO activates a process to collect more specific measurements in the coverage region of this gNB to carry out a deeper analysis of the source of the identified problems (step 2-3). For this purpose, an accurate coverage and space/time traffic characterisation is essential to take adequate decisions of relay placement, relay activation/deactivation, etc. For this reason, the collected measurements that provide this space/time characterisation need to be associated with the timestamp and UE geo-location information where the measurement was taken. According to this, the proposed approach is based on the Minimization of Drive Tests (MDTs) feature [27]. MDT reports consist of periodical UE measurements including the measurement location (i.e. latitude, longitude, altitude), RSRP, RSRQ, and Signal to Interference and Noise Ratio (SINR) values of the serving and neighbour BSs. It also contains information of the Channel Quality Indicator (CQI).

As shown in step 4 in Figure 2-27, the relay control at the SMO sends a command to activate and configure the process of collection of MDT measurements for the gNBs with a drop call rate higher than the established threshold. Then, when a UE with MDT capabilities and MDT consent establishes a RRC connection with one of these gNBs, the network sends a measurement configuration message to the UE with the configuration parameters of the MDT. This configuration message contains information about how the measurement collection and logging is triggered, information about the list of measurements to be collected, and the MDT deactivation condition. In the proposed approach, UEs are configured to collect and send MDT measurements with a periodicity $T_{MDT\_periodicity}$.

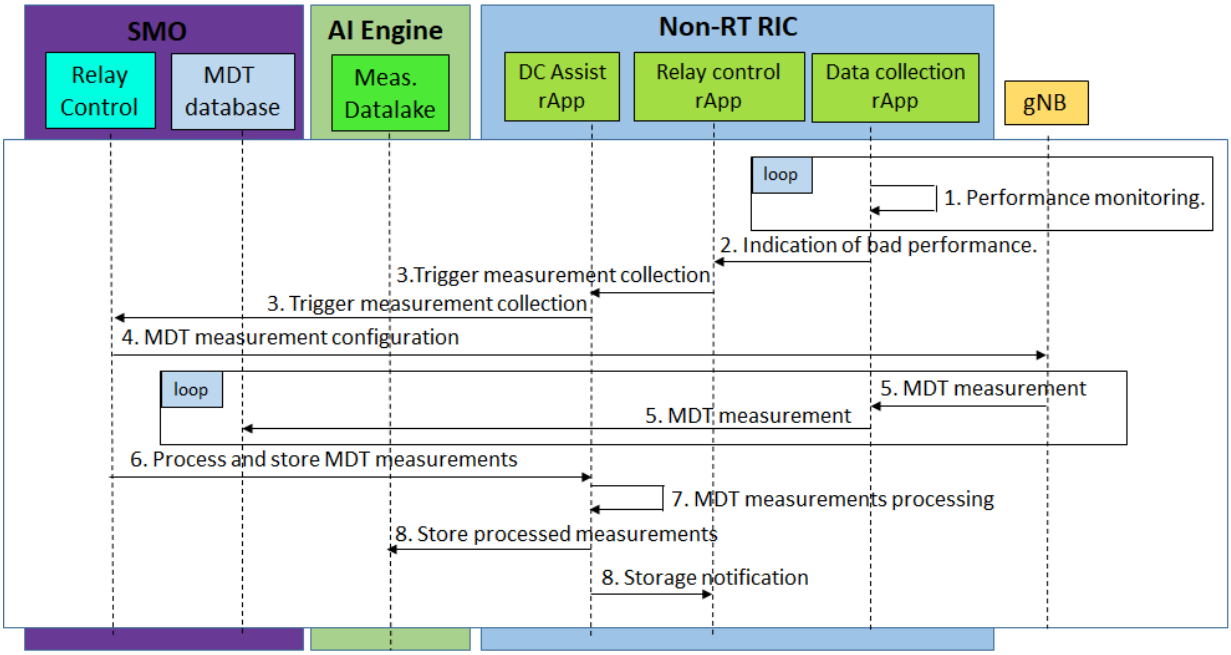


Figure 2-27: Relay Control - Workflow of the collection of network measurements.

Coverage holes can be identified by observing a low value of RSRP of the serving and neighbour cells, together with the geo-location coordinates – obtained via Global Navigation Satellite System (GNSS) or by means of RF fingerprint – and a relative time stamp. Geo-located measurements collected from UEs in *RRC_connected* state are transmitted periodically to the network (with periodicity $T_{MDT\_periodicity}$). In turn, UEs in *RRC_idle* or *RRC_inactive* state can log measurements and transmit them later when the UE enters in *RRC_connected* state. The measurements collection and logging is done until the UE leaves the current cell. Collected MDT measurements are stored in an MDT database (step 5). After certain established period of MDT measurements collection $T_{MDT\_collection\_period}$, the MDT process is deactivated in the corresponding cell. Finally, the collected MDT measurements are filtered and processed by a data collection assist rApp (step 6) and stored in the measurements datalake in the AI Engine (see step 7-8 in Figure 2-27).

2.3 Integration with BeGREEN Edge domain

As was introduced in Section 2.2.2, the management of resources in the O-Cloud is a responsibility of the FOCOM component, located in the SMO, and the IMS, located in the O-Cloud [20]. Both components are connected through the O2-IMS interface, which basically exposes IMS services to the FOCOM component, concretely: inventory, monitoring, provisioning, software management and Life Cycle Management (LCM).

According to the different possible O-RAN deployment scenarios [28], and due to O-RAN disaggregated and virtualized features, the edge domain can be considered to contain components of the O-Cloud or of the Near-RT RIC. While the BeGREEN approach to manage O-Cloud or vRAN energy savings was already introduced in section 2.2.2, the interested reader can also refer to BeGREEN D3.2 [15] for a detailed description of hardware acceleration approaches to enhance the energy efficiency of O-Cloud and Near-RT RIC components. This section addresses scenarios where edge resources are allocated to the UPF of the 5G Core (5GC) and/or to AI-driven applications.

The management of resources in the O-Cloud infrastructure includes two kinds of resources: physical and logical. In the case of the Edge domain, as will be further described in Sections 3.5 and 3.6, BeGREEN aims at intelligently managing physical resources, respectively CPUs and GPUs, to enhance the energy efficiency of UPFs and services. To this end, as was depicted in the main architecture (Figure 2-1), BeGREEN incorporates the following components to extend the baseline O-RAN architecture:

- Edge Control Function: Placed in the SMO, it should implement similar functionalities as the FOCOM component but focused on the resources allocated to non-RAN functions and services.
- Edge Resource Controller: Placed in the Edge, and similarly to IMS, it should expose methods to allow the monitoring and management of edge resources.
- O2+: Interface used to expose Edge Resource Controller capabilities to the Edge Control Function in the SMO. Its design should follow similar principles as O2-IMS.

Note that due to the expected low TRL of the BeGREEN solutions, making use of these components (TRL 2 or 3), it is not expected a detailed definition and specification of them within the scope of the project. Nevertheless, follows a list of the main requirements according to the use cases presented in BeGREEN D4.1 [1] and extended in this deliverable (Sections 3.5 and 3.6):

- Inventory and policies: The Edge Resource Controller shall expose methods to obtain the characteristics of the server, such as the pool of available resources (e.g. number and type of CPUs, number and type of GPUs, etc.) or the available energy saving policies (e.g., performance or energy saving mode, available P-states or C-states, etc.).
- Monitoring: The Edge Resource Controller shall expose status, metrics and alarms, related to the utilization and availability of resources (e.g., the CPU or GPU load, power and energy consumption, etc.) allowing consumers (i.e., the Edge Control Function) to subscribe to them through the O2+ interface. Combined with performance metrics of the hosted functions, for instance the data volume being processed by the CPU, energy related metrics may be used to compute the Energy Score and Rating of the servers.
- Dynamic management: Functions or policies shall be exposed throughout the O2+ interface to allow dynamic management of resources, like controlling the allocation of CPU or GPU resources to specific processes, or their configuration. Non-RT control shall be the minimum time granularity, although near-RT control could be useful in some use cases (e.g., managing C-states).

Other works from the state-of-the-art have proposed similar approaches to incorporate the management of resources not dedicated to RAN. For instance, the project Smart-5G from the Open Networking Foundation (ONF) [29] also targets energy optimisations at the RAN and CN domains. In the ONF approach, the control of CN resources is decoupled from the SMO and implemented in a different external component, which just makes use of the RAN telemetry exposed by the SMO.

The unified approach proposed by BeGREEN can enhance the overall energy efficiency by enabling more holistic optimization strategies. By integrating RAN and non-RAN resource management within the same rApps, BeGREEN leverages the synergy between different network components, potentially leading to better resource allocation and utilization. Indeed, the required functions could be easily integrated into a more general O-Cloud approach, extending the FOCOM, IMS and O2 components to manage the allocation of resources to RAN, Core and services. This would not only enhance the performance and sustainability of the network but also ensure that the allocation of resources matches with the dynamic needs of both RAN, CN and service domains, leading to a more efficient and adaptable network infrastructure. This approach is aligned with research proposals that aim to converge in the same hardware platform the workloads of 5G RAN, Core, AI/ML and services [30].

3 Initial Evaluation of AI/ML-Assisted Procedures to Enhance Energy Efficiency

This chapter details the BeGREEN AI/ML-assisted procedures to enhance energy efficiency in the RAN and Edge domains, along with the evaluations performed. First, we extend the description of the methods reported in BeGREEN D4.1 [1], focusing on the final use cases and the design of the solutions that are being considered within BeGREEN scope. Secondly, for each of the methods we provide validations of the algorithms, workflows and/or design principles. Given the varying maturity levels of these solutions, the scope of the evaluations is heterogeneous and, in some cases, a final evaluation of a specific method or algorithm is presented, while in others only initial insights or validations are provided.

Section 3.1 deals with the application of dimensionality reduction on ML models to achieve energy efficiency in the ML services without sacrificing accuracy. The proposed approach is validated with data from real dataset from an MNO. Section 3.2 addresses the problem of compute resource allocation in vRANs, which is also being considered by O-RAN Alliance [3], as presented in Section 2.2.2. A solution based on RL is applied and validated experimentally, which aims to optimize the allocation of shared compute resources to virtual BS. Section 3.3 deals with another use case being prioritized by O-RAN [3][31]: 5G carrier/cell on/off switching. In BeGREEN we consider a realistic NSA scenario based on a real MNO dataset, and we evaluate strategies performing AI/ML-driven offloading from 5G to 4G Radio Access Technologies (RATs).

Section 3.4 provides a detailed description of the workflows and algorithms required to implement the different methods introduced in section 2.2.4, targeting energy efficient relay-enhanced RAN control. An initial validation of each of the methods is also presented, based on simulative work modelled according to real measurement campaigns. Section 3.5 deals also with the energy efficient allocation of compute resources but applied to edge servers hosting the UPF of the 5GC. An experimental characterization of the energy consumption of a Vector Packet Processor (VPP) and Data Plane Development Kit (DPDK) based UPF implementation is presented, introducing mechanisms to match traffic demands, compute resources and energy consumption. Also considering the Edge domain, Section 3.6 addresses the problem of the joint orchestration of vRANs and Edge AI services. According to the experimental characterization of the problematic, an AI/ML solution based on a Bayesian online learning algorithm is detailed. Section 3.7 presents the initial validation of the Intelligence Plane implementation, focused on the integration of the AI Engine and the non-RT RIC, which is built on the demonstration performed at the 2024 EuCNC & 6G Summit.

Finally, Section 3.8 provides a summary of the use cases and the AI/ML methods presented in this chapter, highlighting the main features and the initial results that have been obtained. Note that this deliverable does not include an evaluation of the RIS integration into O-RAN, described in section 2.2.3. BeGREEN D4.3 will provide details about the results on such integration.

3.1 Dimensionality reduction

Traffic volumes in radio networks have increased exponentially over the last decade. Therefore, although the efficiency of these networks has improved, the continued rise in traffic makes minimizing energy consumption a critical challenge in the telecommunications industry. To ensure maximum energy efficiency it is necessary that the configuration of each element in the network is optimal at all times. In this section dimensionality reduction of input data is considered, which is a technique for reducing the amount of data that need to be processed to detect entities in the network that are operating with low energy efficiency and may need reconfiguration.

3.1.1 Solution design and use case

An example of the utility of dimensionality reduction is a demonstration of how predictive models can

maintain high levels of accuracy when trained with datasets with reduced dimensions. Telecommunication networks typically collect thousands of types of performance data, which represent the operating conditions of the network. Based on these data, a predictive model that predicts, for instance, the energy consumption of a cell in a telecommunication network, can generate accurate predictions.

A comparison between the relative accuracy of predictions made with the complete dataset and predictions made with feature-reduced datasets can be carried out. Accuracy is calculated using the R^2 coefficient of determination that measures the model prediction accuracy. The features that are removed from the model in the feature-reduced datasets are those features where the 'feature importance' to the model is lowest. Feature importance refers to techniques that calculate a score for all the input features for a given model. A higher 'feature importance' indicates that a feature has a larger impact on the model that is being used to predict a certain variable. Then, an evaluation of the relative processing and data movement/storage costs involved in training the model can be used to find the optimum balance between predictive accuracy and costs of making the prediction. The general workflow for applying dimensionality reduction to a predictive model is shown Figure 3-1.

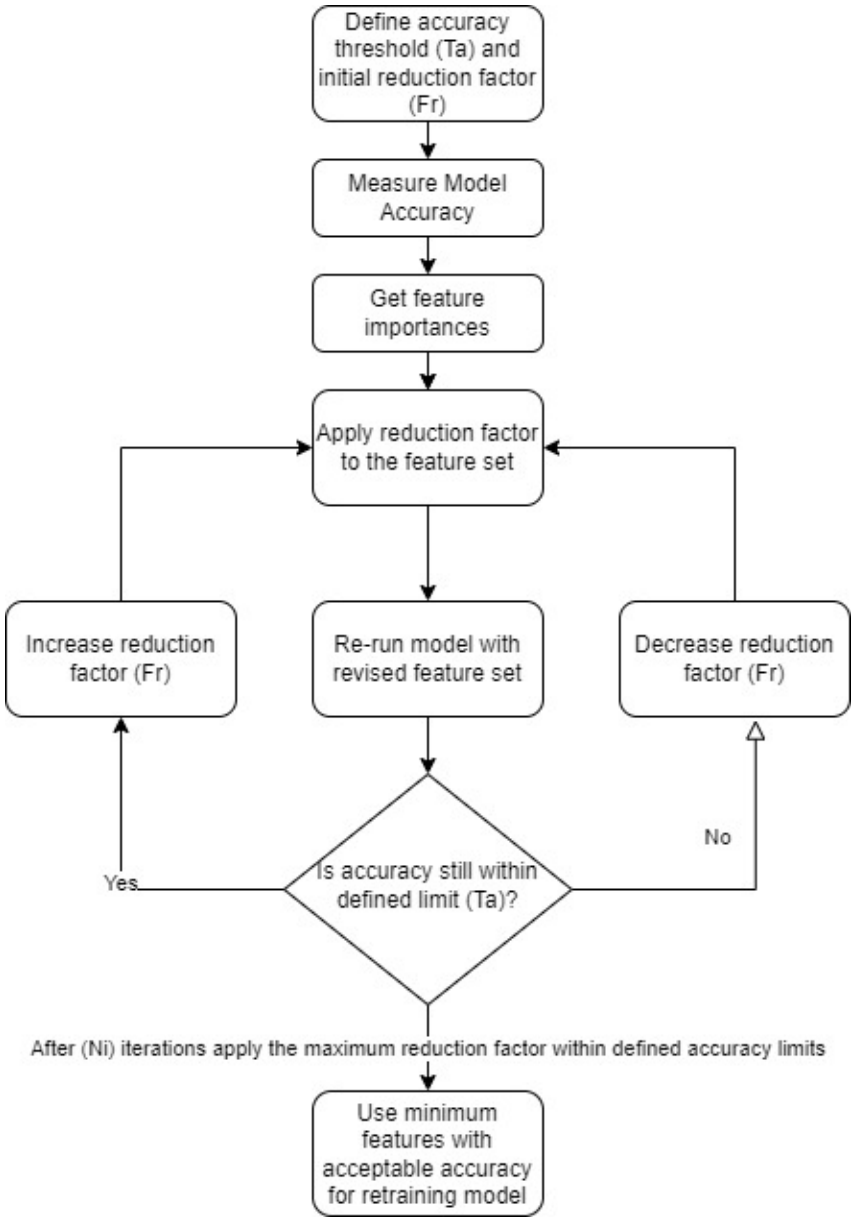


Figure 3-1: Dimensionality reduction – workflow

The decision on increasing or decreasing the number of features is based on the prior setting of an acceptable accuracy limit (**Ta**) for the current predictive task, an initial reduction factor (**Fr**) and the number of iterations (**Ni**) of the workflow that should be performed. The setting of **Ta** will determine the minimum accuracy level of the dimensionally reduced model that is defined by the workflow. **Fr** affects the step size with which iterations of the workflow will approach the ideal balance between accuracy and energy efficiency in terms of lower number of features. **Ni** influences how closely the final result of the workflow will match this ideal balance. Based on experimental results, an **Fr** of 0.5 (corresponding to halving the dataset with each iteration) and **Ta** = 10 iterations of the workflow offers a good trade-off.

The method is generally applicable to predictive models for any metric. In BeGREEN it has been demonstrated with XGBoost models predicting the energy consumption of a cell, as will be presented in the next subsection.

3.1.2 Initial evaluation

The results of this comparison are shown in Table 3-1. The first column shows the number of features used in the predictive model and the second the percentage from the total number. The third column shows the R^2 accuracy score of the model, relative to the accuracy of the model using all the features (first row). The third column shows the number of CPU cycles needed to train the model, relative to the value using all the features (first row).

The following diagrams show the relationship between the number of features used to generate a prediction and its accuracy measured by its R^2 score (Figure 3-2) and the number of CPU cycles used to train the model that generates that prediction (Figure 3-3).

Table 3-1: Dimensionality Reduction - Example

Number of Features	% of Features	% Max Accuracy	% Max CPU Cycles
1100	100%	100.0%	100%
50	5%	99.3%	19%
30	3%	98.8%	17%
20	2%	98.4%	18%
10	1%	95.0%	15%

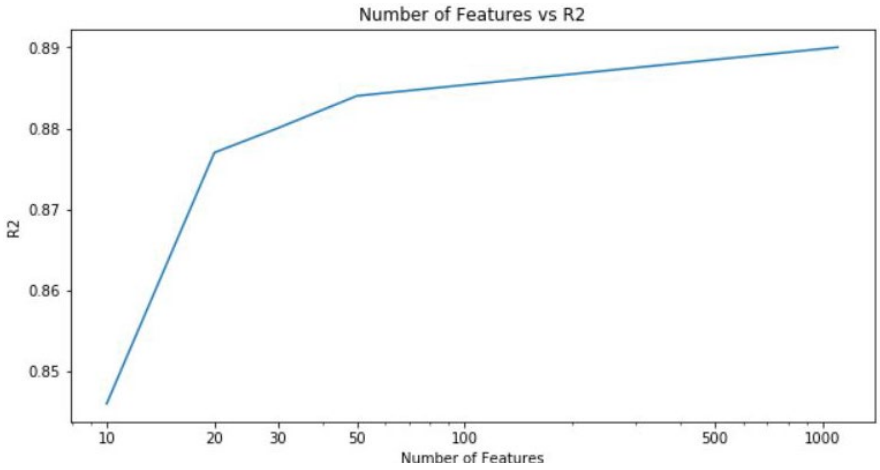


Figure 3-2: Dimensionality reduction example - number of features vs accuracy

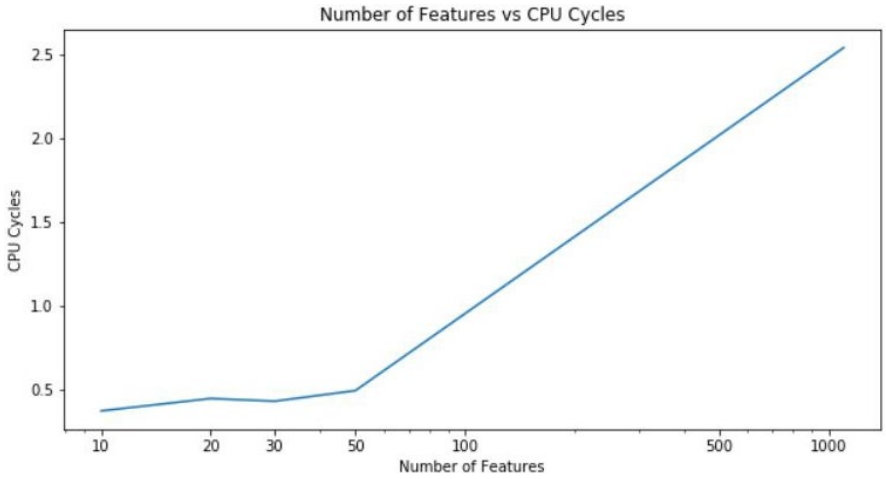


Figure 3-3: Dimensionality reduction example – number of features vs CPU cycles

The 'elbow' on each graph is visible at 50 features. The loss of predictive accuracy when using 50 features is 0.7% but the prediction can be generated using less than 5% of the data volume and less than 20% of the CPU cycles. This means that a large proportion of the energy expended in training models, in addition to the data movement, processing and storage costs associated with this process, can be saved for subsequent model retraining tasks by applying dimensionality reduction to the feature set used to retrain the models.

In addition to the measuring the decrease in CPU usage that can be achieved with this technique it is also planned to create a method whereby it is possible to estimate the energy consumption of the models, dependent on details of the system on which they are executing. It would then be possible to quantify the energy savings of retraining a model using a dimensionally reduced dataset.

3.2 Compute resource allocation in vRAN

In this section, the problem of compute resource allocation in virtualized RAN under shared computing infrastructure is addressed. This problem has already been introduced in the previous BeGREEN D4.1 [1], where the state-of-the-art is discussed and an initial experimental characterization of the problem and an initial problem formulation are provided. The analysis is built on top of this formulation to design an AI/ML solution that relies on RL theory. Then, an experimental evaluation of the proposed algorithm analysing the convergence, the inference time needed to have a practical solution, and its performance with respect to state-of-the-art benchmarks using realistic traffic traces are provided.

The problem addressed in this deliverable is of paramount importance in mobile networks. Unlike other setups, the amount of processing power a vRAN system needs can fluctuate significantly. This dynamism stems from several factors. First, each vBS instance has varying CPU demands depending on the amount of data flowing in both directions (uplink and downlink), the signal quality between the user and the station, and the data transmission method employed. On top of that, efficiently allocating resources becomes even more challenging when multiple vBS instances share the same platform. Here, a delicate balance needs to be struck. Allocating too much computing power (over-provisioning) leads to wasted energy on idle cores, while allocating too little (under-provisioning) can cause performance issues. This resource crunch can lead to problems like data synchronization failures, increased radio errors, and frustrating delays for users.

An ideal scenario would involve dedicating specific processing power to each vBS instance for optimal performance. However, this approach comes at the cost of keeping more cores active, which translates to higher energy consumption.

The solution proposed in this deliverable seeks a middle ground. By dynamically adjusting the number of active cores in a shared pool based on real-time traffic demands, the system can achieve a balance between

performance and energy efficiency. This approach factors in the impact of sharing resources on the overall health of the network, ensuring that cost-saving measures do not come at the expense of user experience.

For the future BeGREEN D4.3, the plan is to address this problem from a different angle. Instead of having a solution that predicts the required computational resources considering interference among processes (noisy neighbour problem), it is planned to propose a resource allocation algorithm to optimize the resource utilization and therefore the energy efficiency of the system.

3.2.1 Solution design and use case

Following the BeGREEN D4.1 [1], in this section the problem formulation, the system model and the considered optimization framework are presented. Specifically, the optimization framework relies on RL, in particular Deep Q-Network (DQN) algorithm[32], enhanced with Relation Networks (RNs) [33] to handle the variability of the input size. Given the characteristics of the objective problem, the RL framework is particularized with discount factor $\gamma = 0$ and episode length $T = 0$ to a contextual bandit problem. The design for the learning agent's context (states), actions, and reward function are presented here.

Context:

In line with the related literature [34][35][36], we use the next metrics to describe the state:

- **Chanel quality**: We use the mean UL Signal to Noise Ratio (SNR) observed by each vBS in the last interval, which allows our agent to infer their UL wireless capacity, and the mean DL CQI to do the same for the DL.
- **Network demand:** The network demand of a vBS is the amount of UE buffered data for both UL and DL during the last decision interval.

We represent DL and UL channel quality for a vBS instance i observed in interval t as $\sigma_{DL,i}^{(t)}$ and $\sigma_{UL,i}^{(t)}$. Furthermore, we let $d_{DL,i}^{(t)}$ and $d_{UL,i}^{(t)}$ denote its DL and UL network demand, respectively. We also assume a known mapping between channel quality and MCS: $g_{DL}^{\square}(\sigma_{DL,i}^{(t)})$ for DL, $g_{UL}^{\square}(\sigma_{UL,i}^{(t)})$ for UL, which is a mild assumption. Because the channel quality bounds the highest MCS, we can estimate the mean number of radio Resource Blocks (RBs) that each vBS can use in both directions given a mean MCS and network demand. This can be estimated using the 3GPP specifications [37]. In this way, we can state the demand for radio resources (in terms of RBs) rather than relying only on the past utilization of RBs, which may differ. Consequently, we denote the number of RBs used for DL and UL for vBS i as p_i^{DL} and p_i^{UL}, respectively. Using the number of RBs and network demand, we define the context of vBS i as:

$$x_i^{(t)} := \left(p_{DL,i}^{(t)}, d_{DL,i}^{(t)}, p_{UL,i}^{(t)}, d_{UL,i}^{(t)}\right)$$

The design of $x_i^{(t)}$ is motivated by the convenience of expressive features and minimal dimensionality and follows the state-of-the-art [34][35][36]. The challenge now is to encode the context information $x_i^{(t)}$ for all vBS instances i in a state vector s with fixed dimensionality D, which is required by the DQN model, in scenarios with a variable number of vBS instances over time. As shown in Figure 3-4, we address this with a RN [32][33].

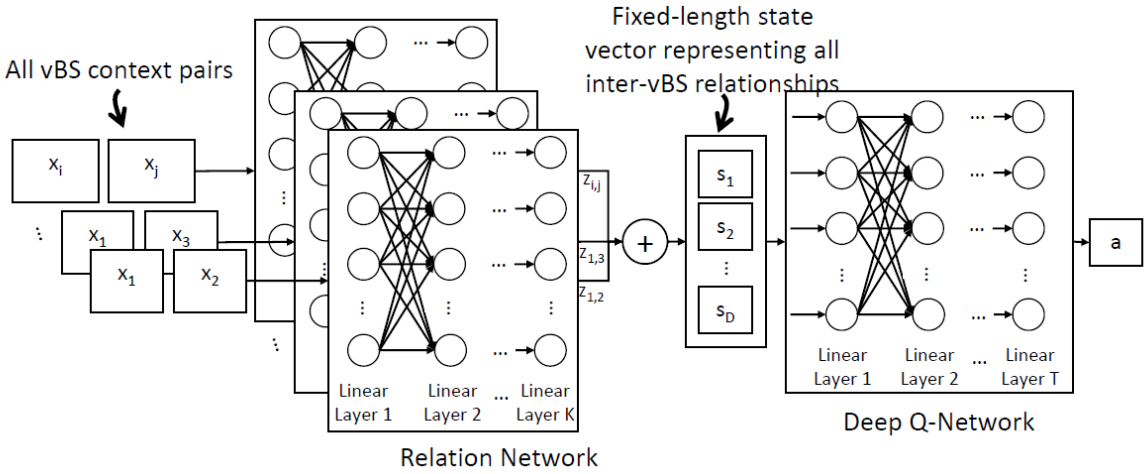


Figure 3-4: Resource allocation in vRAN - proposed ML architecture.

Relation Network (RN):

As the number of vBSs that our solution has to allocate CPU resources for in a particular time interval might be different than in past intervals, the context length changes depending on the number of vBS instances. Rather than building other agents for each of the different numbers of vBS cases or padding the various possible contexts to match a fixed context length, we opted to solve the problem using a RN. A RN can encode the relationship between the context associated to all vBS instances into a fixed-length state vector s. As shown in the literature, RN achieve higher performance and require less computational burden compared to other architectures like multi-layered perceptron (MLPs) [33].

To this end, the RN operates along all possible pairs of objects (context of vBS instances) to capture such hidden relations with a multi-layer perceptron model. Assuming a maximum number of vBS instances supported in the system equal to M, then we have the following possible pairs of context vectors:

$$\mathcal{X} := \{(x_1, x_2), (x_1, x_3), \dots (x_{M-1}, x_M)\}$$

Since the maximum amount of vBS instances at any given moment is bounded, then $|\mathcal{X}|$ is also bounded and fixed over time. The RN ingests sequentially each pair $(x_i, x_j) \in \mathcal{X}$ of possible unpermuted context combinations and generates an output vector $z_{i,j}$ with cardinality D. Once all $\binom{N}{2}$ permutation vectors $z_{i,j}$ are computed by the RN, which is done sequentially, we create an encoded state vector s by aggregating all output vectors, i.e., $s = \sum_{i,j} z_{i,j}$.

In this way, we force order permutation invariance, which is a critical requirement of our problem, i.e., as the RN learns about different latent relations across vBS instances (objects), these learned relations remain invariant regardless the order of the input pair relations. Importantly, our RN not only helps to support variable number of vBS instances over time, but it also provides the DQN model with state information that represents better the relations between them, which is very helpful to capture the impact of the noisy neighbours problem in a state dimension-fixed representation. To this end, we train the RN network jointly with the DQN model as we explain later.

Actions:

Given state $s^{(t)}$, our agent shall activate the appropriate set of CPU cores, described with an activation vector v wherein each element corresponds to the CPU core index that shall be activated. Then, all the vBS instances will fairly share the pool of CPU cores in v. By avoiding pinning vBS workloads into specific cores, we aim at maximizing resource multiplexing and, consequently, at reducing the overall usage of computing resources. To ensure quick convergence, we need to preserve a low action space dimensionality. To address

this, we resolve our action into two steps. In step 1, our RL agent decides the total number of CPU cores that shall be activated to guarantee service. Thus, the set of actions A is $A = \{1, 2, \dots, 2N\}$, where N is the total number of physical cores available. Then, in step 2, we implement a deterministic rule $\rho(a)$ to minimize infrastructure cost. That is $\rho: A \mapsto \mathcal{V}_a$, where $\mathcal{V}_a$ is a set containing all possible activation vectors such that $a = |v|$. Because ρ is a pre-determined rule to minimize cost, the agent can learn its policy π to guarantee service given ρ as part of the environment $\mathcal{E}$.

See, e.g., the General-Purpose Processor (GPP) of Figure 3-5 with $N = 2$. If $a = 1$ then $\mathcal{V}_{a=1} = \{(0), (1), (2), (3)\}$ all the activation vectors in $\mathcal{V}_{a=1}$ are equivalent and any $v \in \mathcal{V}_{a=1}$ could be chosen trivially. However, this is not necessarily the case for other actions a because, as we explained before, modern processors leverage multi-processing CPUs, being two virtual cores for each physical CPU the most common case. For instance, for $a = 2$ (and the same GPP with $N = 2$), the set of possible activation vectors is $\mathcal{V}_{a=2} = \{(0,2), (1,3), (0,1), (0,3), (1,2), (1,3)\}$. Though many of the vectors in $\mathcal{V}_{a=2}$ are equivalent, others are not. Subset $\hat{\mathcal{V}}_{1,a=2} = \{(0,2), (1,3)\} \subset \mathcal{V}_{a=2}$ contains equivalent activation vectors; and so are the activation vectors $\hat{\mathcal{V}}_{2,a=2} = \{(2,3), (0,1), (0,3), (1,2)\} \subset \mathcal{V}_{a=2}$. But any $v_1 \in \hat{\mathcal{V}}_{1,a=2}$ and any $v_2 \in \hat{\mathcal{V}}_{2,a=2}$ are *not* equivalent. On the one hand, any $v_1 \in \hat{\mathcal{V}}_{1,a=2}$ incurs more cache contention than any $v_2 \in \hat{\mathcal{V}}_{2,a=2}$ because all the cores in v_1 share the same physical CPU.

On the other hand, any $v_2 \in \hat{\mathcal{V}}_{2,a=2}$ is more costly than any $v_1 \in \hat{\mathcal{V}}_{1,a=2}$ because v_1 allows turning off more physical CPUs, e.g., if $v_1 = (0,2)$ CPU 1 can be turned off (see Figure 3-5). Figure 3-6 illustrates an example of the operation of our algorithm during three-time steps. Importantly, give a pool of activated CPU cores, all vBS instances will fairly use those cores using a standard scheduler.

In the assumption that, given any static mapping ρ, policy π will provide an appropriate cardinality for the activation vector to guarantee network service ($a = |v|$), we just need to design ρ aiming to minimize the amount of infrastructure (physical CPUs) that has to be activated given a. Consequently, we propose the following simple rule. Let $k(v) \in \{1,2, \dots, N\}$ denote the number of physical CPUs that contain at least one virtual core activated in v. Then, given a set $\mathcal{V}_a$ with all possible activation vectors for action a, we define the ordered superset $\mathcal{W}_a := \langle \hat{\mathcal{V}}_{1,a}, \dots, \hat{\mathcal{V}}_{N,a} \rangle$, where $\hat{\mathcal{V}}_{i,a} = \{v | k(v) = 1, v \in \mathcal{V}_a\}$. In the example above, with $a = 2$ and $N = 2$, $\mathcal{W}_a := \langle \hat{\mathcal{V}}_{1,a=2}, \hat{\mathcal{V}}_{2,a=2} \rangle$. Note that $\hat{\mathcal{V}}_{i,a} = \emptyset$ for some i. For instance, in our toy example with $N = 2$, $\hat{\mathcal{V}}_{1,a=3} = \emptyset$ for $a = 3$. Hence, we let $\rho(a) = v \in \hat{\mathcal{V}}_{m,a}$ such that $m := argmin_i \{i | \hat{\mathcal{V}}_{i,a} \neq \emptyset\}$.

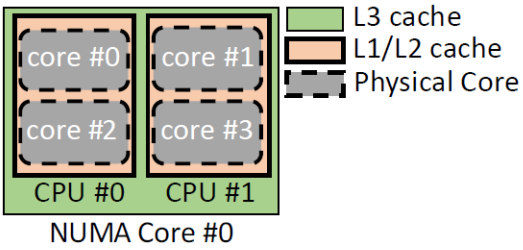


Figure 3-5: Resource allocation in vRAN - simplified example of a GPP

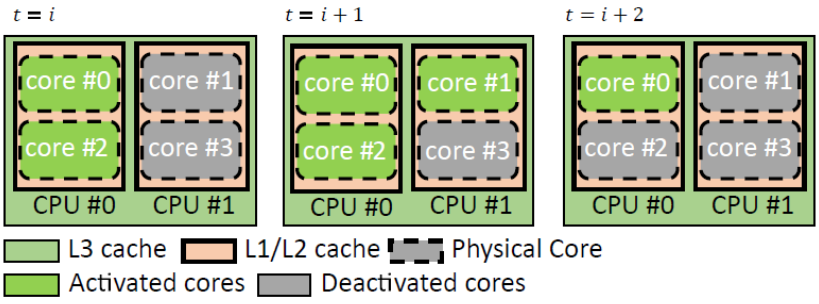


Figure 3-6: Resource allocation in vRAN - example of a sequence of actions from the proposed method

Reward:

Our goal is to meet the traffic demand of all the vBS deployed in the system over time with minimum physical infrastructure (to save costs by turning off CPUs). Assuming a pool with N physical CPUs and $2N$ virtual cores, where cores j and $j+N$ belong to the same physical CPU $\forall j < N$, we let $z(j) \in \{0,\dots,2N-1\}$ denote the *sibling* virtual core i given input virtual core j. A sibling core is that that uses the same physical CPU. For instance, in the toy GPP of Figure 3-5, with $N = 2$ physical CPUs and 4 cores $z(0) = 2$ and $z(2) = 0$.

Following the related literature [38][39], we codify the cost associated to an activation vector v using a linear model. Let us first denote $c_j^{(t)} \in [0,1]$, as the relative usage of computing core j during interval t. If $j \notin v^{(t)}$, then $c_j^{(t)} = 0$; otherwise, $c_j^{(t)}$ is empirically measured. Then, we let $E_j^{(t)}$ model the (energy-related) cost associated with computing core $j \in \{0,1,\dots,2N-1\}$ as follows:

$$E_j^{(t)} := \begin{cases} \alpha_1 + \beta \cdot c_j^{(t)} & \text{if } c_j^{(t)} > 0 \\ \alpha_2 & \text{if } c_j^{(t)} = 0 \text{ and } c_{z(j)}^{(t)} > 0 \\ \alpha_3 & \text{if } c_j^{(t)} = 0 \text{ and } c_{z(j)}^{(t)} = 0 \end{cases}$$

where $\alpha_1 > \alpha_2 > \alpha_3$. Intuitively, α_i models the bias cost of a core, which is different depending on the activation state of core j and its sibling. We choose α_i and β so that $0 \leq E_j \leq 1$.

We now let $\tau_{DL,i}^{(t)}$ and $\tau_{UL,i}^{(t)}$ denote the DL/UL throughput experienced by vBS i during interval t, and then formalize our reward function as:

$$r^{(t)} := \begin{cases} -1, & \text{if } \tau_{DL,i}^{(t)} < d_{DL,i}^{(t)} \text{ for any } i \\ -1, & \text{if } \tau_{UL,i}^{(t)} < d_{UL,i}^{(t)} \text{ for any } i \\ \frac{1}{2N}\sum_{j=0}^{2N-1} - E_j, & \text{otherwise} \end{cases}$$

Algorithm training:

As explained above, the goal is to train a policy to approximate an optimal action-value function Q^*. Our policy π is implemented by the structure of RN+DQN introduced above and, hence, we shall optimize the weights $\Theta := (\theta_1, \theta_2)$ of the combined neural networks to estimate the Q-value function $Q(s,a;\theta) \approx Q^*(s,a)$. To this end, we use a Smooth L1-loss function [40]:

$$L^{(t)}(\Theta_i) := \begin{cases} \frac{1}{2}\frac{x^2}{1} & \text{if } |x| < 1 \\ |x| - \frac{1}{2} \cdot 1 & \text{otherwise} \end{cases}$$

where $x = \mathbb{E}_{(s,a,r,s')\sim\rho}\left[\left(y_i - Q(s,a;\Theta_i)\right)\right]$ and $y_i = r + \gamma\, max_{a^{(t+1)}}\, Q\left(s', a^{(t+1)}; \Theta_{i-1}\right)$. ρ is a replay buffer from where we sample (s,a,r,s'), y_i is the temporal difference target, and $y_i - Q$ is the temporal difference error. We use a target network to stabilize the training process, that is, the learning agent uses a different target network with fixed weights that are used to compute the loss function used in turn to train the primary Q-network. It is crucial to stress that the target network's parameters are periodically synchronized with those of the primary Q-network rather than being trained. The primary Q-network is trained using the target network's Q values in an effort to increase the training's stability. Finally, we use a standard ϵ-greedy approach for exploration.

3.2.2 Experimental evaluation

We have built an O-RAN-compliant experimental testbed to evaluate our solution. The testbed comprises different hosts, which contain the components of an O-RAN deployment and the ones to provide network connectivity to different connected UEs. Figure 3-7 depicts conceptually the testbed that we have built.

First, this testbed has a host, which deploys the SMO and contains the Non-RT RIC where we deploy the AIRIC rApp ① [1]. Second, it has a separate server that hosts the O-Cloud platform where different O-eNB instances can be deployed and also comprises the Near-RT RIC ②. To implement the orchestration and management functions of the O-Cloud platform provided by the SMO, we have opted to implement the O-eNBs deployed in the O-Cloud platform, containerizing srsRAN[8] using Docker. Thus, we use Docker API capabilities to orchestrate and manage containers to implement a minimal O2 interface. In addition, we used a metrics agent as Telegraf[9] to implement the performance monitoring jobs, which allowed us to gather metrics from the O-Cloud platform. Rather than using a commercial orchestrator such as Kubernetes or Docker swarm, we implemented our minimal orchestrator for performance and flexibility. Moreover, we have also implemented minimal O1 and E2 interfaces to allocate resources on the vBSs deployed. Our testbed also includes a host, which contains the Evolved Packet Core (EPC) to provide connectivity to the different UEs attached to each vBS ③. As the vBSs are containerized using Docker, we have isolated the networking from each one another.

The O-Cloud host comprises an Intel i7-7700K GPP with 4 physical CPUs. We use Ubuntu 20.04.5 LTS with kernel 5.13.19. We reserve 1 physical CPU (2 virtual cores) for the OS and custom scripts to manage the experiments, interact with Docker API, and collect data, i.e., we emulate a small GPP vRAN platform with $N = 2$ physical CPUs and 4 virtual cores (as in). The testbed also integrates 4 USRP SDR boards to support up to 4 vBS (and the corresponding UEs to generate network load) ④. To generate uplink and downlink flows, we use *mgen*[10] to initiate a flow from/to the UE to/from the EPC. Given the constrained computing capacity of our testbed, we set the bandwidth of each vBS to 10 MHz. We have generated 60000 context-action-reward data samples, evenly split for scenarios with 2, 3 and 4 vBS instances operating concurrently. We shuffled and split the dataset into a training and a testing set of 40k and 20k samples, respectively.

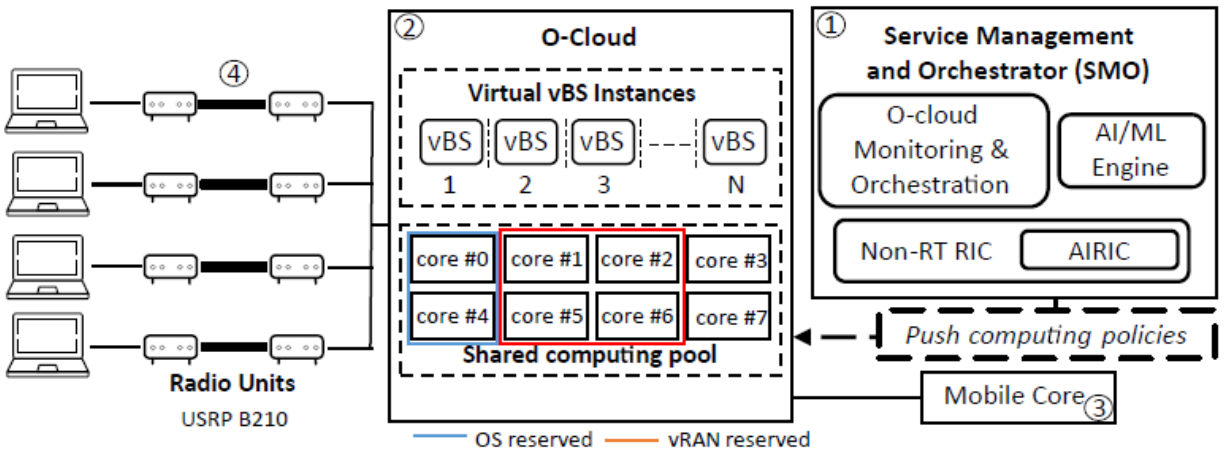


Figure 3-7: Resource allocation in vRAN - Conceptual design of the evaluation testbed

[8] https://www.srslte.com/
[9] https://www.influxdata.com/time-series-platform/telegraf/
[10] https://github.com/USNavalResearchLaboratory/mgen

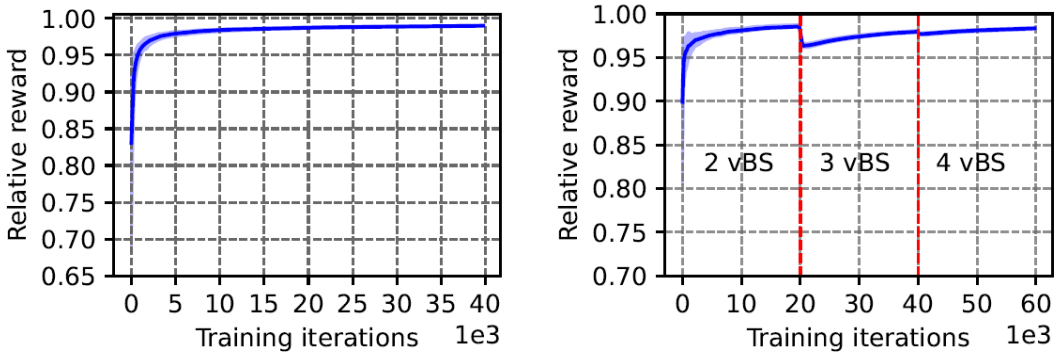


Figure 3-8: Resource allocation in vRAN - convergence evaluation with randomized contexts (left) and an incremental number of vBS (right).

We have implemented our solution using PyTorch[11]. On the one hand, the RN has one hidden layer and the same number of neurons than the output layer, 128. On the other hand, the DQN has one hidden layer with 256 neurons. The initial parameters of the neural networks are initialized from a uniform distribution. We also use the Rectified Linear Unit (ReLU) activation function, and a normalization layer [41] in between hidden layers. For the ϵ-greedy mechanism, we use a decay factor equal to 60% of the size of the training set.

We also use a replay buffer with 20k samples and batches of 128 samples. Finally, we used Adam [42] as our optimizer. These implementation choices are intended to stabilize training based on [41][43].

3.2.2.1 Convergence evaluation

We start evaluating convergence. Figure 3-8 shows the normalized reward of AIRIC over training iterations. The UL/DL load and SNR generated in both plots are chosen uniformly at random. However, while the number of vBS instances is also random (between 2 and 4) in Figure 3-8 (left), they arrive sequentially in Figure 3-8 (right). In the former case, the reward converges to 0.95 in less than 5k iterations. In the latter case, there are expected bumps when new vBSs arrive, but these are small, within 5%. Hence, we conclude that the RN in our proposal correctly learns the relationship across vBSs and how to use its experience to quickly reach close-to-optimal performance.

3.2.2.2 Inference time evaluation

To assess whether our solution is suitable for running in a Non-RT RIC controller, we measured the inference time of our approach for the different number of vBS cases. The results depicted in Figure 3-9 show inference times lower than 1 millisecond (ms) for all cases, which is well below the control-loop cycle of a RIC controller and validates AIRIC to operate therein appropriately.

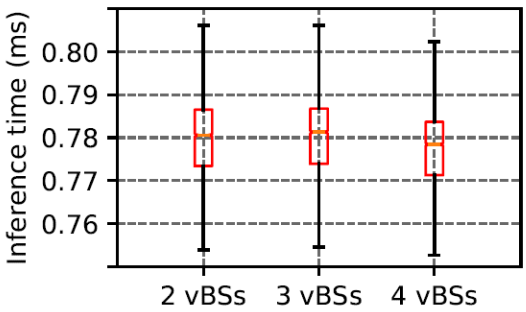


Figure 3-9: Resource allocation in vRAN - Inference time

[11] http://www.pytorch.org

3.2.2.3 Performance benchmark

To better understand the effectiveness of our solution, we now compare our solution against a Single Instance Resource Allocation (SIRA) approach. SIRA is purposely designed to orchestrate optimal resources across vBS instances under the assumption of full computing isolation between instances. Consequently, SIRA represents upper bounds attainable by existing works on vRAN CPU orchestration such as [34] [44].

To evaluate AIRIC, at every interval we choose uniformly at random the number of vBS instances, their DL/UL load and their DL/UL SNR and use both approaches (AIRIC and SIRA) to optimize the allocation of computing resources dynamically. In the case of SIRA, we use different (previously trained) models depending on the number of instances. For comparison, we also depict the performance of an oracle, labelled as "Optimal", that finds the optimal action offline by exhaustive search.

Figure 3-10 depicts the distribution of the normalized aggregate throughput performance of the system (top), the CPU assignments (middle), and the distribution of the reward achieved (bottom), for all the approaches conditioned to the presence of 2 (left), 3 (middle) and 4 (right) vBS instances. Conversely, Figure 3-11 depicts the absolute (left y-axis) and relative (right y-axis) power consumption savings achieved by all three approaches. These savings are in comparison to the power consumed when the default Linux scheduler manages all available CPU cores in the system, as indicated on the x-axis. The box plots represent the 25th and 75th percentiles (edges of the box), the median (line within the box), and the 5th -95th percentiles (error bars). We make three observations: the first observation is that AIRIC provides substantial savings, comparable to the optimal benchmark. Perhaps surprisingly, SIRA shows mildly higher savings in some cases, which leads to our second observation: the savings provided by SIRA come at a huge price in throughput performance, as shown by Figure 3-10. This is worse for denser scenarios: with 4 vBSs, SIRA barely saves 7% computing resources more than AIRIC in average but incurs 50% throughput loss in exchange. This is due to the fact that SIRA ignores the additional computing overhead caused by the noisy neighbour problem and often under-allocates resources, leading to PHY violations and throughput loss. The final observation is that AIRIC provides a throughput performance that is remarkably close to that of "Optimal".

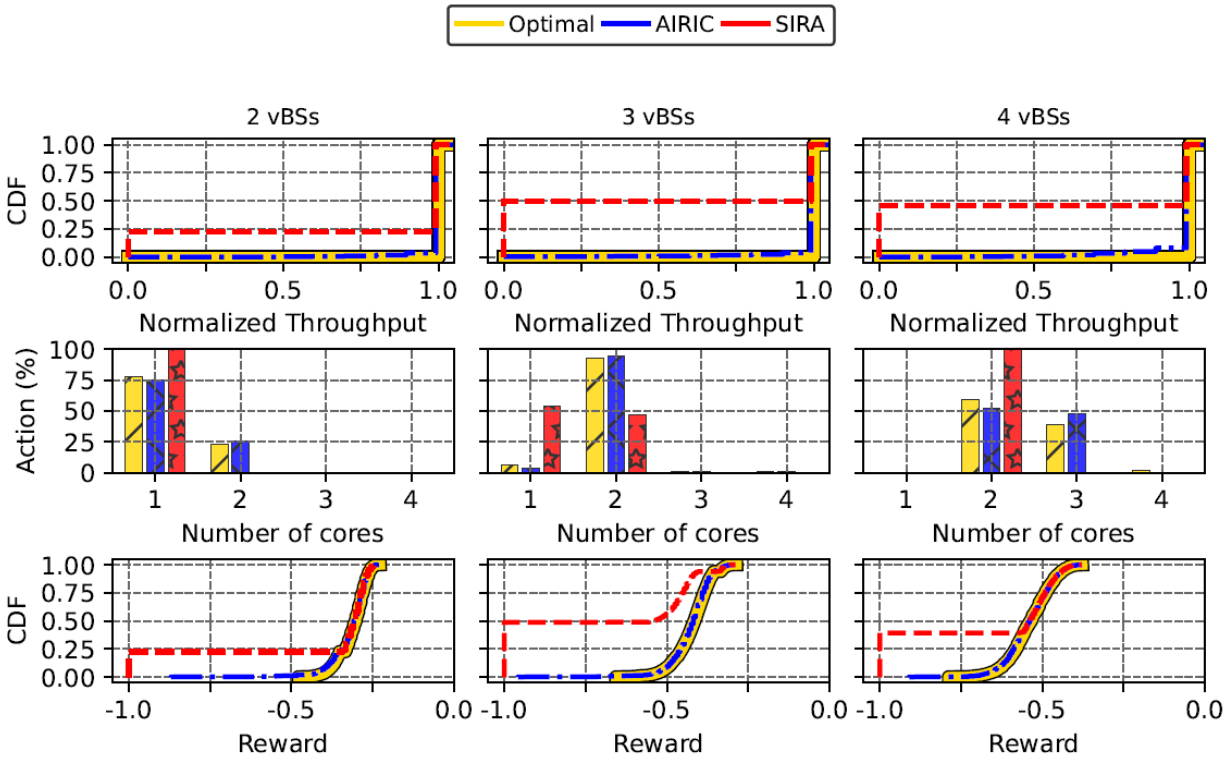


Figure 3-10: Resource allocation in vRAN - performance benchmarking

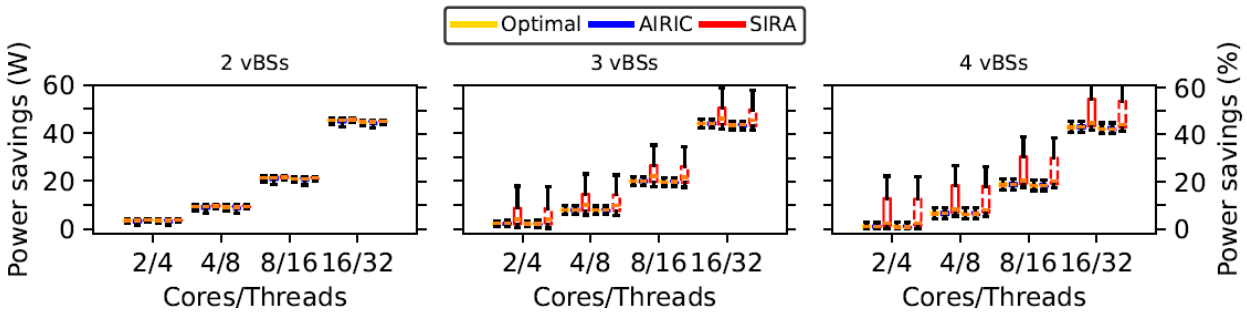


Figure 3-11: Resource allocation in vRAN - Power savings

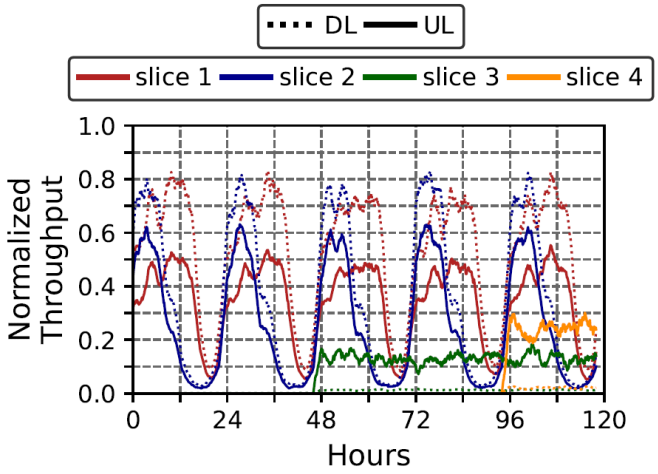


Figure 3-12: Resource allocation in vRAN - realistic load traces

Moreover, Figure 3-10 (bottom) confirms that the reward distribution attained by AIRIC is very close to the provided by the optimal oracle. These observations validate the design.

3.2.2.4 Realistic context traces

We finally test AIRIC with realistic context dynamics. To this end, we have generated context profiles for 4 different vBS instances, implementing network slices with different context profiles, during 5 straight days. Figure 3-12 shows the time evolution of both DL and UL network load for these 4 traces. Slice 1 emulates the behavior of one eMBB vBS in the city center, with common diurnal load patterns. Slice 2 emulates a vBS serving an office building, with a peak load during office hours (9h - 17h). Both context dynamics are adapted from those in [35]. Slice 3 and 4, in turn, emulate IoT-serving vBSs with constant loads when they are operative.

Figure 3-13 depicts the distribution of the throughput performance (left) and the computing resource savings (right) of AIRIC, SIRA and the optimal oracle. Like before, SIRA provides around 5% higher CPU savings in average but incurs almost 25% throughput loss over the 5 days consequently. Conversely, AIRIC performs very closely to the oracle, with no throughput loss and around 17% overall computing resource savings, which validates AIRIC for realistic scenarios.

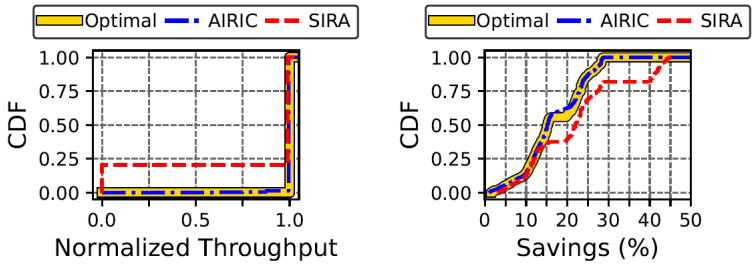


Figure 3-13: Resource allocation in vRAN - Dynamic context profiles based on realistic traces.

3.3 AI/ML and data-driven strategies for energy-efficient 5G carrier on/off switching

Energy-efficient control of RUs by switching cells on and off is a key use case in O-RAN's energy-saving optimizations [3], as these components significantly impact the network's overall energy consumption. As was introduced in [1], the strategy defined in BeGREEN is motivated by the analysis of a real measurement dataset from a Spanish MNO, which contains data from 2G, 3G, 4G, and 5G cells. In this initial analysis, we found a clear correlation between 5G load and energy consumption, and between both KPIs and time (e.g., time of day and day of the week), what opens the door to the definition of non-RT intelligent on/off strategies driven by ML-based predictors. In particular, due to the reported low load of 4G and 5G cells during off-peak hours, we considered a use case based on switching on/off 5G carriers and offloading the traffic to the 4G carriers in the same site and sector.

In following sections, we provide an analysis of the energy saving opportunities and gains, the possible impact on cell and UE performance, and the initial validation of selected on/off strategies. The analysis and validation are mainly focused on the data of a specific urban and high loaded cell. However, in BeGREEN D4.3 we will report a more general analysis based on different types or groups of cells (e.g., urban and suburban). In addition, we will evaluate the designed AI/ML and data-driven strategies and provide the final solution design. Note that the final objective is to implement this solution in the Intelligence Plane and use it to validate its design. Indeed, Section 3.7 provides initial results related to the demonstration done at the 2024 EuCNC & 6G Summit.

3.3.1 Solution design

As mentioned, we have access to two complete months of data from several sites that extend through urban and suburban zones located in a specific Spanish region. Concretely, the number of sites is 70, and each of them has three sectors accounting for a total amount of 210 cells. The list of KPIs is around 1300 for 4G and around 670 for 5G. The granularity of the dataset is of 15 minutes, so most of the KPIs in the list are average values over an interval of 15 minutes. These KPIs are available for the following 5G and 4G carriers:

- 4G carriers: 700 MHz, 800 MHz, 1800 MHz, 2100 MHz and 2600 MHz.
- 5G carriers: 700 MHz, 2100 MHz and 3500 MHz.

It is worth mentioning that the 700 MHz and 2100 MHz carriers of the 4G and 5G technologies are deployed in Dynamic Spectrum Sharing (DSS) mode and, therefore, share bandwidth and radio equipment. The bandwidth for the DSS carriers and the 4G 800 MHz carrier is equal to 10 MHz. The 4G 1800 and 2600 carriers have 20 MHz bandwidth, and the 3500 MHz 5G carrier has 100 MHz of bandwidth.

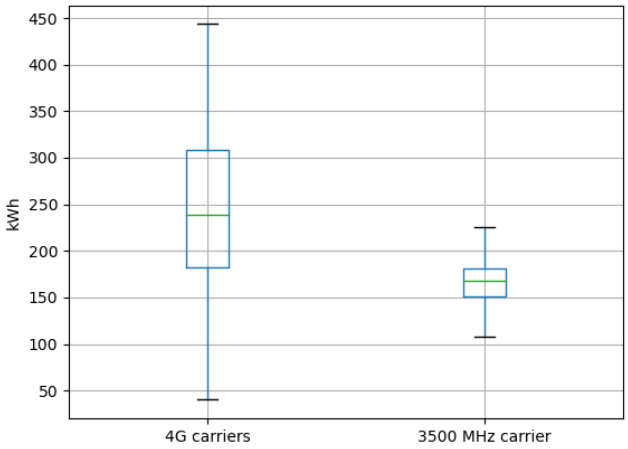


Figure 3-14: 5G carrier on/off - average of the aggregated energy consumption per day and site: 3500 MHz carrier vs rest of the carriers

As has been introduced, the considered energy saving approach is based on switching off the 3500 MHz 5G carrier whenever its traffic can be offloaded to the 4G carriers in the same sector. This approach provides two main benefits. First, it increases the energy efficiency of the 4G carriers by increasing the utilization of available resources. Second, and most importantly, it reduces the overall energy consumption in the network by switching off the 5G carriers when they are under low load. Indeed, as can be concluded from the analysis of the dataset, the 3500 MHz 5G carrier is the one consuming the most energy. Indeed, the aggregated energy consumption per day and site is almost comparable to the consumption of the rest of the carriers on average, as depicted in Figure 3-14. The data dispersion is due to the different configuration of the available sites, which have a variable number of carriers and sectors.

In addition to the aggregated energy consumption per day, the dataset also reports the aggregated energy consumption of the three sectors of the 3500 MHz 5G carrier in each of the sites (Wh, each 15 minutes). This KPI allowed us to learn the trend of the energy consumption at different timescales (days, weeks, and months) and how it was correlated with the average load of the three sectors (percentage of occupied PRBs). As depicted in Figure 3-15, the correlation is evident. Additionally, it clearly illustrates the influence of baseline energy consumption on the overall KPI value, as the variation due to load ranges only from 400 Wh to 550 Wh. This indicates that baseline consumption is significantly higher than the variation in energy due to traffic demand. Therefore, the impact of on/off switching strategies on energy savings will be very notable.

Once established the objective and the potential benefits of the envisioned strategy, we focused on evaluating how many opportunities are available in terms of traffic offloading, i.e., the percentage of time that a 5G carrier could be switched off and its traffic offloaded to the 4G sectors. To do so, we considered a specific site, which is found on an urban zone of a big Spanish city. We have chosen this site because its high traffic demand, what means it could be considered a worst-case scenario, i.e., a scenario with a low number of opportunities to save energy. Figure 3-16 and Figure 3-17 show, respectively, the load pattern of the 5G carrier and the aggregated 4G carriers of a specific sector of this site, during a week. Note that the 4G plot shows the aggregated traffic pattern, which has been calculated as the demand on all the 4G carriers over the sum of total resources of those carriers.

According to these KPIs, for each instant of time, we can evaluate if the 5G load can be offloaded to the 4G carriers. To achieve this, we first need to compute an equivalent PRB value or occupation for all the bands and technologies, by considering:

- The bandwidth of the carriers, i.e. 100 MHz in the case of 3500 MHz carrier versus 10MHz or 20MHz of 4G carriers. This translates into a higher number of available PRBs in the case of 5G.
- Sub Carrier Spacing (SCS) of the technologies, i.e. 30 kHz of 5G versus 15kHz in the case of 4G. This translates into half shorter slot duration in the case of 5G.
- Duplexing techniques of the technologies, i.e. TDD with a pattern of DDDSU in the case of 5G and Frequency Division Duplex (FDD) in the case of 4G. This translates in a lower efficiency of 0.6 for the downlink direction in the case of 5G.

According to these considerations, we can compute a number of PRBs per millisecond (ms) as follows:

- 4G 700 MHz, 800 MHz, 2100 MHz: $\frac{10\ MHz}{15\ kHz \times 12 \frac{subcarriers}{PRB}} \times \frac{1}{1ms} \cong 50\ \frac{PRBs}{ms}$
- 4G 1800 MHZ, 2600 MHz: $\frac{20\ MHz}{15\ kHz \times 12 \frac{subcarriers}{PRB}} \times \frac{1}{1ms} \cong 100 \frac{PRBs}{ms}$
- 5G 3500 MHz: $\frac{100\ MHz}{30\ kHz \times 12 \frac{subcarriers}{PRB}} \times 0.6 \times \frac{1}{0.5ms} \cong 330 \frac{PRBs}{ms}$

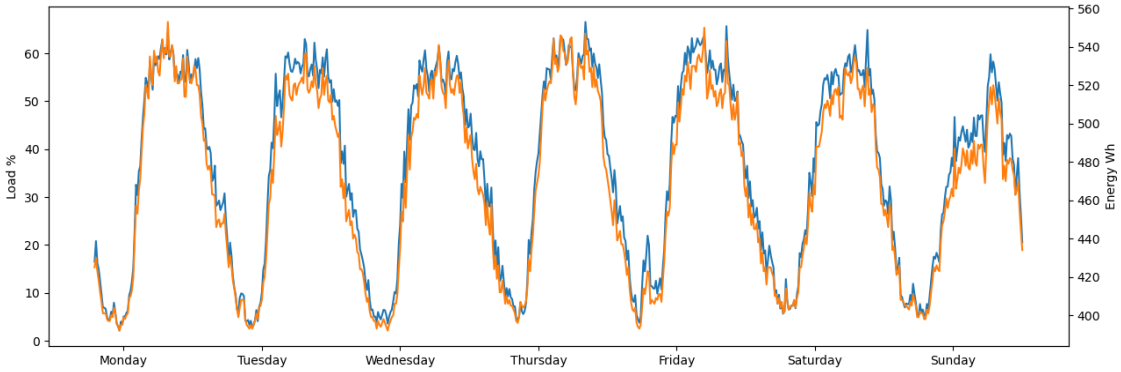


Figure 3-15: 5G carrier on/off - correlation of load (% of PRBs, blue), consumed energy (Wh each 15 mins, orange)

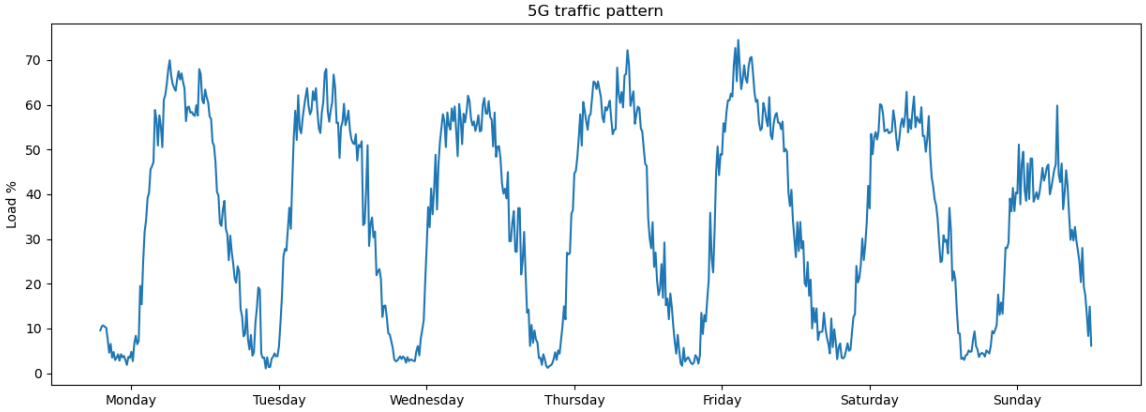


Figure 3-16: 5G carrier on/off - 3500 MHz 5G carrier load pattern (% of utilised PRBs)

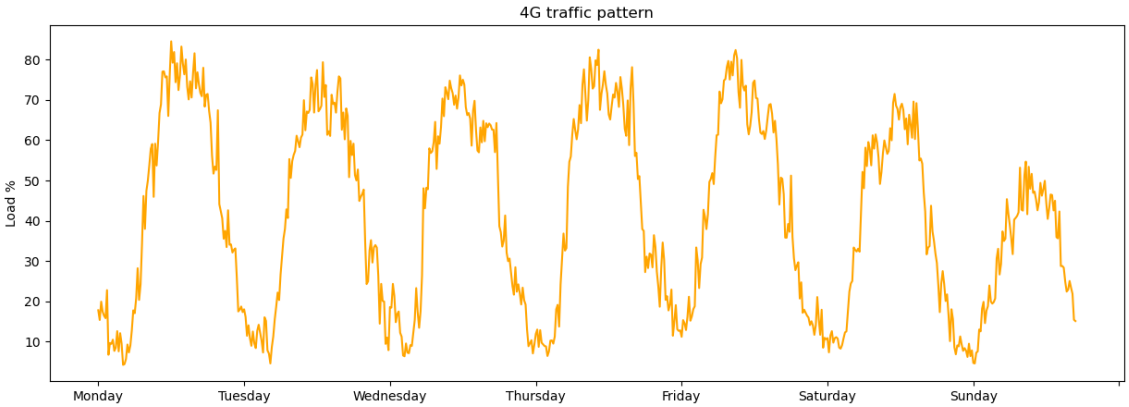


Figure 3-17: 5G carrier on/off - 4G carriers aggregated load pattern (% of utilised PRBs)

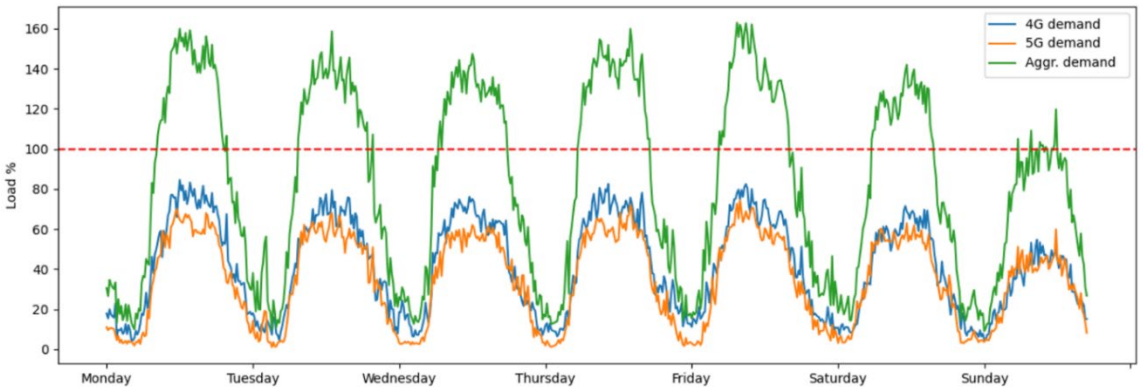


Figure 3-18: 5G carrier on/off - Traffic offloading opportunities – Example

Once obtained the equivalence between 5G PRBs and 4G PRBs, we can analyse, according to the carrier utilisation reported in the dataset, the energy saving opportunities that are available when applying the traffic offloading strategy.

Figure 3-18 exemplifies this strategy for a specific site, sector and week of the dataset. The aggregated demand depicts the percentage of 4G resources occupied when considering the sum of 4G and 5G carrier loads, once computed the PRB equivalence. Whenever the aggregated load is below 100% it means that the 5G carrier could be switched off without suturing the 4G carriers, whenever it is above 100% the 5G carrier needs to be active to allocate all the traffic demand. Note also that 4G and 5G traffic demands are indeed strongly correlated. According to these results, which correspond to a high loaded site, we can expect numerous energy saving opportunities in a real scenario. Indeed, in this case we have calculated that the percentage of week time we could switch off the 5G carrier of this sector was around the 56%. We must recall that these results are for this concrete site and sector, but some other sites found in suburban zones exhibit lower traffic demand patterns. This would translate into a higher number of energy-saving opportunities. This analysis will be reported in BeGREEN D4.3.

Although the benefits in terms of energy consumption and efficiency of this offloading strategy are clear, the impact on the QoS needs also to be considered. First, depending on how this strategy is implemented, it will be more prone to wrong decisions leading to the saturation of the 4G carriers, and which might cause that the offloaded data volume cannot be entirely allocated. More conservative safety margins or thresholds in the 4G carriers might be used to minimize the probability of this condition to occur, but this would also decrease energy savings. In section 3.3.2.3 we will analyse how different strategies, e.g. based on daytime, load thresholds or traffic predictions, perform in terms of energy consumption reduction and 4G carrier saturation. On the other hand, even in non-saturated cases, traffic offloading from 5G to 4G will imply a penalty in the average throughput experienced by the UEs. First, due to the increase of load in the 4G carriers, UEs using this technology may experience a degradation in their QoS. Figure 3-19 illustrates how the KPIs of carrier occupancy and average throughput per UE are strongly correlated, leading to lower throughputs when the load increases.

Figure 3-20 illustrates this correlation for the case of a specific site and two specific 4G and 5G carriers, which shows a more linear relationship. As future work, we plan to use this data to infere the expected throughput (and determine the throughput loss) of the 5G UEs when offloading them to 4G cells.

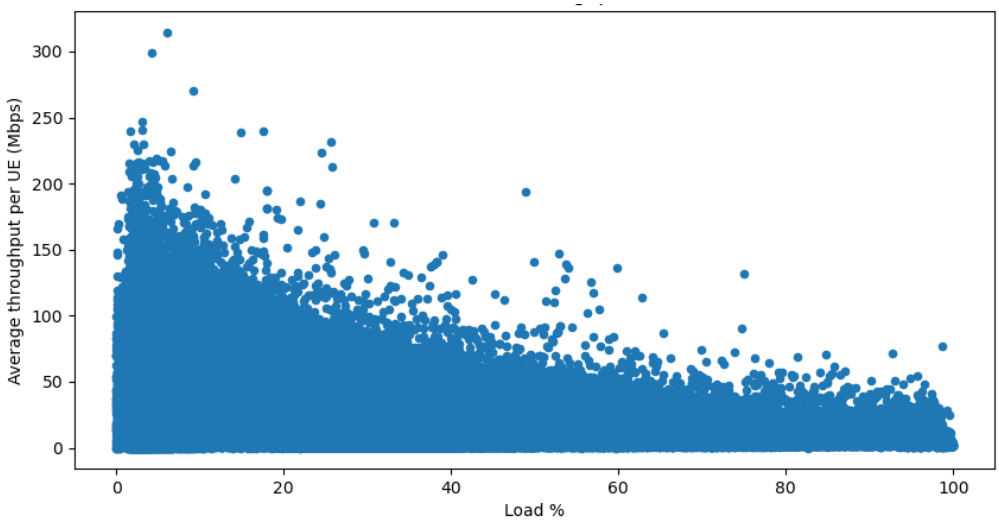


Figure 3-19: 5G carrier on/off - Correlation between load (% of PRBs) and average throughput per UE KPIs – All 4G carriers in sites with active 5G carrier

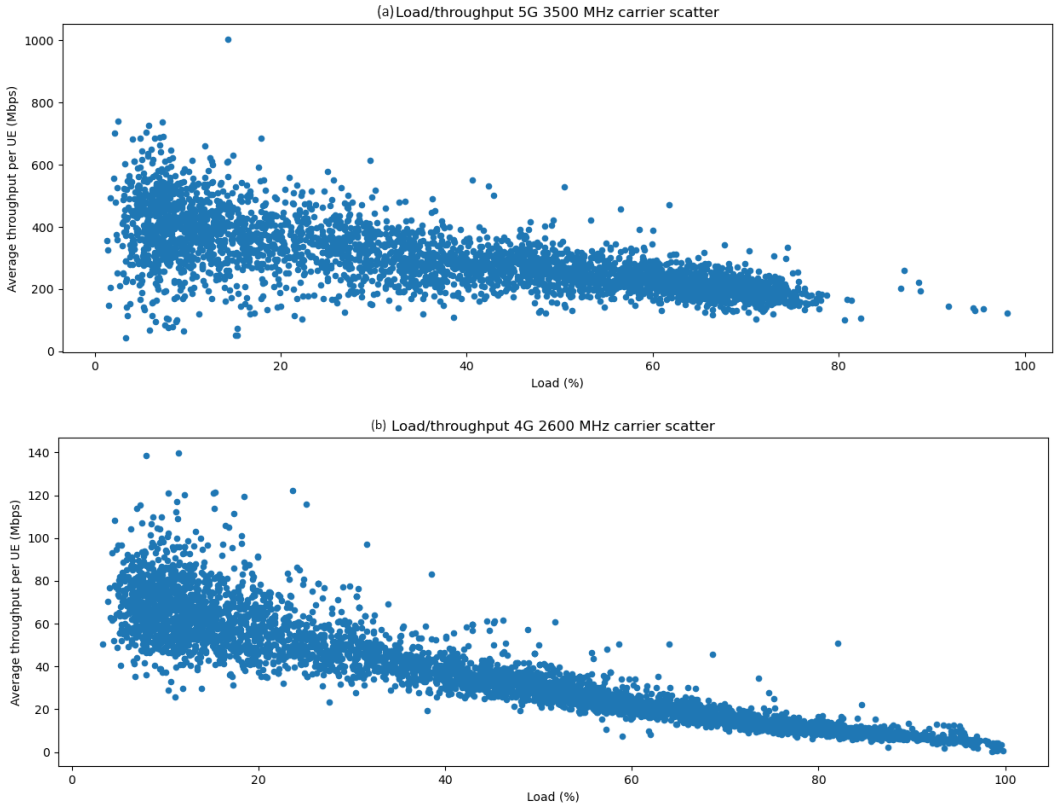


Figure 3-20: 5G carrier on/off - Correlation between load (% of PRBs) and average throughput per UE KPIs – (a) 5G 3500 MHz Cell and (b) 4G 2600 MHz Cell

Secondly, although we are considering offloading the 5G traffic to the 4G cells in the same sector to minimize the impact on MCS, due to the different characteristics of 5G and 4G, such as the channel bandwidth, 5G UEs will experience a reduction of the PHY rate, what will impact their maximum achievable throughput. Figure 3-21 shows the difference between the measured average throughput of 4G and 5G carriers for the same site and sector that we considered in the previous analysis. As depicted in the figures, the difference is notable and is indeed higher in low loaded conditions when such traffic offloading strategies will occur. However, it should be highlighted that the KPI being reflected in the figures, although being denoted as average throughput per UE, only considers the time to transmit a data burst excluding the data transmitted in the slot when the buffer is emptied [25]; i.e., it is a measure more related to achievable PHY rate than to the real average throughput that will experience usual UE applications. Therefore, it should not be used as a measure of QoS degradation due to traffic offloading.

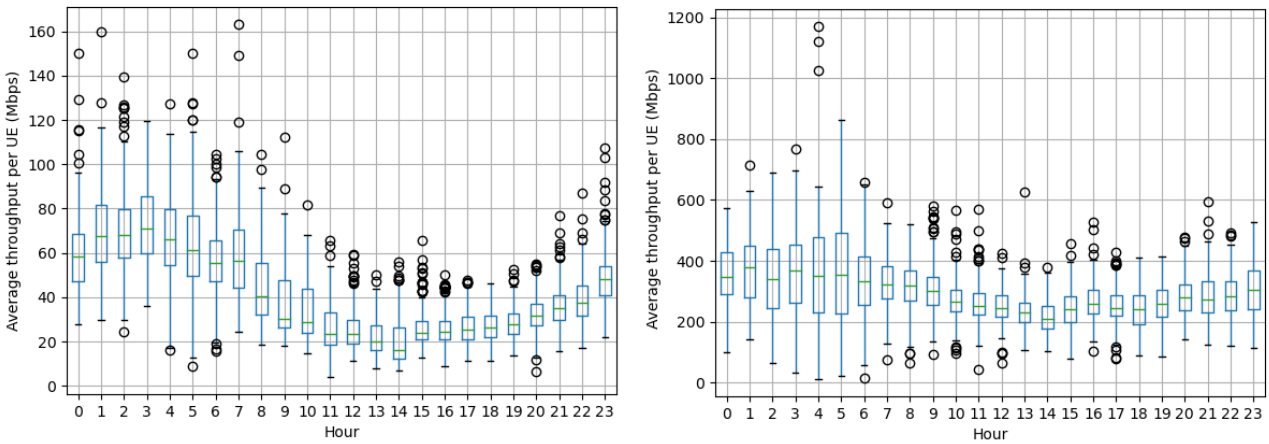


Figure 3-21: 5G carrier on/off - Average throughput per UE KPI during a day - 4G (left) and 5G (right).

The concrete analysis and definition of the QoS penalty, along with how this trade-off will be considered in the designed energy efficiency strategy, will be detailed in the BeGREEN D4.3. In the following section, we will present the initial validation of different on/off switching strategies, focusing on the saturation of 4G carriers as the main negative effect.

3.3.2 Initial evaluation

This section introduces the on/off switching strategies that have been considered at this stage of the project, the motivation behind each of them, and how AI/ML can be used to improve them.

3.3.2.1 Cell energy consumption prediction

Due to the correlation between load and energy consumption shown in Figure 3-15, we first considered the implementation and validation of an energy consumption predictor with the objective of predicting the amount of energy savings when switching off specific sectors. To this objective, we trained an XGBoost Regressor [45], using as input data the load of the sectors and the datetime. XGBoost is a well-known python library[12], which aims to provide easy-to-use models that can be tuned to fit the objective data. Despite providing several ML models, we used the XGBRegressor, which is a gradient boosting regressor widely used to address time series forecasting. Rather than the ML model, one of the key points in the process of developing the complete solution, was the pre-processing stage. Considering that we have access to data from several sites, we decided to do a cluster-based training. We selected sites with similar traffic demand patterns found in an urban area, and then trained a common model with all the data of the selected sites. Since we had access to two complete months of data, we decided to train the model with 45 days and test it with 15 days of data. As expected, due to the clear relationship between load and energy, the results were significantly good. We attained an R^2 score of 0.97 and a mean absolute error of 5.5 Wh. Figure 3-22 shows a comparison between real and predicted energy consumption for a specific 5G node during a week.

Additionally, we used the model to get the estimated baseline consumption of the 5G nodes, since this value was not reported in the dataset. To this end, we used as input a simulated zero-traffic scenario for all the sectors of the cells, obtaining the following average results:

- Node baseline energy consumption: 381 Wh every 15 minutes (1524 Wh per hour).
- Sector baseline energy consumption: 127 Wh every 15 minutes (508 Wh per hour).

These results are aligned with the dataset analysis introduced in the previous section. Unfortunately, in the case of the 4G carriers we do not have access to energy-related KPIs with a 15 minutes granularity, but only to aggregated values per day (see Figure 3-14).

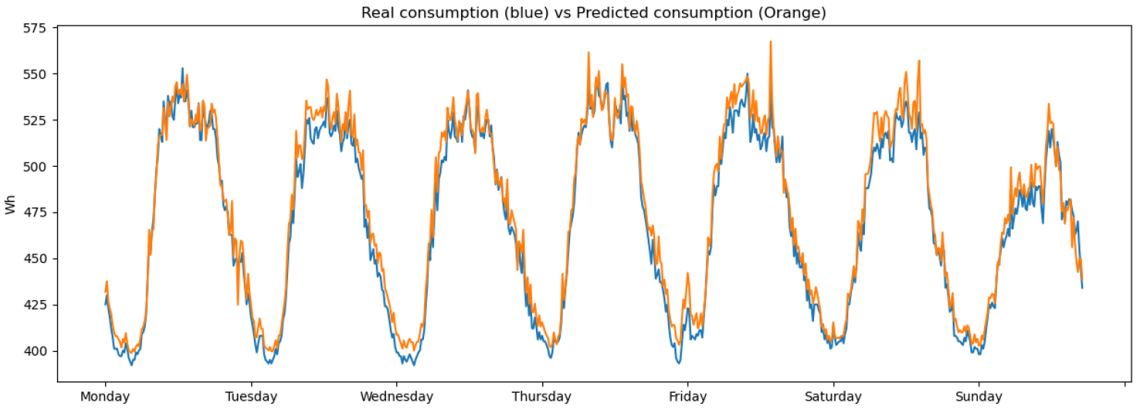


Figure 3-22: 5G carrier on/off - Energy consumption predictor results

[12] https://xgboost.readthedocs.io/en/stable/

This limits our estimation on the energy saved due to traffic offloading, which can be expressed as:

$$EC_{saved} = EC_{5G} - \Delta EC_{4G} = ECB_{5G} + \Delta EC_{5G} - \Delta EC_{4G}, \text{where } \Delta EC_{5G} > \Delta EC_{4G}$$

$$EC_{saved} > ECB_{5G}$$

where EC_{5G} represents the energy consumed by the 5G carrier, which is composed by the baseline consumption ECB_{5G} and the consumption due to the actual load ΔEC_{5G}, and ΔEC_{4G} represents the consumption due to the load increase in the 4G carriers. As previously introduced, actually we don't have a way to estimate or compute ΔEC_{4G}, which is left for future work. But due to the higher energy consumption of 5G carriers, we can ensure that the provided energy savings will be at least as much as the baseline consumption of the 5G carriers.

3.3.2.2 Evaluated strategies

To evaluate different strategies to control the 5G carrier on/off switching, we first considered an optimal case based on the traffic offloading opportunities analysed in Figure 3-18. In particular, for each instant of time in the dataset, off decisions were made when the PRBs of the 5G cell could be offloaded to the 4G cell without leading to saturation. We then used the results of this optimal strategy to benchmark the other strategies.

First, we defined three simple strategies respectively based on the hour of the day, on the current aggregated load of the 4G carriers, and on the current load of the 5G carrier being monitored. In the case of the strategies based on the load, we considered a threshold of 40% to determine if the 5G carriers should be switched on (load higher than the threshold) or off (load lower than the threshold). This threshold was determined according to the observation of the 4G and 5G load trends depicted in Figure 3-18.

In addition, we also considered two strategies based on ML models. First, using 5G load predictions instead of actual load values, aiming at improving the decision-making process of the strategy based on the 5G load. To this end, an XGBRegressor was trained to predict the future occupancy of 5G carrier sectors. The main difference between the energy and the load predictors is that the load model is used to predict the carrier load of the next period of 15 minutes, while in the energy case it estimated the energy consumption of the past 15 minutes.

We carried out a filtering process of the available KPIs to be used as the model's input. As reported in the previous section, the dataset contains hundreds of KPIs, but many of them were not giving any relevant information to the regressor. After a feature importance analysis, we selected a dozen of KPIs related to the average throughput, number of UEs, UL and DL load, and datetime. As in the energy case, the model was trained with the same cluster of cells. The difference in this case is that it was trained with just 20 days and tested with 10 days (i.e., one complete month). The current load predictor being used attained an R^2 score of 0.93 and a mean absolute error of 2.9% (percentage of PRBs). Figure 3-23 shows a comparison between real and predicted sector occupancy for a specific 5G node during a week.

In addition to the predictor, we also developed a specific AI/ML strategy based on a classifier. The motivation of this classifier was to take on/off decisions just by considering 5G KPIs, i.e. the 5G load and the number of connected users, as an alternative to the strategy based on the 5G load predictions. This would eliminate the need for 4G data to make decisions, thereby saving network resources and enabling the 5G cells to independently decide when to switch on and off. As introduced in the previous section and illustrated in Figure 3-18, energy saving opportunities relies on the occupancy of both 4G and 5G carriers, which in this dataset an are very correlated.

The classifier used was a Logistic-Regressor classifier from the Scikit-Learn python library. This model was chosen after a selection process, on which several models such as Decision Trees (DTs), Random Forests, Support Vector Machines (SVMs) etc., were tested.

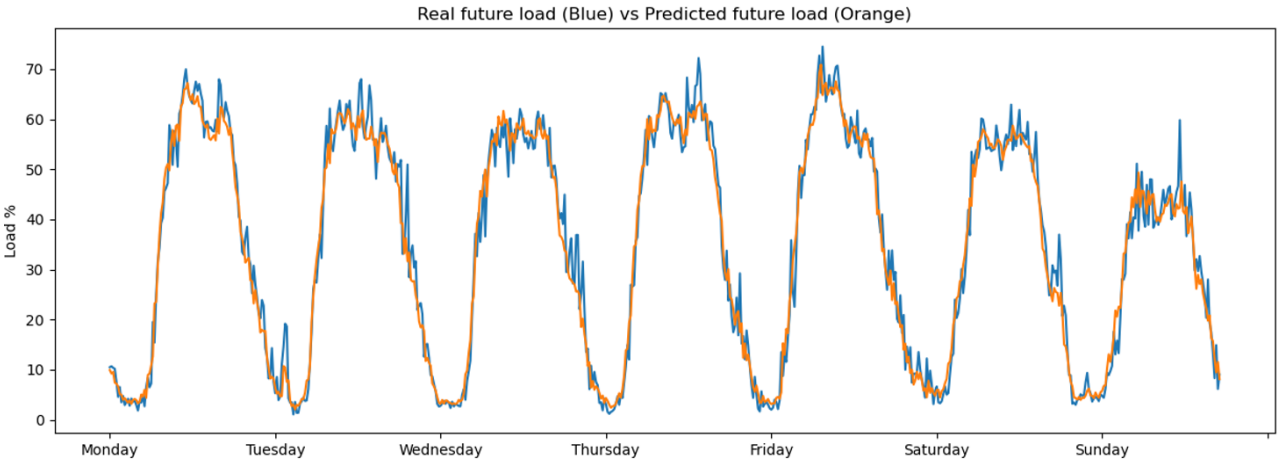


Figure 3-23: 5G carrier on/off - 5G carrier load predictor results

Although the SVM provided slightly better accuracy (0.5%-0.75%), it took a lot more to be trained when compared to the Logistic Regressor, which just took in the order of seconds, leading to a better efficiency. We also changed the default optimizer to the Newton-Cholesky optimizer which, in the documentation of the library, appeared as the most appropriate one in those cases in which the number of temporal samples, *N*, was much higher than the input features, *P*. In our case, in the training data, we had *N* equal to 130.000 (45 days of data with 15 minutes granularity and a cluster of 30 sectors), and *P* equal to 2 (average number of RRC connected UEs and average load of the 5G carrier). In fact, changing this optimizer translated into an increase of the 4%-5% in the final score or accuracy of the model, which was finally equal to 94%.

To generate the model, we trained the classifier in a supervised-learning way, using the decision of the optimal approach as the ground truth. The classifier was trained with approximately the 70% of the data of the dataset and tested using the remaining 30% of the data. The penalty of wrong decisions (i.e., incorrect on or incorrect off status) was the same. We found that most of the errors were in the switch on decision rather than in the switch off. In the evaluation subsection, it will be shown that this behaviour translates into a higher number of missed opportunities (i.e., wasted energy), but to a lower number of errors (i.e., saturation of 4G carriers). This conservative behaviour was caused by the uncertainty about the 4G load. This impacted specially cases where the 5G load was high (i.e., peak hours) and the 4G load was not totally correlated (i.e., lower than expected).

Note that in all cases, we avoided strategies that combine 4G and 5G KPIs. This decision was made to minimize data requirements and simplify the strategies for evaluation in this first phase. Since these design choices also impact the system's energy consumption due the need of data sending, storing and processing, we wanted to first assess whether these simple strategies could achieve significant energy savings. Nevertheless, in future deliverables we will also consider additional strategies combining 4G and 5G data, evaluating the trade-off between the energy consumption of the model and the achieved energy savings.

3.3.2.3 Initial evaluation of strategies

In this section, we evaluate different on/off switching strategies according to the data from the measurements dataset from a specific sector and during a specific week. As previously introduced, we considered a high loaded site located in the centre of a big city. The evaluated strategies are detailed as follows:

- Optimal strategy: it knows in advance the load of the cells in each instant and avoids 4G cell saturation during the off phase. Used to benchmark the other strategies.
- Hour strategy: it switches off the 5G carrier during the night, from 1 AM to 9 AM, and turns it on during the other hours of the day.

- 4G load strategy: Reactive decision according to current load. It switches off the 5G carrier whenever the aggregated load of all the 4G carriers is below the 40%; otherwise, it switches it on.
- 5G load strategy: Reactive decision according to current load. It switches off the 5G carrier whenever the 5G carrier load is below the 40 %: otherwise, it switches it on.
- 5G predicted load strategy: Proactive decision based in the load predictor presented in Section 3.3.2.2. Whenever the prediction in the 5G carrier load is below 40% it switches it off; otherwise, it switches it on.
- Classifier strategy: it directly relies in the decision taken by the classifier introduced in 3.3.2.2, which relies in the 5G carrier load and the number of connected UEs.

We decided to evaluate the performance of each strategy according to the following metrics obtained from the dataset. Note that since we are limited by the time granularity of 15 minutes, we consider how each taken on/off decision in a given time impacted the results of the next 15 minutes period.

- The percentage of correct switch on and switch off decisions in relation to the decision taken by the optimal strategy.
- Percentage of total time spent in the off state.
- The average saturation in PRBs in the 4G carriers due to wrong switch off decisions (i.e., 5G load cannot be fully offloaded).
- The aggregated energy consumption saved due to correct switch off decisions (according to total 5G carrier consumption).
- The aggregated energy consumption wasted due to missing a feasible switch off decision (according to total 5G carrier consumption).
- The aggregated energy consumption saved due to correct switch off decisions according to baseline 5G carrier consumption.
- The aggregated energy consumption wasted due to missing a feasible switch off decision (according to baseline 5G carrier consumption).

Table 3-2 summarizes the obtained results and how the different strategies performed compared to optimal strategy. First, it is remarkable that for all the strategies the amount of energy saved is very significant since the analysed 5G cell could remain off a significant period of time. Recall that this is for one week and one sector, and moreover, the data belongs to a high-loaded site in the dataset. Secondly, due to the high correlation between load and daytime, all strategies behave similarly regarding correct on decisions during peak times. As expected, the 4G load strategy results in the lowest PRB saturations by considering the load of the 4G carriers. Regarding off decisions, the hour-based strategy, being more conservative, achieves the least amount of energy savings. In contrast, the 5G load-based strategy is more aggressive and attains the highest energy savings for both the actual and predicted load strategies. Finally, since the classifier was trained to avoid PRB saturations by only considering 5G load, it performed more conservatively, interestingly yielding results very similar to the 4G load-based strategy.

Table 3-2: 5G carrier on/off - Initial Validation of the 5G Carrier on/off Switching Strategies

Strategy	Correct ON Decisions	Correct OFF Decisions	Time in OFF State	PRB Sat.	Energy Saved (total)	Energy Wasted (total)	Energy Saved (baseline)	Energy Wasted (baseline)
Optimal	100%	100%	56.1%	0.0%	217 kWh	0	191 kWh	0

Hour	96%	64%	36.0%	7.9%	133 kWh	85 kWh	123 kWh	68 kWh
4G Load	**99%**	85%	47.6%	**3.4%**	181 kWh	36 kWh	162 kWh	29 kWh
5G Load	98%	**91%**	51.0%	6.2%	195 kWh	22 kWh	174 kWh	17 kWh
Prediction	**99%**	**91%**	**51.2%**	7.4%	**196 kWh**	21 kWh	175 kWh	16 kWh
Classifier	**99%**	83%	46.5%	5.5%	176 kWh	41 kWh	159 kWh	32 kWh

These initial results indicate that simple strategies such as considering the actual 4G or 5G load could lead to huge benefits. However, we still need to characterize the impact that they will have on the QoS of the 4G and 5G UEs, what will be the main trade-off to be considered. In addition, these results considered just a single site, while the behaviour of these strategies in different areas (e.g. suburban and urban) may differ. In addition to the presented AI/ML-based strategies, other solutions such as combining 4G and 5G KPIs, determining the load threshold according to the site and daytime, or predicting the impact on QoS, may help to optimize the energy efficiency of the network. These aspects will be addressed in future deliverables. Also, we plan to evaluate the impact of the different AI/ML strategies on the energy consumption of the Intelligence Plane.

3.4 AI/ML-based algorithmic solutions for relay-enhanced RAN control

This section presents a description of the proposed algorithmic solutions for the different relay control functionalities described in section 2.2.4. For each of these functionalities, a detailed workflow is presented to illustrate the interaction of the involved entities of the network. Then, a description of the algorithmic solution is presented, and an initial evaluation of the proposed algorithms is provided. Section 3.4.1 focuses on the detection of coverage holes functionality by using a clustering methodology. Then, Section 3.4.2 presents a solution for the placement of fixed relays with the objective of addressing the identified coverage holes. Section 3.4.3, deals with the identification of UEs that may be good candidates to become RUE and act as a relay between the gNBs and neighbour UEs that may be located at the coverage holes. Finally, section 3.4.4, proposes a solution for the activation/deactivation of the different relays/RUEs by means of Reinforcement Learning with the objective of addressing the coverage holes and reducing the overall energy consumption. Several initial results have been presented for each of the proposed solutions.

3.4.1 Detection of coverage holes

This process aims to identify geographical regions with a relatively high traffic demand and poor coverage. As depicted in Figure 3-24, it is activated by the RFM rApp in the Non-RT RIC for specific gNBs with a too high drop call rate (see section 2.2.4.2). Then, the CHD process is executed at the AI Engine. This process collects the required measurements from the measurements datalake in the AI Engine and executes a clustering algorithm to identify the geographical location of the coverage holes (see steps 5-6).

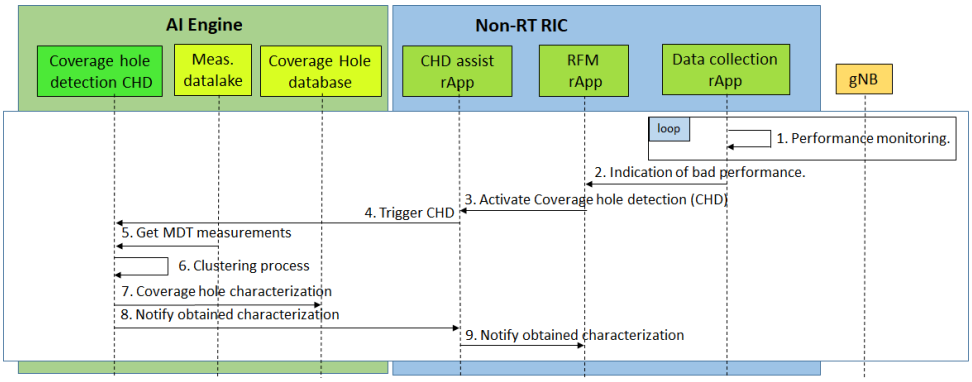


Figure 3-24: Relay control - coverage hole detection workflow

This clustering algorithm is described in Section 3.4.1.1. The resulting coverage hole characterization is stored in the coverage hole database located in the AI Engine. Finally, the end of the clustering process is notified to the RFM rApp (steps 8-9).

3.4.1.1 Coverage hole detection algorithm.

The clustering process described in step 6 in Figure 3-24 is executed in two phases. The first phase aims to identify geographical regions with potential coverage holes, while the second phase consists of a validation of the identified coverage holes. A detailed description of the process is presented in the pseudo-code below (see Algorithm 3-1).

Phase 1: Identification of potential coverage holes

This process analyses the RSRP measurements collected during *D* days at a specific gNB. These measurements are available in the AI Engine datalake. In order to do this analysis, the measurements of each *d*-th day (with *d=1,…,D*) are split in *N* subsets. Each *n*-th subset (with *n=1,…,N*) contains the measurements collected during a period of time τ in the different *N* periods of the day. Each of these measurements contains the RSRP value, the time and geographical location of the measurement and the associated serving gNB. Then, the coverage hole detection process is run iteratively for each *n*-th period of each *d*-th day (see lines 1-2 in Algorithm 3-1). In each iteration, the collected measurements are filtered to obtain a list of measurements with a RSRP lower than a specific threshold Th_{RSRP} (see line 3 in Algorithm 3-1). The resulting set of measurements are used as input for a clustering process which groups the measurements in different clusters according to their geographical location (see line 4 in Algorithm 3-1). Different clustering methodologies can be considered, such as K-means, DBSCAN, etc. The proposed approach considers the DBSCAN methodology. The key idea of this clustering methodology is to identify a group of samples inside a neighbourhood of radius ε with a minimum number of samples *min_samples*. The considered clustering methodology is described in BeGREEN D4.1[1]. The output of the clustering process for a given *n*-th period in a specific *d*-th day is a list of I clusters $\{C_1(d,n),\ldots, C_i(d,n),\ldots, C_I(d,n)\}$ that represent the potential identified coverage holes. Each *i*-th cluster $C_i(d,n)$ is characterised with a circle centered at the cluster centroid $Centr(C_i(d,n))$ with a radius equal to the cluster radius $Rad(C_i(d,n))$. This radius is calculated as the distance between the cluster centroid and the furthest geographical location that belongs to the cluster. Additionally, a cluster is also characterised with a list of the geographical locations that belong to the cluster and the total number of measurements associated to the cluster $num\_meas(C_i(d,n))$. Only clusters with a total number of measurements higher than a threshold Th_{num_meas} are considered as valid (see lines 5-7 in Algorithm 3-1). After running the clustering process in each of the *N* periods for all the *D* days, a list of valid clusters *C* is obtained. This list contains all the valid clusters that represent the potential coverage holes that have been identified with the measurements collected during the *D* days of measurements.

Algorithm 3-1: Relay Control - Coverage hole detection algorithm

	# Phase 1: Identification of potential coverage holes in a specific gNB.
1	**For** d=1 with d<D
2	**For** n=1 with n<N
3	Select the geographical locations with RSRP<Th_{RSRP}
4	Run a clustering process of the selected measurements.
5	**For** i=1 with i<I
6	**if** num_meas(Ci(d,n)) / τ > Th_{num_meas}
7	C <- Ci(n) #Add the identified cluster Ci(n) in the list of all the identified clusters C.
	# Phase 2: Validation of the identified coverage holes
8	**For** all i-th clusters in the list C
9	**For** all j-th clusters (with i!=j) in the list C
10	**If** $\alpha_{i,j} > Th_{overlap}$

11	num_times_cluster(i)++
12	Merge cluster i-th and j-th and recalculate cluster centroid, radius and number of measurements.
13	Remove cluster j from the list C
14	**For** all i-th clusters in the list C
15	Compute repetitiveness = num_times_cluster(i) / (N·D)
16	Build matrix ***P*** and identify temporal patterns of the presence of each validated coverage hole.

Phase 2: Coverage hole validation

The validation of the identified potential coverage holes is done by comparing the clustering results obtained in the *N·D* iterations in phase 1. In order to consider a coverage hole as valid, it is necessary that the coverage hole is detected very frequently in different *n*-th periods of duration τ in the different *d*-th days. For this purpose, the algorithm compares the different potential coverage holes available in the list of clusters *C*. Two clusters *i*-th and *j*-th in the list *C* are considered to represent the same coverage hole if the overlap $\alpha_{i,j}$ between them is higher than a specific threshold (i.e. $\alpha_{i,j} > Th_{overlap}$). This overlap $\alpha_{i,j}$ can be measured according to the distance between their centroids and the summation of their cluster radius according to:

$$\alpha_{i,j} = 1 - \frac{dist[centr(C_i), centr(C_j)]}{rad(C_i)+rad(C_j)}$$

where *dist()* represents the Euclidean distance, $centr(C_i)$ and $centr(C_j)$ represent the centroid of cluster C_i and C_j, and $rad(C_i)$ and $rad(C_j)$ represent the *i*-th and *j*-th cluster radius, respectively. According to previous equation, $\alpha_{i,j}=1$ means a total overlap between clusters *i*-th and *j*-th, while $\alpha_{i,j}<0$ means no overlap between them.

For each *i*-th cluster in the list *C*, the algorithm iteratively searches a cluster *j*-th that satisfies the previous condition (i.e. $\alpha_{i,j}>Th_{overlap}$) and, in case it is found, the algorithm increases the number of times that cluster *i*-th is repeated in the list *C* (i.e. *num_times_cluster(i)++*), see line 11. In this case, cluster *i*-th and cluster *j*-th are merged into a single one and cluster *j*-th is removed from the list *C* (see line 12-13). The centroid of the merged cluster is calculated as the midpoint between the centroids of clusters *i*-th and *j*-th, and the radius of the merged cluster is calculated as the averaged radius of clusters *i*-th and *j*-th. The number of measurements associated to the merged cluster is determined as the summation of the values for both *i*-th and *j*-th clusters. After running this process for all the elements in the list *C*, the parameter *num_times_cluster(i)* is useful to represent the number of periods with duration τ in which the coverage hole represented by the *i*-th cluster has been identified.

As a result, the output of this process is a list of clusters that contains the following items for each validated coverage hole:

- *Coverage hole centre*: This corresponds to the geographical location the cluster centroid.
- *Coverage hole radius*: This corresponds to the radius of the coverage hole.
- *Number of measurements associated to each coverage hole:* This corresponds to the total number of measurements that have been obtained in the region of the coverage hole. It allows to represent the amount of traffic in the coverage hole.
- *Repetitiveness:* This corresponds to the percentage of time periods with duration τ in which the *i*-th coverage hole was detected. It is calculated as *num_times_cluster(i)/(N·D)*. A repetitiveness of 100% corresponds to a coverage hole that is detected in a specific geographical region in all the *n*-th time periods of all the *D* days.
- *Coverage hole presence matrix P:* This is a *NxD* matrix where each term $p_{n,d}=1$ if the coverage hole was detected at the *n*-th period of the *d*-th day and $p_{n,d}=0$ otherwise. This matrix contains

information about the time periods when the coverage hole was identified and is useful to characterise the temporal patterns of the presence of the coverage hole.

3.4.1.2 Initial evaluation

The proposed coverage hole detection methodology has been executed in a realistic scenario in a University Campus in *Universitat Politècnica de Catalunya* (UPC) [5]. The considered region is a 350 m x 125 m area with 25 buildings of 3 floors. 5G NR coverage on the Campus is provided by three outdoor macrocells of a public MNO in band n78 (3.3-3.8 GHz). This scenario has been modelled by means of a system level simulator that models the geographical location of the different buildings and the propagation loss of the transmitted signal according to [46]. This simulator makes use of real measurements of the time evolution of the number of users located in different geographical regions [47] . The CHD algorithm has been tested using as input a collection of *D=182* days of measurements. The considered time span for each day is between 8:00h and 22:00h, divided in *N=14* time periods of *τ=1* hour. The simulation parameters and the CHD parameters are presented in Table 3-3.

Table 3-3: Relay Control - Simulation Parameters

Parameter	Value	Parameter	Value
D	182 days	**min_samples**	10
N	14	**BS Carrier Frequency**	3.7GHz
τ	1 hour	**BS Channel Bandwidth**	100MHz
Th_{RSRP}	-90dBm	**BS Transmitter power**	35dBm
Th_{num_meas}	2	**BS Transmitted antenna gain**	21dB
$Th_{overlap}$	0.5	**Path loss model**	UMa - 3GPP TR 38901
ε	10 meters	**UE antenna gain**	3dB

Figure 3-25: Relay Control - Validated coverage holes with a repetitiveness higher than 0.25

Table 3-4: Relay Control - Characterization of the Validated Coverage Holes

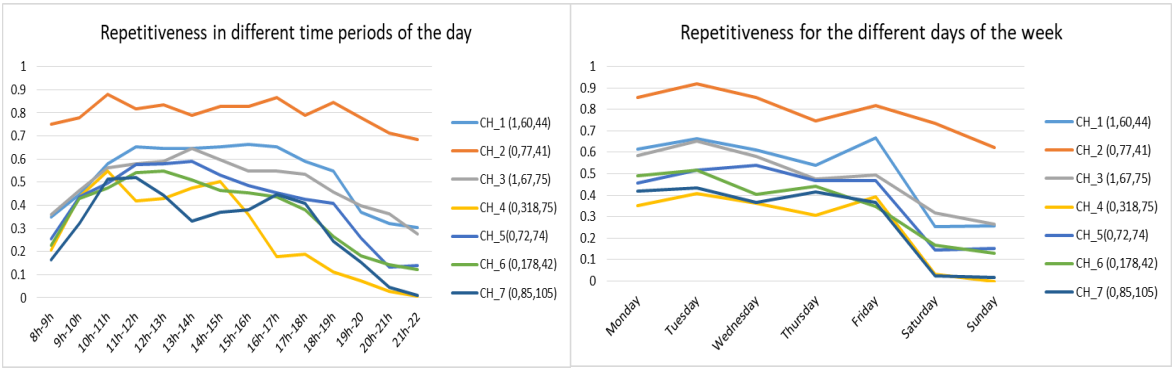
	CH_1	CH_2	CH_3	CH_4	CH_5	CH_6	CH_7
Centroid coordinates	[60,44]	[77,41]	[67,75]	[318,75]	[72,74]	[178,42]	[85,105]
Floor	1	0	1	0	0	0	0
Radius [m]	8.6	8	5.5	9.11	7.13	8.26	5.4

Repetitiveness	0.51	0.79	0.48	0.26	0.39	0.36	0.29

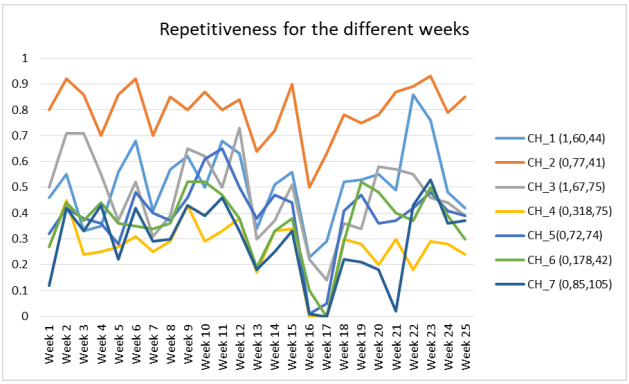
Figure 3-25 shows a map of the considered scenario with the coverage holes identified with a repetitiveness higher than 25% after running the CHD process. As shown, seven validated coverage holes have been identified in different buildings of the Campus.

Table 3-4 illustrates the main coverage hole parameters, namely the centroid coordinates and the floor where coverage hole is detected (ground floor is represented with 0), the coverage hole radius and the coverage hole repetitiveness. It is worth noting a very large repetitiveness in coverage hole CH_2. Note also that coverage holes CH_1 and CH_2 are in the same building but at different floors. A similar situation happens with CH_3 and CH_5.

To give a more detailed characterization of the identified coverage holes, Figure 3-26 shows different statistics of the average repetitiveness in different time scales. In particular, Figure 3-26a presents the average repetitiveness for different hours of the day. As shown, all the coverage holes exhibit a higher repetitiveness between 10:00h and 14:00h that corresponds to the periods of the day with higher number of connected users. Figure 3-26b, shows the average repetitiveness observed in different days of the week. As shown, the coverage holes have a higher presence in the weekdays (i.e. from Monday to Friday) that correspond to the days with higher number of users. Finally, Figure 3-26c, illustrates the average repetitiveness observed in the different weeks of the dataset. Note that the repetitiveness has a relatively large variability depending on the considered week. In particular, a low average repetitiveness is observed in weeks 16 and 17 that correspond to the period of Christmas, when the number of users in the Campus is low. It is worth noting that coverage hole CH_2 has high repetitiveness in most of the time periods at the different time scales while the rest of identified coverage holes have higher repetitiveness variability depending on the considered time period and time scale.



(a) (b)



(c)

Figure 3-26: Relay Control - hourly, daily, and weekly variability of the coverage hole repetitiveness

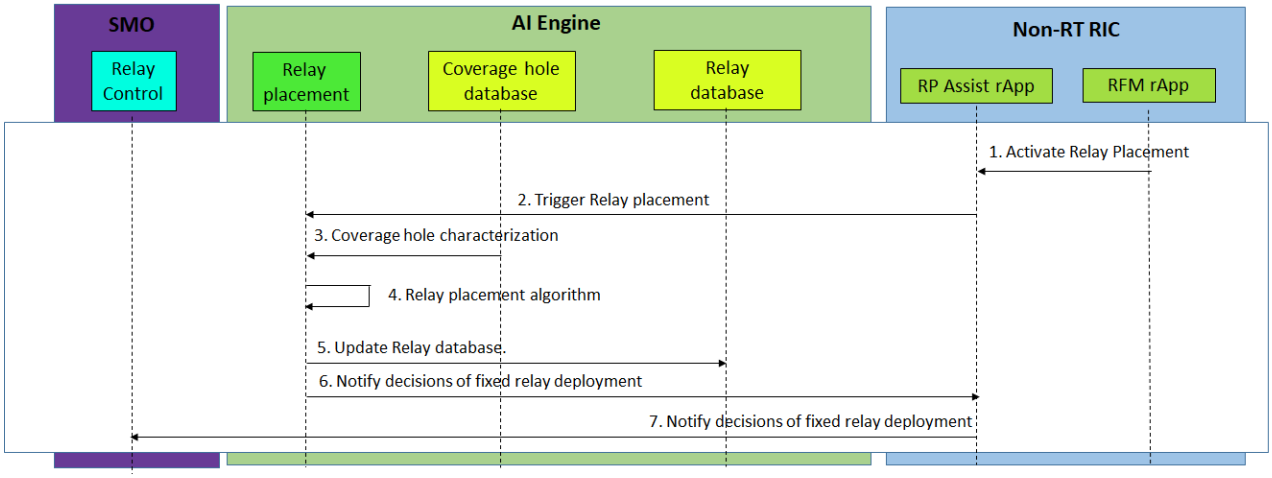


Figure 3-27: Relay control - relay placement workflow

An initial evaluation of the computation time and the associated energy consumption of the Coverage Hole Detection process has been done. As described in 3.4.1.1, the clustering process based on DBSCAN is repeated *N* times every day, each time with the measurements collected in each period of duration τ. In the considered scenario that consists of 3 gNBs and setting *N=14* and *τ=1*hour (see Table 3-3), the result of the Coverage Hole Detection process of the collected measurements during one day can be obtained in approximately 150 seconds. This process has been run with an Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz processor. The average power consumption of this processor is around 25W, leading to an energy consumption of 1.04Wh (3.74kJ) for each day. The energy consumption of the Coverage Hole Detection process is highly dependent on the network size (e.g. the number of gNBs and the size geographical region to be covered). Moreover, the required number of days *D* to be analysed in order to validate the identified coverage holes has also a relevant impact in the energy consumption. It is worth mentioning that the identified coverage holes may be valid for a limited period of time. For this reason, the periodicity in the execution of the Coverage Hole Detection process in each gNB is relevant to guarantee a precise coverage hole characterization but it is also important to assess the algorithm energy consumption. BeGREEN D4.3 will deal with these aspects and their impact in terms of energy consumption.

3.4.2 Fixed relay placement

This process aims to identify the geographical locations to deploy fixed relays with the objective of improving the performance at the identified coverage holes. It is activated by the RFM rApp in the Non-RT RIC when the placement of a new fixed relay is required in the coverage region of a specific gNB. As shown in Figure 3-27 the Relay Placement (RP) process is executed at the AI Engine. It collects information about the coverage hole characterization available in the coverage hole database in the AI Engine. This information is used as input for the relay placement algorithm executed in step 4 (see Figure 3-27). This relay placement process is described in section 3.4.2.1. In case a new fixed relay needs to be deployed, its geographical location and configuration parameters is determined. This information is included in the relay database (see step 5). Finally, the result of the relay placement process is notified to the relay control (steps 6-7) that informs the network operator about the necessity of deploying a new fixed relay.

3.4.2.1 Fixed relay placement algorithm

The fixed relay placement process is executed according to the pseudo-code presented in Algorithm 3-2, which is responsible for deciding the geographical coordinates to deploy the new fixed relay. This is done on the basis of the validated coverage holes characterised in the coverage hole database. For each coverage hole, the search space of possible geographical locations where the fixed relay can be placed is initially defined. The proposed methodology considers a square region of side *x* meters around the centre of the coverage hole (line 4 of Algorithm 3-2). Note that the value of *x* must be higher than the coverage hole

diameter. Note also that the fixed relay must be deployed outside the region of the coverage hole to guarantee good propagation conditions between the fixed relay and its associated gNB. This square region is tessellated in pixels of 1x1meter. For each pixel of this region, the path loss of the signal coming from the different gNBs is determined. Then, the pixel with the highest RSRP from the signals coming from the different gNBs is considered as the best position for placing the fixed relay. This process is repeated for all validated CHs. The output of this process is the geographical locations for each of the new fixed relays to be deployed. Once the fixed relays are deployed, this information is updated in the relay database.

Algorithm 3-2: Relay Control - Relay Positioning

1	Collect list of validated CHs
2	**For** each validated Coverage Hole in the list
3	Set the search area of possible locations where the relay can be placed
4	Compute path loss in each 1×1m pixel inside the area defined in step (3)
5	Obtain the coordinates of the pixel with highest RSRP from the signals coming from the different **gNBs**
6	**End for**
7	Gather optimal coordinates for each relay in an output file.

3.4.2.2 Initial evaluation

The considered fixed relay placement methodology has been initially executed to determine the best location of a new fixed relay to provide coverage to Coverage Hole CH_2, identified in section 3.4.1.2, which is the validated coverage hole with the highest repetitiveness (see Table 3-4). The relay placement methodology considers all the potential geographical locations inside the building where the coverage hole CH_2 was detected. After running the relay placement methodology, the best geographical location to place the relay R1 is represented in orange colour in Figure 3-28.

To evaluate the network performance improvement and the potential benefits in terms of energy consumption reduction with the placement of this fixed relay, an exhaustive search of different values of transmitted power of the different BSs has been done. For the different combinations of transmitted power values, an estimation of the coverage hole area has been done by determining the number of pixels of 1x1m in which the RSRP is below a threshold Th_{RSRP}=*-90dBm*. On the other hand, the overall total power consumption has been estimated as:

Figure 3-28: Relay control - best geographical location to place the fixed relay R1 (in orange) to address coverage hole CH_2

$$P_{tot} = P_{0,r} + \alpha_R \cdot P_{T,R} + \sum_{b=1}^{N_{BS}} P_{0\_b} + \alpha_b \cdot P_{T,b}$$

where $P_{0,r}$ represents the relay power consumption at zero RF output power associated to circuits, signal processing, etc., and a_R corresponds to the linear dependency between the total relay power consumption and the radiated power $P_{T,R}$. Similarly, the terms $P_{0,b}$, α_b and $P_{T,b}$ are the corresponding power model parameters for the *b*-th BS. In the considered scenario, the number of BSs is $N_{BS}=3$. The considered simulation parameters are the ones that were presented in Table 3-3. Additionally, Table 3-5 presents the considered parameters of the deployed relay and the parameters of the power consumption model.

Table 3-5: Relay Control - Model Parameters

Parameter	Value
BS transmitted power range	[35-50] dBm
Fixed Relay transmitted power	10 dBm
Relay antenna gain	3 dB
Relay bandwidth	20 MHz
Relay propagation model	InH - 3GPP TR 38901
Power consumption parameters [48]	$P_{0,r}=6.8W$, $a_R=4$, $P_{0,b}=130W$, $a_{BS}=4.7$

Table 3-6 presents a comparison in terms of power consumption and the area of the coverage hole for the different configurations of BS and relay transmitted power. The first solution in Table 3-6 represents the benchmark configuration with an initial value of transmitted power of 35dBm at each gNB. In this case, a relatively large coverage hole is observed as it was shown in Figure 3-25. A possible solution to address this coverage hole is to increase the transmitted power of one of these gNBs. As shown in Table 3-6, increasing the transmitted power of one of the gNBs to 49dBm leads to a power consumption P_{tot}=793W, i.e. an increase in 82% in the total power consumption with respect to the benchmark configuration (where all three BS transmitted with 35dBm). Even in these cases, the coverage hole is not completely solved, as shown in Table 3-6. In order to address the coverage hole, an exhaustive search of the transmitted power of the $N_{BS}=3$ gNBs has been done with the aim of minimising the total power consumption P_{tot}.

Table 3-6: Relay Control - Comparison of Different Combinations of Transmit Power at the BS and Relay

	$P_{T,1}$(dBm)	$P_{T,2}$ (dBm)	$P_{T,3}$(dBm)	$P_{T,R}$(dBm)	P_{tot}(W)	Coverage Hole Area (m²)
Benchmark solution	35	35	35	--	434.58	224
Sub-optimal solutions	49	35	35	--	793.04	24
	35	49	35	--	793.04	7
	35	35	49	--	793.04	11
Best solution without relay	38	38	48	--	745.86	0
Solution with a relay	35	35	35	10	441.42	0

* $a_{BS}=4.7$, $P_{0,b}=130W$, $a_R=4$, and $P_{0,r}=6.8W$

The optimum solution found without deploying a relay is shown in Table 3-6 but it requires a total power consumption of 745.86W that corresponds to an increase of 71% with respect to the benchmark configuration. However, the deployment of the relay at the identified location (see Figure 3-28) with a transmitted power of $P_{T,r}=10dBm$, addresses the problems at the coverage hole with a total power consumption P_{tot}=441.42W (see Table 3-6). This corresponds to a reduction in 40.8% of the total power

consumption with respect to the best solution found without relay. This reduction has a high dependence of the considered parameters in the energy consumption model. To illustrate this, Table 3-7 presents the power consumption reduction that can be obtained by deploying this fixed relay with respect to best solution without relays for different combinations of the terms $P_{0,r}$, a_R, $P_{0,b}$ and a_{BS} [48][49]. As shown in Table 3-7, this power consumption reduction ranges between 37% and 71%.

Table 3-7: Relay Control - Power Saving by Deploying Relay for Combinations of the Energy Parameters

Parameters for the BS and relay energy consumption model	**a_{BS}**	28.4	28.4	4.7	4.7	2.8	2.8	2.57	2.57
	$P_{0,b}$	156.38	156.38	130	130	84	84	12.85	12.85
	a_R	20.4	4	20.4	4	20.4	4	20.4	4
	$P_{0,r}$	13.91	6.8	13.91	6.8	13.91	6.8	13.91	6.8
Power savings by deploying the relay		71.2%	71.5%	39.8%	40.8%	36.9%	38.5%	66.9%	70.1%

As shown in Table 3-6, it is worth noting that the deployment of the fixed relay leads to a slight increase (around 1.5%) in the power consumption with respect to the benchmark solution. However, the deployment of the fixed relay may be useful to reduce the gNBs transmitted power (and consequently the total power consumption) and guarantee the coverage requirements in the whole cell. As a future line of work, an extension of these initial results will be done in BeGREEN D4.3, taking into account all the validated coverage holes identified in the scenario. Then, an assessment of the gNBs transmitted power reduction that can be obtained with the deployment of fixed relays (and the corresponding total power consumption reduction) will be evaluated guaranteeing, at the same time, the overall coverage requirements in the scenario.

3.4.3 Candidate RUE identification.

As shown in section 3.4.2, coverage holes may be addressed by deploying fixed relays at specific geographical locations. However, it may lead to an increase in CAPEX and an increase in the total power consumption. Fixed relays consume certain amount of power $P_{0,r}$ even when no user is connected. In order to address these drawbacks, this section explores the possibility of taking advantage of UE relaying capabilities. According to this, some specific UEs can become RUE and act as a relay between the gNBs and neighbour UEs that may be located at the coverage holes. This approach requires the identification of UEs with some specific characteristics that make them good candidates to become RUEs. In general, static UEs with good propagation conditions with the BSs that remain active for long periods of time may be good candidates to become RUEs.

Human mobility has usually a strong regularity and predictability since it is usually driven by daily/weekly schedules. With the development of recent technologies for the collection of historical user location and other context information, and the capacity of processing this information by means of AI/ML algorithms, an accurate characterization of future UE locations and UE mobility can be obtained. These technologies can be useful for the identification of periodicity/seasonality in the regions visited by the UEs during the day/s. Identifying metrics related to UE presence in different geographical regions, regularity of this presence, session duration statistics in each region, etc., is essential for the adequate identification of candidate RUEs.

Figure 3-29 illustrates the workflow of the candidate RUE identification process. The main objective of this process is to identify UEs with some specific characteristics that may indicate that they can be good candidate UEs to become RUEs and act as a relay to serve neighbour users located in a coverage hole. This process is activated by the RFM rApp in the Non-RT RIC after the identification of coverage holes in specific gNBs. The candidate RUE identification process is run in the AI Engine. This process collects the characterization of the coverage holes from the coverage hole database and the available measurements of the historical sessions of UEs in this gNB available in the measurements datalake (see steps 3-4). This information is processed in order to identify good candidate UEs to serve as RUE. The details of this process is explained in section 3.4.3.1.

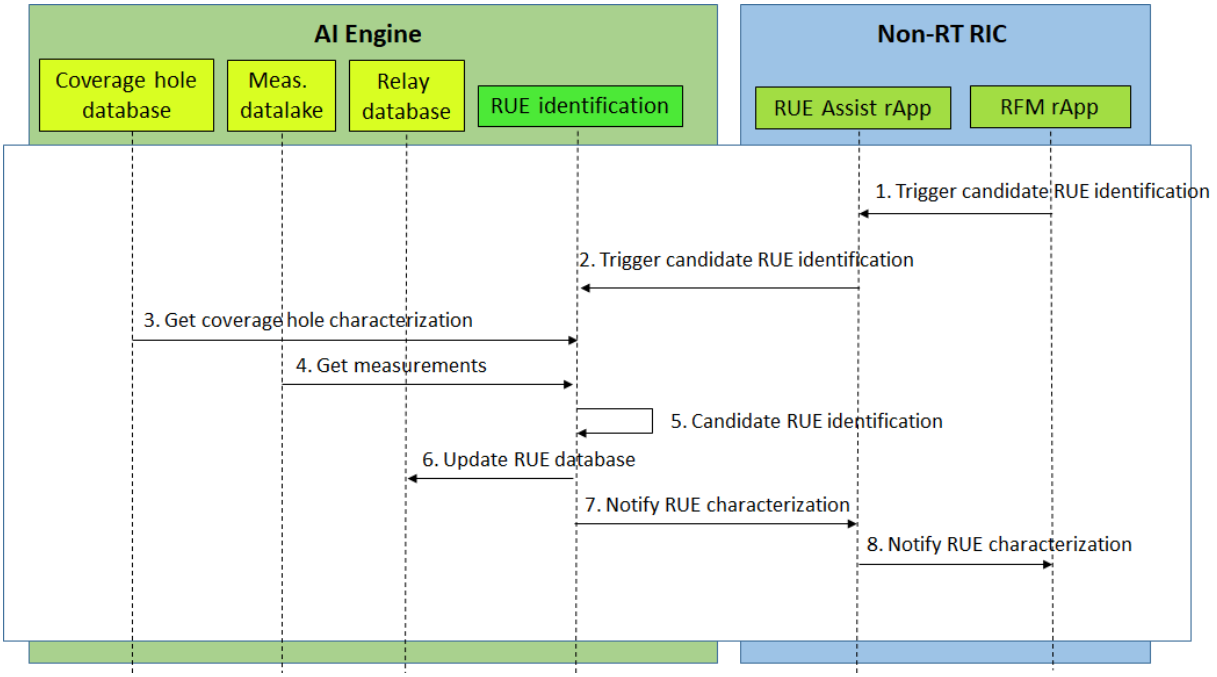


Figure 3-29: Relay Control - candidate RUE identification process

The list and characterization of the candidate UEs to become RUEs for each identified coverage hole is send to the relay database in the AI Engine. Then, the end of the process is notified to the RFM rApp in the Non-RT RIC (see step 8).

3.4.3.1 Candidate RUE identification algorithm

The candidate RUE identification process described in step 5 in Figure 3-29 is presented in Algorithm 3-3. First, the information of the identified coverage holes is extracted from the coverage hole database. For each coverage hole, the proposed algorithm determines the list of UEs that have been located in a square region of side *x* centred at the coverage hole centroid, where *x* is a distance in meters, during at least one time period of *T* seconds in the available dataset. Note that the value of *x* must be higher than the coverage hole diameter and the candidate RUEs may be located outside the coverage hole region to guarantee good coverage for them. For this purpose, information about the sessions carried out by the different users at this geographical square region is collected from the measurements datalake. In particular, the time when each session is established and released is collected for each UE. According to the obtained information of these UEs, the proposed algorithm builds a matrix ***R*** that represents the presence of each UE in the specific geographical square region in each *m*-th time period of duration *T* for each *d*-th day (with $m=1,\ldots,M$ and $d=1,\ldots,D$). This analysis is done in a UE-by-UE basis. Each element of the matrix ***R***, $r_{m,d}=1$ if the UE was connected at the *m*-th time period of the *d*-th day, and $r_{m,d}=0$ otherwise.

Algorithm 3-3: Relay Control - Candidate RUE Identification

1	Get the list of validated Coverage Holes
2	**For** each validated Coverage Hole
3	Determine a square region with side x centered at the coverage hole center.
4	Determine the list of UEs that have been located inside the region defined in step (3) during at least one time period of duration T.
5	**For** each UE in the list of UEs Build matrix R
6	Determine presence, average RSRP and session duration, daily and weekly regularity.
7	**End**
8	Select candidate RUEs

9	**End**

This ***R*** matrix is processed for each UE to determine the following statistics:

- *Presence (%):* Percentage of time periods with duration *T* in which a specific UE is present in the specified square area. A UE with a high presence is a good candidate to become a RUE.

$$Presence(\%) = \frac{\sum_{d=1}^{D}\sum_{m=1}^{M} r_{m,d}}{M \cdot D}$$

- *Presence (%) w.r.t. the time periods when the coverage hole is detected:* Percentage of time periods with duration T in which a specific UE is present in the specified area with respect to the number of time periods where the coverage hole is detected. The presence of the coverage hole in the different time periods is obtained from the matrix ***P*** available in the coverage hole database (see section 3.4.1.1). This term is quite relevant because it indicates the availability of the corresponding UE to become a RUE in the time periods in which the coverage hole is detected.
- *Average RSRP* reported by the UE when it is in the specific region. A UE with favourable propagation conditions with respect to its serving gNB is a good candidate to become RUE.
- *Average session duration:* It is calculated as the time the UE is active in this specific area divided by the number of sessions established by the user in this area. A UE with a high average session duration is a better candidate to become a RUE.
- *Daily regularity (d_r):* This index measures the regularity of the presence of a UE in the specific area for the different time periods of the day. In order to determine this index, the ***R*** matrix is used. For the sake of clarity, this process is represented in Algorithm 3-4. For a given *m*-th time period of the day (with *m=1,…,M*), the percentage of time in which the UE is present at each specific *m*-th time period is determined for all the weekdays (i.e. from Monday to Friday) of the dataset. Then, the number of time periods of the day in which this percentage is higher than a specific threshold ($N_{periods\_above\_th}$) is divided by the total number of periods of the day M (see step 4 in Algorithm 3-4). A UE with a high daily regularity means that it is located in the same geographical area at the same periods of the day in the different days, exhibiting a high regularity for the different weekdays. Users with a high daily regularity may be good candidates to become a RUE.

Algorithm 3-4: Relay Control - Calculation of the Daily Regularity

1	D' is the number of weekdays (from Monday to Friday) in the dataset.
2	For each m=1 with m<M
3	If $\frac{\sum_{d'=1}^{D'} r_{m,d'}}{D'} > Th$
4	$N_{periods\_above\_th}=N_{periods\_above\_th}+1$
5	$d_r=N_{periods\_above\_th}/M$

- *Weekly regularity:* This index measures the regularity of the presence of a UE in the specific area for the different time periods of the week. In order to determine this index, the week is divided in *N=M·D* number of time periods, each one with duration T. For a given n-th time period of the week (with n=1,…,N), the percentage of times in which the UE is present for all the weeks of the dataset is calculated. Then, the number of time periods of the week in which this percentage is higher than a specific threshold is determined and divided by the total number of periods in the week. A UE with a high weekly regularity indicates that it is located in the same geographical area at the same time periods of the week for the different weeks, exhibiting a high regularity for the different weeks. These users may be good candidates to become a RUE.

After analysing these metrics for all the UEs in the list, a ranking of the most adequate UEs that may become RUE is obtained. This process is repeated for all detected CHs. The output of this process is a list and characterization of candidate RUEs for each coverage hole. This information is updated in the relay database.

3.4.3.2 Initial evaluation

The candidate RUE identification process has been run for all the coverage holes identified in section 3.4.1.2. For each coverage hole, the considered methodology searches for candidate RUEs located in the same building and floor where the coverage hole was identified. Table 3-8 presents the best candidate RUEs identified in each of the coverage holes.

Table 3-8: Relay Control - Best Candidate RUEs Identified in Each Coverage Hole

	Candidate RUE Identifier	Presence (%)	Weekly Regularity	Weekday Regularity	Avg. Session Duration (h)
CH_1 (1,60,43)	UE_943	**58.28**	**0.75**	**0.10**	**11.9**
	UE_456	53	0.64	0.47	10.6
CH_2 (0,81,41)	UE_765	**100**	**1**	**1**	**14**
	UE_474	44.5	0.49	0.26	6.18
CH_3 (1,69,76)	UE_992	**77.30**	**0.56**	**0.59**	**5.38**
	UE_093	63.1	0.69	0.56	4.04
CH_4 (0,318,75)	UE_184	**39.7**	**0.53**	**0.04**	**1.56**
	UE_382	30.68	0.47	0	1.46
CH_5 (0,74,74)	UE_920	**100**	**1**	**1**	**14**
	UE_429	60.0	0.83	1	8.4
CH_6 (0,178,42)	UE_544	**56.9**	**0.23**	**0**	**1.71**
	UE_992	44.2	0	0	2.5
CH_7 (0,85,105)	UE_302	**26.4**	**0.65**	**0**	**1.66**
	UE_032	25.2	0.57	0	1.15

In coverage hole CH_1, user *UE_943* can be a RUE during almost 60% of the time when the coverage hole is detected. As shown, this user has a relatively high weekly regularity and a long average session duration. However, it should be necessary to check that there is any other UEs that can serve as RUE in the rest of the time in which this coverage hole is detected. In case that no such UEs are found, the deployment of a fixed relay may be necessary. In turn, in coverage hole CH_2, user *UE_765* is available to serve as RUE in the time periods when the coverage hole is present. In this case, it may not be necessary to deploy a fixed relay for addressing coverage hole CH_2. Finally, there are other coverage holes, such as CH_7, in which the best-found candidate RUE would only be available to serve UEs in the coverage hole during a 25% of the time when the coverage hole is detected. In this case, many different UEs would be needed to serve as RUE alternatively in order to cover the 100% of the time. In these cases, the deployment of a fixed relay may be a better solution.

An initial evaluation of the computation time and the associated energy consumption of the Candidate RUE Identification process has been done. The result of this process for the collected measurements during one day can be obtained in approximately 6 seconds in the considered scenario with 7 coverage holes in the University Campus region covered by 3 gNBs. The process has been run with an Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz processor. The average power consumption of this processor is around 25W, leading to an energy consumption of 0.04Wh (144J) for each day. This corresponds to a very low energy consumption.

3.4.4 Relay activation/deactivation process

The proposed methodology aims to smartly activate the relays to improve the performance of UEs located in coverage holes and deactivate the relays when they are not necessary in order to reduce energy consumption. For this purpose, this methodology makes use of recent collected measurements and status of the different relays (available in the relay database). This information and an activation/deactivation trained model is used to take adequate decisions of relay activation/deactivation. Two kinds of relays are

considered: i) fixed relays deployed in the network and ii) RUEs, (i.e. UEs with relaying capabilities).

The relay database must contain updated information of all the fixed relays and candidate RUEs connected to each BS. On the one hand, concerning fixed relays, this database contains information about the relay geographical location, the relay status (i.e. whether the relay is active/inactive) and an availability probability. On the other hand, for the particular case of RUEs, similar parameters are considered:

- RUE location: when a UE establishes a connection with a gNB, it is necessary to check whether this UE has relaying capabilities and consent and whether it belongs to the list of candidate RUEs for this gNB. Additionally, it is necessary to check that the UE has good propagation conditions with the serving gNB. In this case, the RUE location is collected and updated in the relay database. It is worth noting that, although RUEs are usually static/semi-static UEs, their location may change.
- RUE status: It indicates whether the relaying functionality is active/inactive.
- RUE availability probability: The spectral efficiency in the link BS-RUE is measured, and in case that this value is higher than certain threshold, then, the RUE is considered to be available. Other metrics such as battery level, etc. may also be checked to determine the RUE availability probability.

For all the relays connected to each gNB, the relay activation/deactivation process is run continuously, and the relay status is updated with a certain periodicity $T_{relay\_status\_update}$. The relay activation/deactivation decisions are done based on a trained DQN that combines RL with DNNs. As explained in BeGREEN D4.1 [1], with a periodicity $T_{relay\_status\_update}$, an agent takes the decision of activation/deactivation of a relay (i.e. action) to be applied for the next time period. This action is based on the state observed in the previous time period and a policy π that has been learnt in the DQN training process. The state observed in the previous time period is based according to the following information:

- The current status of the relay (i.e. whether each relay was active or not in the previous time period).
- The average number of UEs that have been served by the relay, in case that the relay was active in the previous time period.
- An estimation of the number of potential users that would have been served by the relay, in case the relay was inactive in the previous time period. For the case of fixed relays, this estimation can be done by observing the number of users that have been served in previous periods of time in which the relay was active (e.g. at the same period of the day in previous days). In turn, this kind of estimation may not be valid for the case of RUEs since they may change its geographical location as a function of time. However, the estimation of potential users that would have been served by the RUE in the previous time period can be obtained by means of proximity analytics information collected by means of the NWDAF.

The procedure for obtaining NWDAF proximity analytics can be done as described in the following [50]:

1. A request is sent to the NWDAF for analytics related to relative proximity from a specific RUE (i.e. statistics of number of UEs that satisfy a proximity criterion with respect to the RUE in the previous time period with duration $T_{relay\_status\_update}$).
2. The NWDAF may follow the UE Input Data Collection Procedure via the Data Collection Application Function (DCAF). The DCAF may collect proximity related input data directly from the UE Application, for NWDAF to determine a list of UEs fulfilling certain proximity criterion [51].
3. The NWDAF derives requested analytics.

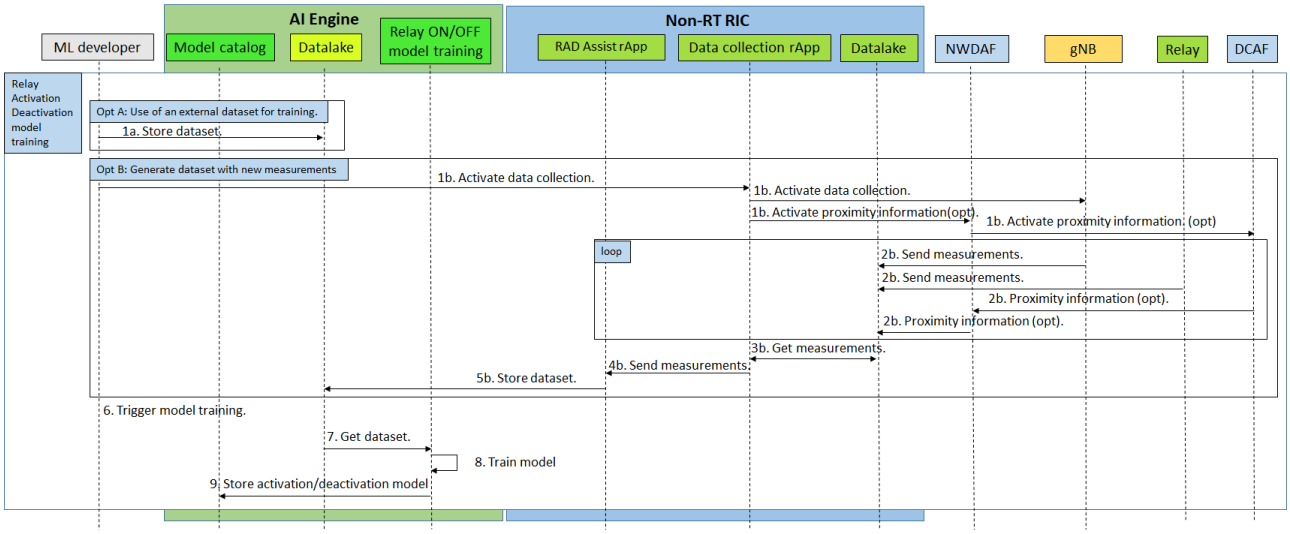


Figure 3-30: Relay Control - workflow of the relay activation/deactivation training process.

It is possible to set a continuous reporting of relative proximity information with a periodicity $T_{relay\_status\_update}$. By combining the status of the relay and the previously mentioned metrics, the proposed methodology makes use of a policy based on a trained DQN model, see BeGREEN D4.1 [1], to take adequate relay activation/deactivation decisions. Section 3.4.4.1 presents the process and workflow for the model relay activation/deactivation model training while section 3.4.4.2 focus on the process of inference to take the relay on/off decisions.

3.4.4.1 Relay activation/deactivation model training

Figure 3-30 illustrates the process of training of the relay activation/deactivation model. The training process can be triggered manually by the ML developer. Alternatively, under certain specific conditions, or with a given periodicity, a retraining process may be triggered by the Relay Activation/Deactivation (RAD) assist rApp that continuously monitors the performance of the ML model. In case that an available dataset is used for training, then, this dataset is directly stored in the datalake of the AI Engine (see step 1a in Figure 3-30). In turn, in case that it is necessary to generate a new dataset for training the model (option b in Figure 3-30), then, the Data collection rApp in the Non-RT RIC triggers the collection of new measurements. During a specific training period, collected measurements (such as the status of the relay, the number of users served by the relay or the number of potential users) are collected and stored in the Non-RT RIC datalake (see step 2).

For the case of RUEs, the number of potential users that would have been served if the relay was active in the previous time period is obtained by means of proximity analytics information by means of the NWDAF. Then, the number of UEs fulfilling a specific proximity condition to the RUE is requested to the NWDAF (via DCAF) and sent to the datalake in the Non-RT RIC (see step 2b). Once the process of generation of the dataset is finished, the dataset is stored in the datalake of the AI Engine (step 3-5). Then, the training process may be triggered manually by the ML developer or by the RAD Assist rApp. Training data stored in the datalake is collected and the model training process is run in the AI Engine. Finally, the trained AI/ML relay activation/deactivation model is stored in the AI/ML model catalogue in the AI Engine. The details of the training process were described in BeGREEN D4.1 [1], and are briefly explained as follows.

The state s(t) is represented as a vector associated with a particular BS *b*, and it has different components listed in the following:

- $C_b(t)=\{a_{b,1}(t), a_{b,2}(t),\ldots,a_{b,R}(t)\}$ denotes the configuration (ON/OFF) of all relays in the previous time period *t*.

- $N_b(t)=\{N_{b,1}(t), N_{b,2}(t),..., N_{b,R}(t)\}$ corresponds to the average number of UEs that have been served by each relay in the previous time period *t*.
- $N'_b(t)=\{N'_{b,1}(t), N'_{b,2}(t),..., N'_{b,R}(t)\}$ is the average number of UEs that would have been served by the *r*-th relay in the previous time period if the relay had been active. The total number of components in the state is $3 \cdot R$ where R is the number of considered relays.

A given action *a(t)* can be seen as a vector $C_b(t)=\{a_{b,r}(t)\}$ that contains the relay activation configuration applied every time window accounting for all the considered relays. The so-called action space contains all relay activation configurations. Since a relay has only 2 possible modes (activated and deactivated), the total number of possible actions in the action space is 2^R.

In order to learn the policy that leads to the best action given a specific state, a reward function is used in the training process. As described in D4.1, the reward function *r(t+1)* that assesses the action *a(t)* that is selected in a specific state *s(t)* can be expressed as:

$$r(t+1)=1-\frac{1}{R}\sum_{r=1}^{R} c_{b,r}(t)$$

with

$$c_{b,r}(t)=\begin{cases}\alpha & if \quad a_{b,r}(t)=1 \text{ and } N_{b,r}(t)<Th_{num}\\ \beta & if \quad a_{b,r}(t)=0 \text{ and } N'_{b,r}(t)\geq Th_{num}\\ 0 & \text{otherwise}\end{cases}$$

Where $N_{b,r}(t)$ is the average number of UEs served by the *r*-th relay in period (t, $t+T_{relay\_status\_update}$) while $N'_{b,r}(t)$ is the average number of potential UEs that would have been served by the *r*-th relay in period (t, $t+T_{relay\_status\_update}$) if the relay had been active. According to the previous equation, $c_{b,r}(t)=\alpha$ if the *r*-th relay was active in the previous time period (i.e. $a_{b,r}(t)=1$) and the number of served users was below a threshold (i.e. $N_{b,r}(t)<Th_{num}$). In turn, $c_{b,r}(t)=\beta$ if the *r*-th relay was deactivated (i.e. $a_{b,r}(t)=0$) but the number of users that would have been by this relay in the previous time period was higher than a threshold (i.e. $N'_{b,r}(t)\geq Th_{num}$). Initially, $\alpha=1$ and $\beta=1$ can be considered but also other possible values may be studied.

The training process described in step 8 in Figure 3-30 is detailed in Algorithm 3-5. At each training step, the agent observes the state and chooses an action a(t) following an ε-greedy policy that selects the action based on the current policy with probability 1-ε and a random action with probability ε. This random action selection is needed in the training process for incorporating the capability to explore new actions that are different from the ones that the current policy would select. After applying the selected action, the obtained reward is measured and saved in an experience tuple dataset. Every time that the experience dataset reaches its storage capacity, older experiences are removed and substituted by recent ones.

Algorithm 3-5: Relay Control - DQN Training

```
1   Initialise DNN counter p=0
2   For t=0 with t<Num_train_steps
3       Collect state s(t)
4       Generate random number ε' between 0 and 1
5       If t<InitialCollectSteps
6           Select a random action a(t)
7       else If ε'< ε
8           Select a random action a(t)
9       Else
10          Select an action a(t) based on policy π
11      End if
```

```
Compute r(t+1) and s(t+1) according to a(t)
If D is full
    Delete the oldest experience
Store experience <s(t), a(t), r(t+1), s(t+1)> in D
Sample randomly a minibatch of experiences U(D)
Compute loss function L(θ)
Compute the mini-batch gradient descent ∇L(θ)
Update weights θ of evaluation DNN
If p<P
    Update the weights of target DNN θ⁻ = θ and set p=0
Else
    p=p+1
End if
End for
```

Moreover, at the beginning of the training, the agent selects actions randomly (i.e., ε is set to 1) to gather a wide variety of experiences. This is maintained during a number of *InitialCollectSteps* training steps. The update of the weights of the evaluation DNN is done at every training step by considering the experiences accumulated in the experience dataset. An updating process consists of making a random selection of a mini-batch U(D) of past experiences J belonging to the dataset. Then, the update is performed by means of a mini-batch gradient descent procedure. To this end, the average Mean Squared Error (MSE) for all the experiences in U(D) is computed. Then, the mini-batch gradient descent is computed by the derivative of L(θ) with respect to θ. The final step consists of updating the weights of the evaluation DNN. Following each update of θ, the obtained Q(s,a,θ) will be used to select new actions. In relation to the weights θ− of the target DNN, they are updated as θ^{-} =θ after every P update of the evaluation DNN. More details of this process were presented in BeGREEN D4.1 [1].

3.4.4.2 Relay activation/deactivation model inference

Figure 3-31 illustrates the process of relay activation/deactivation inference based on the trained model. As shown in Figure 3-31, the process is repeated iteratively with a periodicity $T_{update\_relay\_status}$. The first step consists on the collection of measurements related to the number of users served by each relay $N_{b,r}(t)$, number of potential users $N'_{b,r}(t)$ and the status of the relay (i.e. whether the relay was active/inactive in the previous time period). For the case of RUEs, the number of potential users served by a RUE is obtained from the NWDAF via DCAF. Collected measurements are sent to the AI Engine via the RAD assist rApp (see step 2 in Figure 3-31). The status of the relay that is stored in the Relay database is also used by the AI/ML model inference (step 3). Then, the trained model is collected from the AI/ML model catalogue. The collected information and the trained model are used to generate a relay activation/deactivation recommendation for the next period $T_{update\_relay\_status}$ (see step 5). This recommendation is sent to the relay control at the SMO that sends a command to the nodes with the activation/deactivation action of the relay (steps 6-7). Finally, the relay status is updated in the relay database (Step 8). The RAD assist rApp continuously evaluates the performance of the used activation/deactivation model to determine the necessity of model retraining.

3.4.4.3 Initial evaluation

The proposed relay activation/deactivation methodology has been initially evaluated for taking adequate decisions of the activation/deactivation of a fixed relay deployed to address the coverage hole CH_2 (see Figure 3-25). The DQN model has been trained using measurements collected during 6 days. The step duration is 30 seconds. The DQN configuration parameters are summarised in Table 3-9.

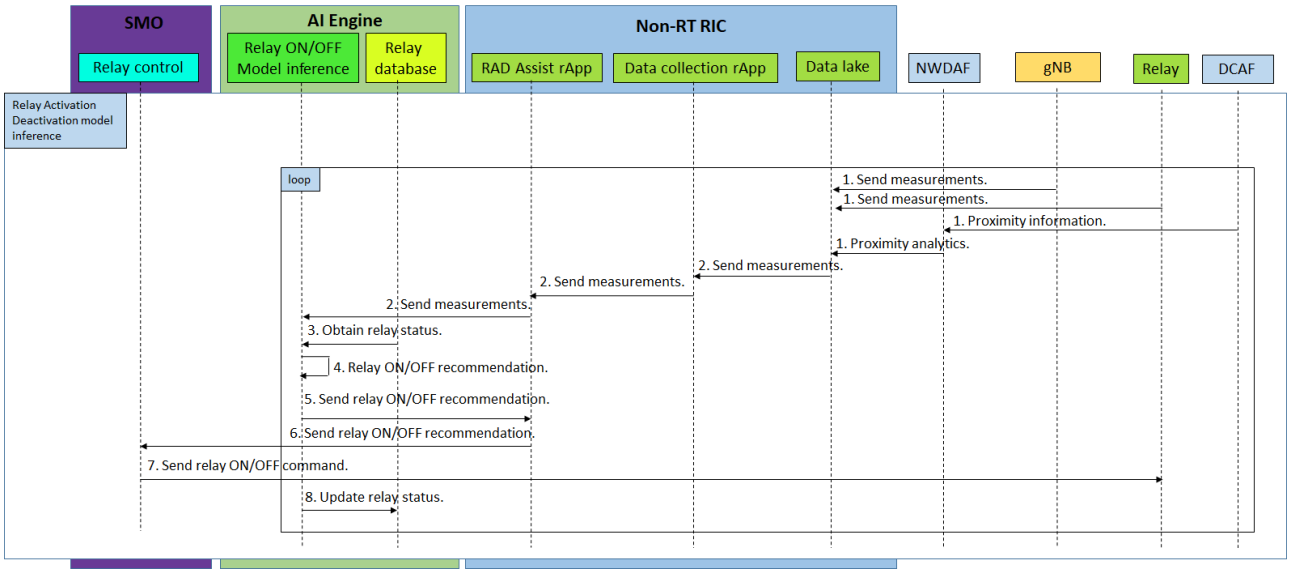


Figure 3-31: Relay Control - workflow of the relay activation/deactivation inference

Table 3-9: Relay Control - DQN Algorithm Configuration Parameters

Parameter	Value
InitialCollectSteps	500 steps
Num_train_steps	17280 steps
Experience replay buffer length	$100 \cdot 10^3$
Mini-batch size (J)	64
$T_{relay\_status\_update}$	30s
DNN updating period (P)	500 steps
Discount factor (γ)	0.9
Learning rate (α)	0.001
ε value (ε-greedy)	0.1
Th_{num}	0.5
DNN architecture	Input layer: 3 nodes Two Hidden layers: 100 and 50 nodes Output layers: 2 nodes

In the training process, the obtained policy is evaluated every 12 hours. The evaluation of each policy is executed with a simulation of the system during 2 hours. Then, the average reward obtained by this policy is computed. Figure 3-32 shows the evolution of the average reward of the obtained policies. As shown, the average reward increases as a function of the training time. Note that after 36 hours of measurements used for training, the average reward is around 0.95 (i.e. very close to the maximum reward of 1) and the learning process has converged since the average reward remains almost constant for the rest of the training time.

Figure 3-33 presents an example of the time evolution of the relay activation/deactivation decision according to the policy learnt in the training process. This example illustrates the number of served users/potential users in the period between 8h and 12h of a specific day. As shown, in the time periods when there are no users to be served by the relay, the policy decides to switch off the relay (Relay status=0 in Figure 3-33) with the objective of reducing energy consumption. In turn, in time periods when there are users to be served by the relay, the relay is switched on (Relay status=1).

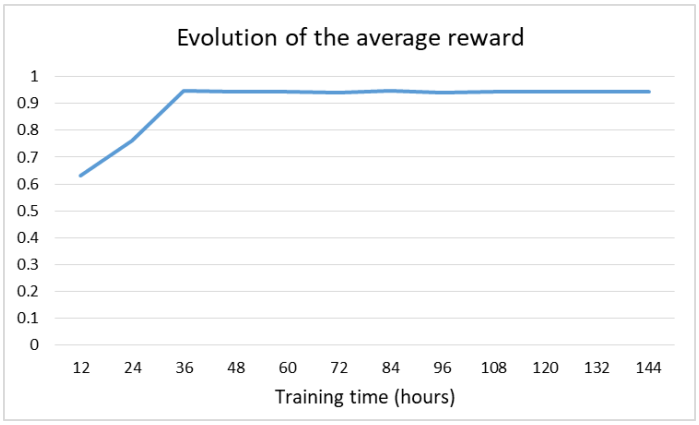


Figure 3-32: Relay Control - Evolution of the average reward of the obtained policies in the training process

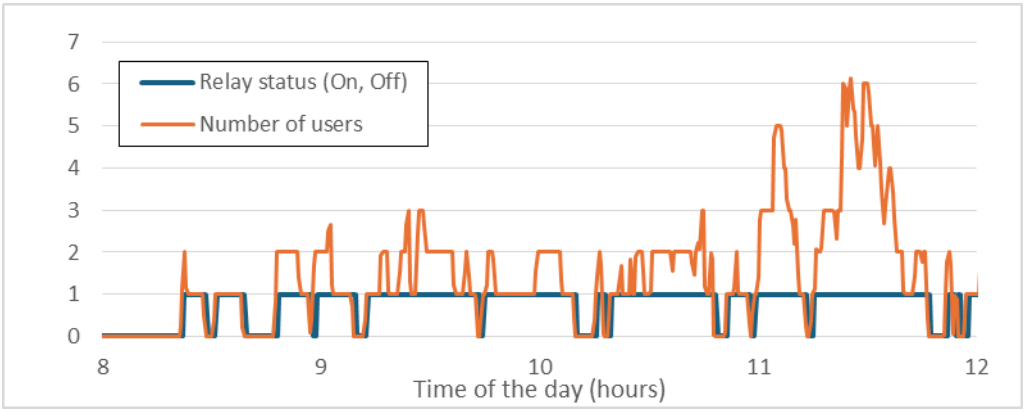


Figure 3-33: Relay Control - Evolution of the activation/deactivation actions with the obtained policy

After the analysis of the relay activation/deactivation decisions during the day in which the proposed methodology was tested, it was observed that the relay was active in 946 periods of 30 seconds (i.e. *t_active*=7.88 hours) and the relay was deactivated in 1934 periods of 30 seconds (*t_innactive*=16.12 hours). Then, the energy saved during a specific day, by turning off the relay in periods when there are no users to be served can be calculated as:

$$Energy\_saving = t\_innactive \cdot (P_{0,r} - P_{sleep})$$

According to [48], the power consumption when the relay is active at zero RF output power is $P_{0,r}$=6.8 W. Concerning the term P_{sleep}, the Advanced Sleep Modes defined in [52]consider the case of the sleep mode SM4 that turns off most of the components of the relay when it is deactivated. Then, the value of P_{sleep} is determined as 10% of the value of the term $P_{0,r}$. According to previous equation, the energy that can be saved during this specific day by deactivating the relay is around 98.6 Wh (354.9 kJ). It is worth noting that the relay can be set to sleep mode SM4 since the relay activation time is quite fast (i.e. in the order of hundreds of milliseconds [52]) with respect to the considered period of activation/deactivation decisions of 30 seconds. The energy savings that can be obtained may differ depending on the presence of users near the relay location and also on the considered day. As an example, in a geographical region without users to be served by a specific fixed relay (e.g. in a weekend day), the relay can be deactivated during the whole day leading to a maximum energy saving equal to 147 Wh (529.2 kJ). In BeGREEN D4.3 a more detailed evaluation of the total energy savings that can be obtained considering all the relays/RUEs deployed in the scenario.

Concerning the computation time and the associated energy consumption of the Relay Activation/Deactivation process, an initial evaluation has been done for the training of a single relay. As shown in Figure 3-32, the training process provides a good relay activation/deactivation policy after analysing 36 hours of measurements. In the considered scenario, this is obtained by executing the training process in

approximately 40 minutes with an Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz processor. The average power consumption of this training processor is around 25 W, leading to an energy consumption of 16.6 Wh (59.76 kJ).

Concerning the computation time for the model inference to take relay activation/deactivation decisions, the execution time of this process is approximately 0.44 ms. Considering that the model inference is executed with a periodicity of 30 seconds and that the power consumption of the processor is 25 W, the energy consumption is around 0.0088 Wh (i.e. 31.68 J) for each day and relay.

3.5 Traffic-aware compute resource management to enhance UPF energy efficiency

This section presents strategies to manage the compute resources of the edge servers hosting UPF instances of the 5G according to the traffic demand and aiming at reducing the energy consumption. In BeGREEN D4.1 [1] we introduced the main concepts and strategies to be considered when addressing this problematic: dynamic scaling the CPU frequency through the CPU P-states [53] and the number of threads. However, during the experimental characterization using the UPF implementation of the Open5Gs[13] open source 5GC, we found some limitations regarding its packet processing capabilities, i.e., single-threaded operation and low performance due to kernel-based packet processing [54]. Therefore, we decided to switch to an alternative open-source UPF implementation from OpenAirInterface (OAI)[14], which is based in VPP[15] and DPDK[16]. These fast packet processing technologies foster the increase of throughput, making them more suitable for real operational scenarios. However, they intensively utilize compute resources, what may lead to higher energy consumption [55] and open the door to energy efficiency-focused strategies.

In the following subsections we will detail the UPF implementation, the studied strategies and the initial evaluation based on an experimental characterization. Note that, as was introduced in D4.1 [1], the final objective is to enable proactive resource allocation based on traffic forecasting. To this end, we will use the real data from the Packet Data Network Gateway (P-GW) of Spanish MNO, as was presented in Section 3.3. We have started evaluating some ML models based on Facebook's Prophet[17], and the results will be reported in BeGREEN D4.3.

3.5.1 Solution design and use case

The OAI UPF-VPP is an open-source implementation of the 5GC UPF (Release 15 & 16) based on VPP and DPDK technologies. The main components of its implementation are i) the UPF-VPP application logic, (ii) the DPDK framework management for the NICs, and iii) the multi-thread management policy. Figure 3-34 illustrates the main architecture of UPF-VPP and how these components are involved during packet processing flow. The UPF-VPP runs on top of kernel bypass technologies, described as low-level building blocks (i.e. netmap, DPDK, or Open Data Plane). In the case of DPDK, once the packet batches arrive in the user space, VPP processes them in form of vectors (depicted as (2) and (6) in Figure 3-34). Then, the vector processing nodes perform packet management functions like memory management or buffering. For instance, in the case of the UPF, the vector processing nodes will perform the N3/N6 GTP decapsulation/encapsulation and the forwarding to the N6/N3 interfaces (noted as (3) and (7)}). The output node will finally forward packets to the required Network Interface Card (NIC) interface (steps (4) and (8)).

DPDK uses a Poll Mode Driver (PMD) that employs busy-polling to access NIC descriptors without

[13] https://open5gs.org/
[14] https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-upf-vpp
[15] https://fd.io/technology/
[16] https://www.dpdk.org/
[17] https://facebook.github.io/prophet/

interruptions. This method eases and expedites network packet management, i.e., the retrieval, processing, and delivery of packets to user space applications. Also, the DPDK interface driver and abstraction modules are used to manage the interface on user space. Therefore, other kernel network drivers are no longer needed (e.g., iptables or route tables). VPP, when bypassing kernel with DPDK libraries, handles packets in batch, allowing packet processing acceleration. However, VPP also inherits the intensive CPU usage of PMD, nearly leading to full utilization. In the case of the UPF-VPP, the NIC descriptors (1) and (5) in Figure 3-34 are constantly being pulled by DPDK to process the incoming GTP packets in the N3 or N6 interfaces. This leads to full CPU usage, irrespective of the network load conditions, thus obtaining high energy consumption and low energy efficiency in low loaded scenarios.

In terms of performance, the UPF-VPP implementation leveraging DPDK outperforms the results of other open-source solutions, such as Open5Gs [56]. In preliminary results using the experimental setup described in Section 3.5.2[18], we obtained around 35 Gbps of TCP throughput with a single-threaded UPF-VPP instance, compared to a maximum of 1 Gbps using Open5Gs [1]. However, this high performance comes at the cost of high energy consumption. Figure 3-35 depicts the performance of default governors in terms of energy consumption and achieved throughput. The performance governor, which selects always the maximum available frequency (2.7 GHz in this case), was able to reach 35 Gbps. However, even in idle mode, i.e. without processing GTP traffic, the energy consumption was high due to poll modem driver intensively using the CPU. On the other hand, the powersave mode always selected the minimum available frequency (1 GHz in this case), saving significant energy (~30%) in idle mode but reaching a maximum throughput of 22.6 Gbps. This highlights the need of a traffic-aware strategy to dynamically manage CPU frequency and optimise the energy efficiency of the UPF.

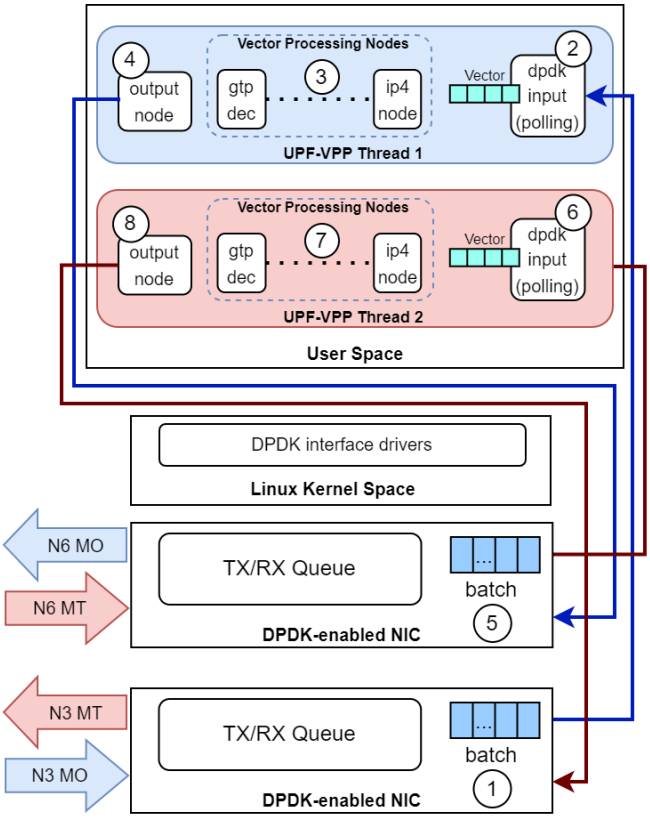


Figure 3-34: UPF resource allocation - UPF-VPP architecture and processing flow

[18] UPF hosted in a Supermicro E302-9D server: Intel Xeon processor D-2123IT 4 CPUs, Base freq. 2.2 GHz, Max Turbo freq. 3GHz, Max power consumption 60W

Figure 3-35: UPF resource allocation - CPU frequency governors' performance

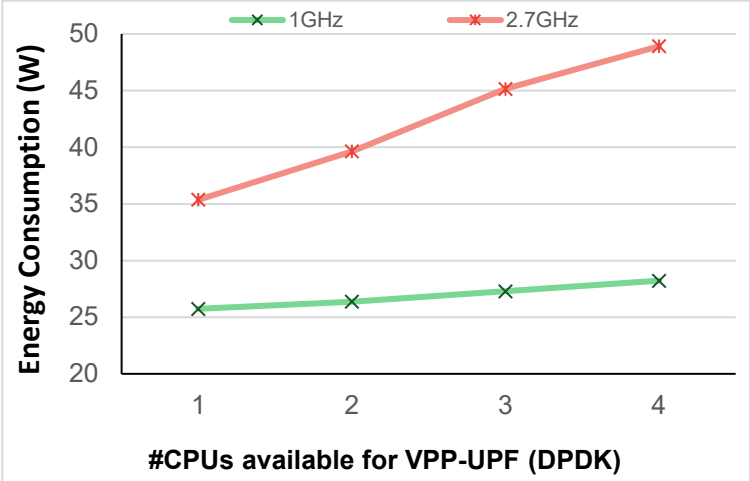


Figure 3-36: UPF resource allocation – Energy consumption vs number of threads (idle mode)

The evaluated UPF-VPP implementation supports single and multi-threading working modes. In single-thread mode, one main thread handles packet processing and other management functions. In the case of a multi-threading setup, the CPU cores made available during the initial configuration are managed by the VPP application in the user space and assigned to worker threads. These worker threads manage the NIC cards to perform end-to-end packet processing. For instance, in the case illustrated in Figure 3-34, two independent worker threads are separately managing the end-to-end packet processing of MO and MT packets. Since the workload in each thread can be managed by specific CPU cores, it enables the application of independent traffic-aware CPU policies such as frequency scaling. If more CPU cores than NICs are included in the initial configuration setup, more threads will be created, and each CPU core will be bound to a new worker thread.

Once the worker capacity, which is measured as "average vectors per node", goes close to the maximum capacity (i.e., 256 according to [57]), the RX queue of the worker starts dropping packets. Using more threads provides scaling capability for high-performance packet processing on the UPF-VPP, but at the same time, increases CPU power consumption.

Figure 3-36 depicts how the energy scales with the number of threads in idle mode for minimum and maximum CPU frequencies. According to these results, we can anticipate scenarios where using more threads at lower frequencies will offer better energy efficiency than using fewer threads at higher frequencies. Note that hyperthreading is not considered since VPP and DPDK can lead to performance

degradation when using several logical cores due to sharing resources of the physical core like the L1 and L2 caches [58].

According to the components and features presented in the previous section, we can conclude that the default UPF-VPP DPDK configuration and operation presents the following challenges to provide high performance plus energy-efficiency:

- Due to the intensive CPU usage required by the PMD, the energy consumption of each worker thread is very high, even when not handling any packets or any RX queue,. Besides, the workers are unable to dynamically control the PMD mechanism.
- The pool of available resources is set up only in the initial configuration (i.e., fixed number of workers and associated cores), and it does not apply or allow any dynamic resource reallocation according to the measured or expected traffic load.
- After initialization, the defined workers are associated with the available RX queues of the NICs and no built-in method is available to offload the packets being processed to a different core and releasing the one assigned by default.

To address these challenges, Figure 3-37 schematically depicts the resource allocation model that is being considered, where according to the number of NICs and their incoming load of the NICs, we can determine the following energy-efficient strategies:

- CPU frequency scaling: In the initial setup, a reception (RX) queue on a NIC is associated with a worker thread, which is then managed with a CPU. While it is not possible to modify PMD behaviour, which requires CPU resources in an exhaustive way, CPU p-states policies may be applied to adapt the frequency of the core to the incoming load in the NICs. Matching core frequency with actual packet processing requirements, will optimise energy-efficiency.
- Worker/Thread reallocation: Fixed allocations of threads can lead to inefficiencies. Processing capacity per NIC is increased by assigning more RX queues and workers to a NIC, what also allows to assign additional core resources. This enables strategies such as assigning to a NIC several cores at low frequency instead of a single core at high frequency, which may provide additional energy savings. On the hand, under low traffic demands, decreasing the number of threads by allocating several NICs to a common worker may also enhance energy efficiency.

Note that the combination of these two strategies will be highly relevant in real deployments, where uplink and downlink traffic demand is usually unbalanced. Therefore, proper thread allocation and CPU frequency scaling, tailored the predicted traffic demand for each NIC, will significantly enhance the overall energy efficiency of the system.

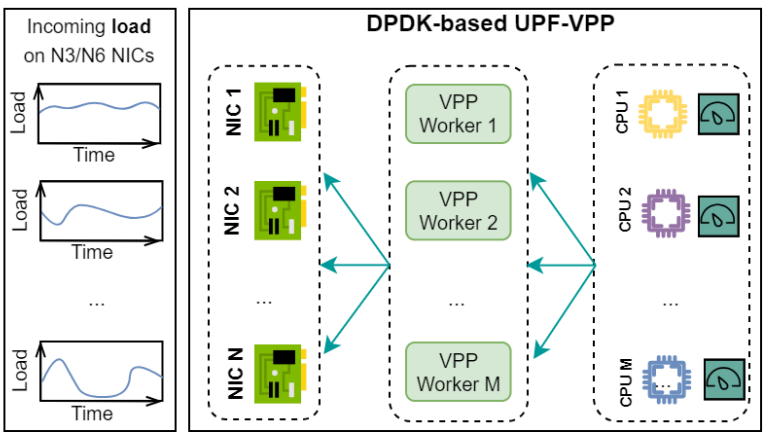


Figure 3-37: UPF resource allocation – Allocation model

In addition to these resource allocation strategies, in case one worker or core is not needed during a specific interval of time, two main approaches may be applied according to the use case. On the one hand, thread/CPU idle mechanisms, such as the enhanced C-states C0.1 and C0.2 available in recent Intel CPU generations [59], may be used to hibernate both core and worker functions such as the PMD. On the other hand, released cores may be reallocated to other services present in the same server, for instance AI/ML workloads.

In the next section, we will present an experimental characterization of these energy-efficiency strategies, with focus on the trade-off between performance and energy consumption. We will also validate the feasibility of dynamic approaches according to the current implementation of the DPDK-based UPF-VPP and the available tools in Linux kernels.

3.5.2 Initial evaluation

This section presents the experimental characterization performed to study the relationship between UPF performance, CPU resource allocation techniques and energy consumption. First, we describe the testbed and the software tools being used. Then, we discuss the obtained results.

3.5.2.1 Experimental testbed

The architecture of the experimental testbed is depicted in Figure 3-38 and described as follows:

- VPP-DPDK UPF server: The VPP-DPDK implementation of OAI's UPF[19] is deployed as baremetal in a server. The CPU of the server is an Intel Xeon processor D-2123IT with 4 physical CPUs (base frequency 2.2 GHz, max turbo frequency 3 GHz) and a maximum power consumption of 60 Watts. The server has four 10 Gbps interfaces, what allows us to reach throughputs of around 20 Gbps in uplink and 20 Gbps in downlink. Hyperthreading was enabled in the BIOS though not used by the UPF implementation as recommended in [58]. Turbo frequencies were also available.
- gNB & UE servers: These servers host the PacketRusher tool[20], which emulates the 5G gNB and the UE. Compared to other open-source emulation tools, PacketRusher is specially indicated for high performance scenarios, being able to reach 5 GB/s per UE. It requires an N2 connection to the 5GC Control Plane and a N3 connection to the UPF. We used Iperf3[21] in TCP mode to generate the traffic to and from the Application servers.
- Open5GS Control Plane server: Open-source solution which Implements the 5GC Control Plane[22]. It allows to connect network functions from different vendors or open-source implementations through standard interfaces, as is the case of the VPP-DPDK UPF via N4.
- Application servers: Host the Iperf3 servers and are connected to the UPF via the N6 interface.

The throughput experiments were conducted using Iperf3 TCP sessions in dual-mode, i.e. generating the same amount of traffic in both the uplink and downlink directions.

For the sake of comparison, all four interfaces were used during the tests, distributing the traffic equally among them. The reported results are based on the average of 3 to 5 experiments, though confidence intervals are not provided due to their negligible values. We used the tool powerstat[23] to obtain the consumption of the UPF server, which uses Intel's Running Average Power Limit (RAPL) interface.

[19] https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-upf-vpp
[20] https://github.com/HewlettPackard/PacketRusher
[21] https://iperf.fr/
[22] https://open5gs.org/
[23] https://github.com/ColinIanKing/powerstat

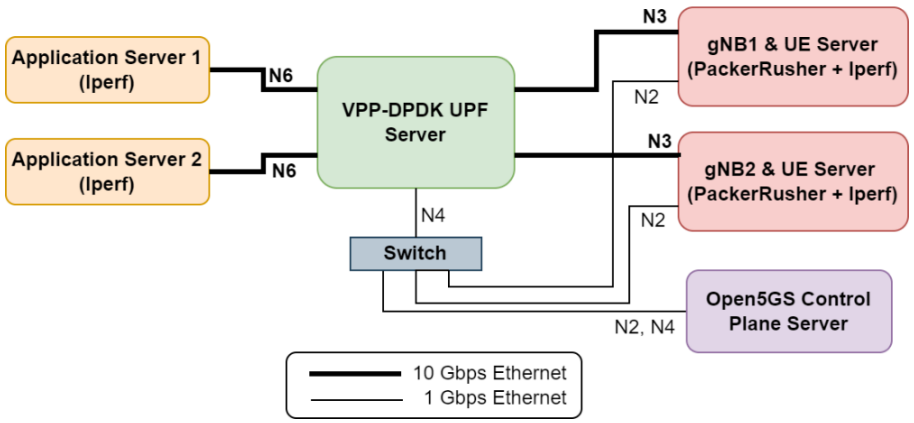


Figure 3-38: UPF resource allocation - experimental testbed

The option to obtain separated core and uncore power consumption was not available in our Operating System (OS). Thus, it is reported the total power consumption of the processor package of the server.

3.5.2.2 Experimental results

The performed experiments aimed at characterising the influence of two factors on the maximum achievable throughput and on the consumed energy of the VPP-DPDK UPF: (i) CPU frequency and (ii) the number of assigned threads. Note that in the first case, we only assigned a CPU to the UPF, while the others were unused and configured at minimum frequency.

CPU frequency scaling:

As was mentioned in Section 3.5.1, the PMD being used by DPDK in order to poll NICs requires a high utilisation of the CPU (around 100%). This increases the energy consumption even in the cases with no traffic when the UPF is idle. Figure 3-39 depicts the measured consumption in idle status for different CPU frequencies. Note that the measured power consumption of the server before starting the UPF was 16W, i.e. just initiating it increased significantly the energy consumption, ranging between 60% and 120% depending on the CPU frequency.

Additionally, the results highlight the impact of the turbo frequencies (i.e., frequencies higher than 2.1 GHz), which introduced a noticeable knee point between low and high CPU frequencies and significantly increased the energy consumption trend. As depicted in Figure 3-40, this causes that, with low traffic demands, low CPU frequencies can process incoming packets requiring of less power than higher frequencies. For instance, until it saturates at approximately 20 Gbps, working at 1 GHz significantly decreases the energy consumption compared to higher frequencies. Note that 2.7 GHz is equivalent to the performance governor.

Results in Figure 3-40 also show that the increase of energy consumption with the throughput is steeper at lower CPU frequencies compared to higher frequencies, where the increase is more gradual and smoother. This limits the benefits of using low CPU frequencies under high throughputs. This could be caused by uncore power consumption, for instance due to an increase of the utilisation and misses at the L3 cache due to the high throughout and low CPU frequency. Cache misses force DPDK to access slower DRAM memory more often, which is more power-hungry. Future work will include characterizing this possible problematic.

Figure 3-41 compares the power consumption of the Performance governor (i.e., at 2.7 GHz) with an adaptive strategy which selects the minimum CPU frequency able to serve the incoming traffic (from 1 GHz up to 1.7 GHz as show in the graphic). As previously mentioned, the Energy Savings (left y-axis) are substantial under low load conditions, exceeding 25%, but they gradually decrease as traffic increases. However, even in this case, improvements in energy savings ranging from 15% to 5% can still be achieved. Additionally, in real scenarios, to allow reaching higher throughputs during peak hours, the baseline UPF configuration will include several cores operating in performance mode. This scenario will be analysed in the next section.

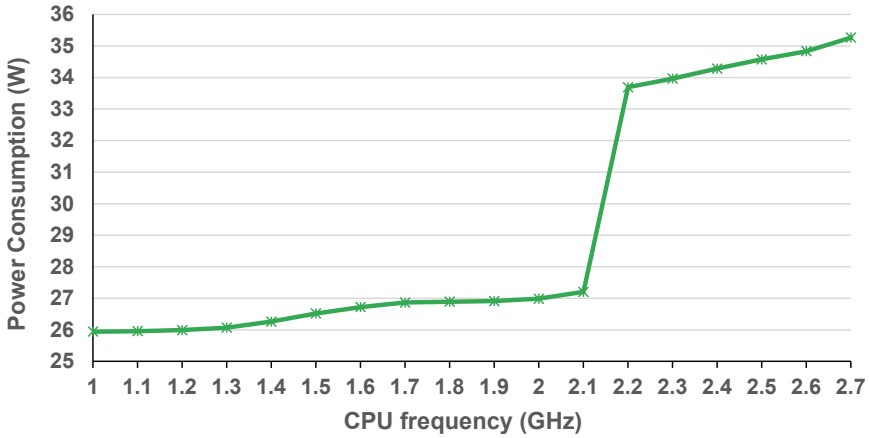


Figure 3-39: UPF resource allocation - Power consumption in idle status (no traffic)

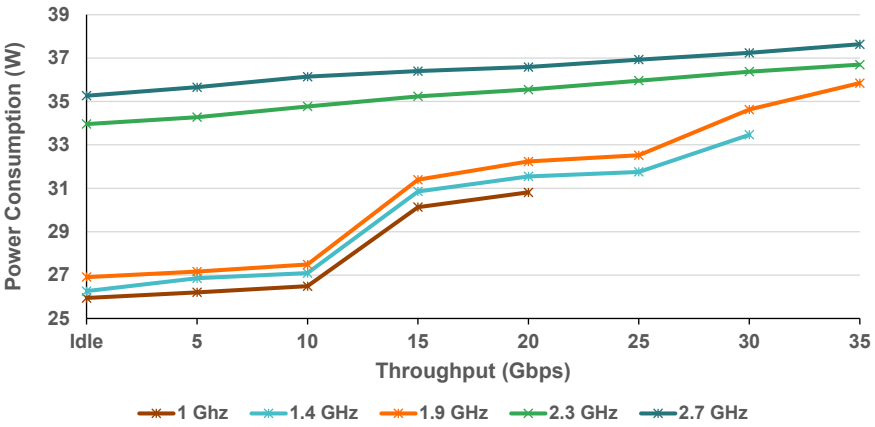


Figure 3-40: UPF resource allocation - Power consumption and achieved throughput according to CPU frequency

Workers/Threads scaling:

We next evaluated the performance of the VPP-DPDK UPF when increasing the number of workers and threads, i.e., when increasing the number of CPUs dedicated to the UPF. Particularly, we considered a scenario with 2 physical CPUs, i.e., one dedicated to N3 NICs and the other one to N6 NICs, and another one with 4 physical CPUs, each one dedicated to an individual NIC. We also varied the CPU frequencies, using the same one in all the CPUs being used.

Figure 3-42 depicts the results of 1, 2 and 4 CPUs, combined with CPU frequencies of 1, 1.9, 2.3 and 2.7 GHz (X-Y in the legend stands for X CPUs at Y frequency). As was expected according to the measurements based on idle status presented in Figure 3-36, the impact on energy consumption of increasing the number of CPUs is more noticeable at higher CPU frequencies.

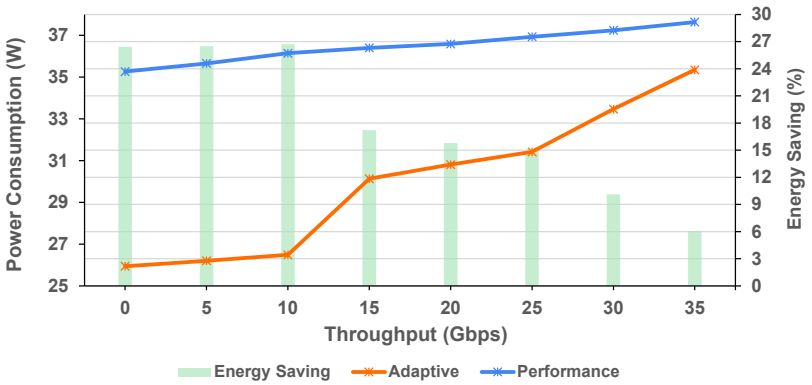


Figure 3-41: UPF resource allocation - Energy Consumption of the optimal vs performance CPU allocation

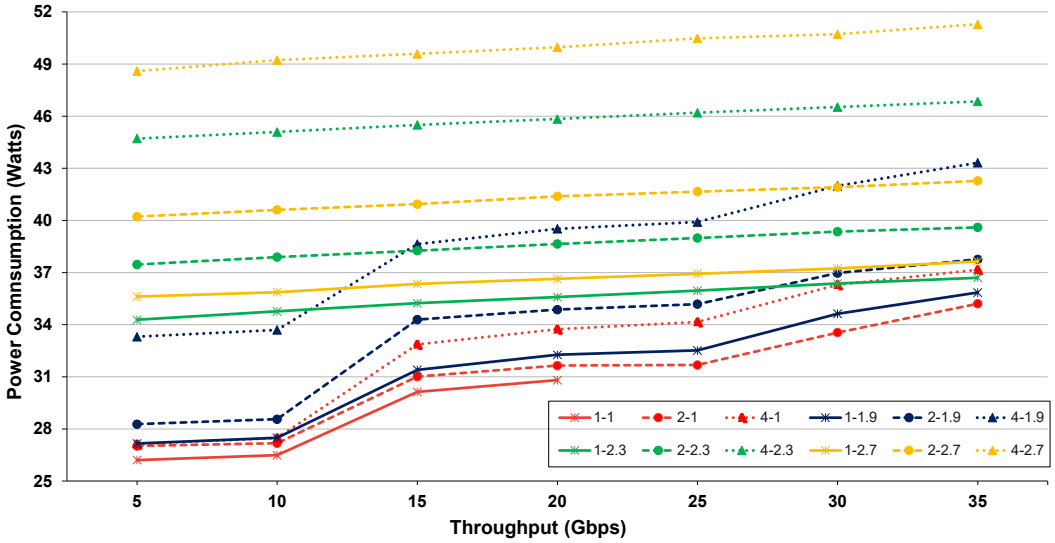


Figure 3-42: UPF resource allocation – Power consumption and achieved throughput according to number of CPUs and CPU frequency. For instance, 4-2.3 stands for 4 CPUs at 2.3 GHz

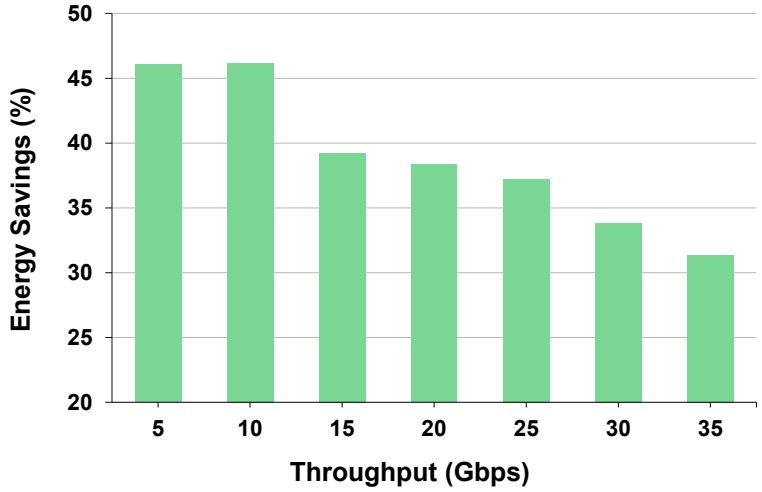


Figure 3-43: UPF resource allocation – Energy Saving benefits according to proposed strategies

Therefore, results at 1 GHz offer the best trade-off between performance and energy consumption, being able to provide the maximum throughput by using 2 CPUs.

As previously mentioned, in real deployments that do not apply energy efficiency optimizations, the default resource allocation for the UPF typically involves activating all CPU cores and setting them to the maximum frequency (i.e., performance mode). This approach ensures the system can handle peak throughputs without the need for restarting the system but will lead to a very high energy consumption. On the other hand, by applying a dynamic setting of CPU frequency and of CPU-worker assignment, we could allocate the required resources in each period according to the incoming traffic demand. Figure 3-43 illustrates the achievable energy savings in our setup when considering the performance (i.e., 4-2.7 in Figure 3-36) and the optimal (i.e., a combination of 1-1 and 2-1 in Figure 3-36) modes. As in previous cases, the energy saving benefits are more relevant at lower loads, but even under high traffic demands savings can still reach approximately 30%.

Future work will include the evaluation of these strategies according to real traffic dynamics in uplink and downlink directions by using the P-GW dataset from an MNO. Additionally, we will also integrate and evaluate traffic forecasting to enable proactive decision-making.

3.6 Joint orchestration of vRANs and Edge AI services

In this section we address the problem of the joint orchestration of virtualized RANs and edge AI services. This problem is already introduced the previous BeGREEN D4.1 [1] and it is crucial for the development of energy efficient services at the edge of the network.

In the following, we first extend the experimental characterization of the system introduced in BeGREEN D4.1, and then we detail the design of a Bayesian online learning algorithm to tackle the problem. In BeGREEN D4.3, we plan to evaluate the proposed algorithm experimentally. Specifically, we will conduct an analysis of convergence under stationary and dynamic network conditions, an empirical study of optimality and a comparison with state-of-the-art ML approaches in terms of data efficiency and adaptability to changes in the requirements of the system.

3.6.1 Use case

The performance indicators and policies were already introduced in BeGREEN D4.1 [1]. We provide a summary as follows:

- Performance indicators:
    - *Service delay:* End-to-end delay that includes the image pre-processing at the user side, its transmission, the processing at the server (GPU delay), and the return of the bounding boxes and labels.
    - *Mean Average Precision (mAP):* It is used to quantify the service accuracy [60].
    - *Server power consumption:* Power cost associated with the computational load of the service's requests, which is dominated by the GPU power consumption.
    - *Base Station power consumption:* Power consumption associated with processing the baseband unit in a virtualized RAN environment.
- Policies:
    - *Image resolution:* This policy sets the average encoding of every image (number of pixels) which the service can enforce.
    - *Radio Airtime:* This radio policy imposes a constraint on the radio resources (duty cycle) the vBS allocates to the service traffic.
    - *GPU speed:* The server's policy is a GPU power limit that adapts the processing speed of a GPU (or a pool of GPUs) in a slice to meet the adopted power constraint.
    - *Radio MCS:* This policy imposes a constraint on the maximum MCS eligible by the vBS to transport the service's data over the air.

In BeGREEN D4.1, we characterized the relationship between the service delay and the server power consumption through the image resolution and the airtime. Moreover, we also analysed the impact of the MCS policy, airtime and image resolution on the power consumption of the BS. Now, we extend the characterization to analyse the behaviour of the system as a function of other policies.

Figure 3-44 shows the trade-off between delay and mAP for the COCO dataset images encoded with different resolutions. The remaining configuration policies are fixed. The findings reveal interesting and quantifiable trade-offs: (i) Higher-resolution images carry more pixels encoded in a larger amount of data; thus, they incur a higher delay due to longer transmission time over the radio interface. (ii) Lower-resolution images cause the service to provide lower mAP performance because they carry less information for the object detection engine. Specifically, we measured a 72% improvement in delay at the expense of precision reduction ranging between 10% to 50%.

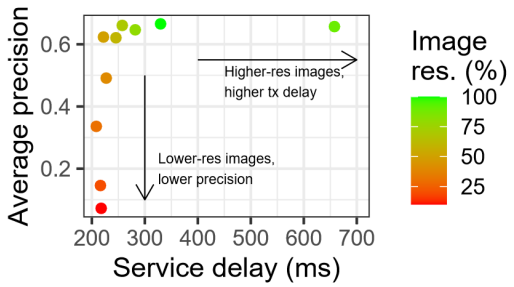


Figure 3-44: vRAN and Edge - Mean average precision (mAP) vs. service delay for images with different resolutions.

Figure 3-45 (top) depicts the service delay and the server's power consumption for several image resolution configurations. We now fix the airtime to 100% and vary the policy allocating computing resources. A higher amount of computing resources increases the server's power consumption, as we are relaxing the power limit imposed to the GPU. We observe that low-res images contribute to increasing the server's power consumption as the rate of requests also grows. However, it is interesting to note that higher-res images ease the work on the GPU, as evidenced by Figure 3-45 (bottom), which shows the delay associated with the GPU tasks only. All in all, despite this improvement in the GPU delay, the corresponding increase in transmission delay when using higher-res images dominates. It is important to observe that, while this is true in our experimental testbed, it may well be different for diverse deployments (e.g., a more energy-efficient GPU, or a higher-bandwidth RAN). This motivates the need for learning algorithms that adapt to the different deployments.

The trade-off presented before between service delay and the server power consumption, certainly appears for other performance metrics, such as the mAP. To assess this, Figure 3-46 shows the mAP achieved by the service as a function of the server's power consumption for various image resolutions. The findings confirm the service cost depends on the mAP.

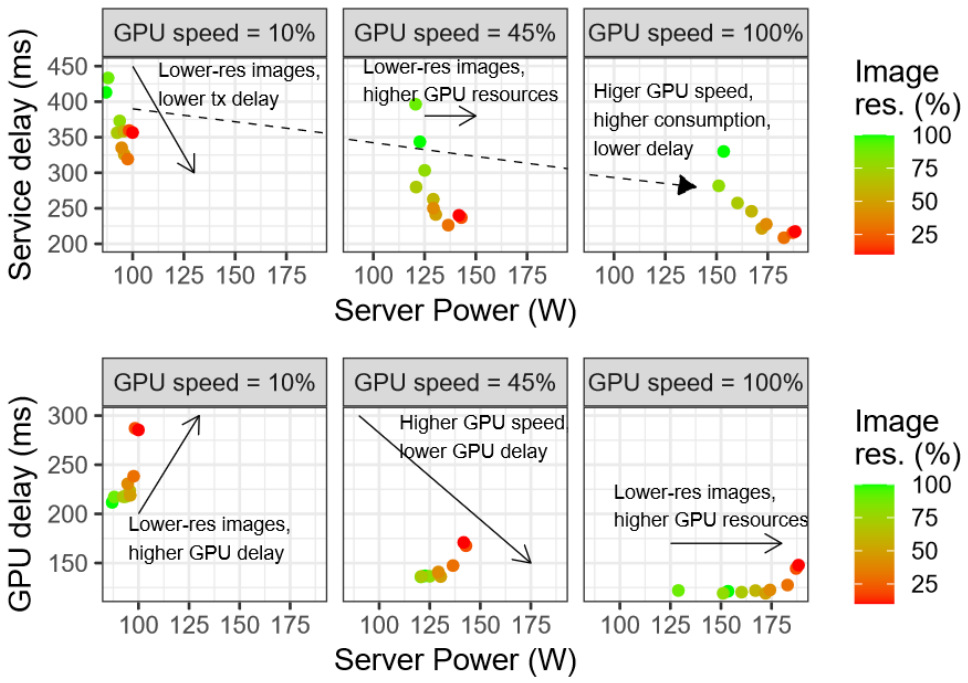


Figure 3-45: vRAN and Edge - Delay vs. server's power consumption for images with different resolutions and GPU policies.

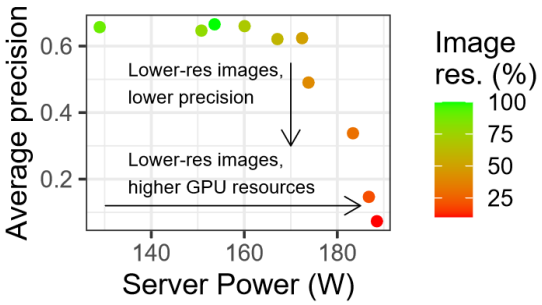


Figure 3-46: vRAN and Edge - Mean average precision vs. server's power consumption for images with different resolutions. The radio and computing resources are allocated to minimize the delay.

Importantly, however, the relationship with mAP is substantially different from that with the service delay. In this case, higher mAP performance actually requires less server power consumption. The reason lies in the fact that higher-res images (which render higher mAP) facilitate object detection and hence require less computation, see Figure 3-46 (bottom).

As a conclusion of our experimental characterization, we would like to highlight that our system consists of a large number of intertwined parameters with non-trivial effects on the performance and energy consumption. As a consequence, we resort to model-free machine learning approaches to design a controller that adapts autonomously to context changes and the vBS and server hosting platforms. We provide a summary of the experimental findings from this section in Table 3-10.

Table 3-10: vRAN and Edge - Summary of the Experimental Findings

Control Policy	Impact on the Performance Indicator
Image Resolution	Higher image resolution implies higher delay, better mAP, and lower GPU delay.
Airtime	Higher airtime implies lower delay, and higher server and BS consumed power.
GPU Speed	Higher GPU speed implies higher server consumed power and lower delay.
MCS	Higher MCS reduces the consumed power at the vBS with low traffic or the opposite with high traffic.

3.6.2 Solution design

In this section, we design an online learning algorithm that solves the problem defined in D4.1, which is formulated as a contextual bandit. Most of the existing contextual bandit algorithms assume a linear relationship between the contexts-control space and the associated reward [61]; or assume a certain structure in the reward function [62]. However, as our experimental characterization reveal, our performance metrics have a non-linear and unknown curvature, but we do observe a high correlation with the control policies. That is, a small change in one of the policies (e.g., image resolution) will produce a small change in delay and power. This allows us to get information about unobserved context-control points via nearby points, hence reducing the exploration time.

Based on the above points, we propose a Bayesian online learning method that models the cost and constraint functions as samples of Gaussian Processes (GPs) over the joint context-control space. This non-parametric estimator deals with the aforementioned non-linearities and correlations, and quantifies the function estimation uncertainty, addressing effectively the exploration vs. exploitation trade-off.

Function approximator:

In order to estimate the cost and constraint functions we use GPs, which consist of a collection of random variables that follow joint Gaussian distributions [63]. Let $z \in \mathcal{Z} = \Omega \times \mathcal{X}$ denote a context-control pair. We model each of the unknown functions as a sample from $GP\big(\mu(z), k(z,z')\big)$, where $\mu(z)$ is its mean function and $k(z,z')$ denotes its kernel or covariance function. Without loss of generality, we assume $\mu = 0$ and

$k(z,z) < 1$, which we refer to as the *prior distribution*, not conditioned on data. Given the prior distribution and a set of observations, the *posterior distribution* can be computed using closed-form formulas.

The sets of observations of the cost and constraint functions at points $Z_T = [z_1, \dots, z_T]$ up to time period T are denoted by $y_T^{(0)} = [u_1, \dots, u_T]$, $y_T^{(1)} = [d_1, \dots, d_T]$, $y_T^{(2)} = [\rho_1, \dots, \rho_T]$, respectively, assuming i.i.d. Gaussian noise, $\sim N\left(0, \zeta_{(\mathrm{i})}^2\right)$. The posterior distribution of these functions follows a GP distribution with mean and covariance:

$$\mu_T^{(\mathrm{i})}(z) = k_T^{(\mathrm{i})}(z)^\top \left(K_T^{(\mathrm{i})} + \zeta_{(\mathrm{i})}^2 1_T\right)^{-1} y_T^{(\mathrm{i})} \quad (1)$$

$$k_T^{(\mathrm{i})}(z,z') = k^{(\mathrm{i})}(z,z') - k_T^{(\mathrm{i})}(z)^\top \left(K_T^{(\mathrm{i})} + \zeta_{(\mathrm{i})}^2 1_T\right)^{-1} k_T^{(\mathrm{i})}(z') \quad (2)$$

where $k_T^{(i)}(z) = \left[k^{(i)}(z_1, z), \dots, k^{(i)}(z_T, z)\right]^\top$, $K_T^{(i)}(z)$ is a kernel matrix defined as $\left[k^{(i)}(z,z')\right]_{z,z' \in Z_T}$, I_T is the T-dimension identity matrix, and $\zeta_{(\mathrm{i})}^2$ the variance of noise in observations. Index i denotes the objective function, with $i = 0$ for the cost function, $i = 1$ for the delay, and $i = 2$ for the mAP. The distribution of unobserved values of $z \in \mathcal{Z}$ for function i is computed from the prior distribution, vector Z_T and the observed values $y_T^{(\mathrm{i})}$ using the equations above.

Kernel selection:

The kernel shapes the GP's prior and posterior distributions, and thus encodes the correlation of the function values for every pair of context-control points. In other words, the kernel characterizes the *smoothness* of the functions [64]. The properties of the kernel should be thoroughly selected for each specific application and the functions to be learned.

We observe in our experimental characterization that the performance indicator functions exhibit different smoothness for each dimension (control policy). In order to approximate these functions accurately, we select our kernel function satisfying two properties: *stationarity* and *anisotropicity*.

This means that $k(z,z')$ is invariant to translations in $\mathcal{Z}$ but not invariant to rotations in $\mathcal{Z}$. The kernel smoothness for each dimension of function *i* is encoded in the length-scale vector $\mathcal{L}^{(i)} = \left[l_1^{(i)}, \dots, l_N^{(i)}\right]$, where N is the number of dimensions of $\mathcal{Z}$. The distance between two points based on the length-scale vector is:

$$d^{(i)}(z,z') = \sqrt{(z-z')^\top (L^{(i)})^{-2} (z-z')},$$

where $L^{(i)} = \mathrm{diag}\left(\mathcal{L}^{(i)}\right)$ is a diagonal matrix of the length-scale vector. In order to satisfy the properties stated above, we select the Matérn kernel on its anisotropic version [63]. Moreover, following standard practice, we particularize it with parameter $\nu = \frac{3}{2}$ (details in [63]), indicating that the function is at least once differentiable. Thus, the expression of the kernel can be particularized as follows:

$$k^{(i)}(z,z') = \left(1 + \sqrt{3} d^{(i)}(z,z')\right) \exp\left(-\sqrt{3} d^{(i)}(z,z')\right)$$

Note that although we are using the same kernel for all cost and constraint functions, their hyperparameters differ and depend on each function's shape. In fact, $\mathcal{L}^{(i)}$ and noise variance $\zeta_{(\mathrm{i})}^2$ should be optimized for each function i before running the algorithm, by maximizing the likelihood estimation over prior data. During execution, the hyperparameters remain constant, since otherwise (optimized with newly acquired data) it is not guaranteed the GPs' confidence interval will cover the actual function within, causing the optimization to fall into poor local optima [65].

Safe set:

It is crucial to identify first which controls satisfy the constraints, which, however, depends also on the

context. For instance, when the user's channel quality decreases (the context changes), the user uses a lower MCS, which increases the transmission time hence increasing the service delay. Therefore, the controls that are suitable for high channel quality may not meet the delay constraint with low channel quality. We define the safe set as the set of policies that satisfy all the constraints for a given context c:

$$S(c) = \{ x \in \mathcal{X} \mid d(c,x) \leq d^{max} \wedge \rho(c,x) \geq \rho^{min} \}.$$

Nevertheless, the computation of the safe set is challenging. Firstly, the observations of the performance metrics are noisy due to the stochastic nature of the system (e.g., noise in the measurements, random variations in the performance), as we observed in the experimental characterization. And secondly, the number of controls $|\mathcal{X}|$ is very large in practice, making it unattainable to explore all possible configurations, for all possible contexts. For these reasons, we use GPs to compute an estimation of the safe set:

$$S_t = S_0 \cup \{ x \in \mathcal{X} \mid \mu_{t-1}^{(1)}(c_t,x) + \beta\sigma_{t-1}^{(1)}(c_t,x) \leq d^{max} \wedge \mu_{t-1}^{(2)}(c_t,x) - \beta\sigma_{t-1}^{(2)}(c_t,x) \geq \rho^{min} \}.$$

where $\left(\sigma_t^{(i)}(z)\right)^2 = k^{(i)}(z,z)$ and β is a weight parameter. Note that the safe set changes over time for two reasons. First, it is a function of the context and, therefore, when the context changes, the set of control policies meeting the constraints varies. Second, as we get more observations of the constraint functions their estimated values and uncertainties also change, allowing us to compute the safe set more precisely. In other words, at each period t, point z_t is observed and vectors Z_t and $y_T^{(i)}$ $\forall i$ are updated. Due to their correlation, the posterior distribution of points near z_t will be updated, hence affecting which controls will be included in the safe set.

Acquisition function:

It indicates, at each time period t, which control x_t shall be used in the system given context c_t. This task is crucial for the convergence of the algorithm and needs to interleave an exploration process in order to expand the safe set while seeking a safe control with high performance. Many previous works have proposed acquisition functions for constrained Bayesian optimization [66] [67] [68], but they do not consider contexts. To the best of our knowledge, SafeOpt [67] is the only work using contexts. However, while SafeOpt provides theoretical performance guarantees, we found in our experiments that its acquisition function has overly slow convergence; an issue that has been reported in other works as well, e.g., [69]. Therefore, we expand this approach by using the contextual Lower Confidence Bound (LCB) proposed in [70] as an acquisition function, but *constrained to safe set*, i.e.:

$$x_t = argmax_{x \in S_t} \mu_{t-1}^{(0)}(c_t,x) + \sqrt{\beta_t}\sigma_{t-1}^{(0)}(c_t,x)$$

Algorithm 3-6 summarizes the whole workflow our solution. At the beginning of the time period t, the context c_t is observed (line 4). Based on the observed context c_t and the vectors Z_{t-1} and $y_{t-1}^{(i)}$ $\forall$ i from the previous time period, the posterior distribution of all the functions is computed using eq. (1)-(2) (line 5). Note that when we do not have observations (i.e., Z_0 and $y_0^{(i)}$ are empty sets, $\forall$ i) the posterior distribution is equal to the prior distribution. Using the expectation and uncertainty of the constraint functions and the equation to estimate the safeset, the safe set S_t is built (line 6).

Algorithm 3-6: vRAN and Edge – Solution Workflow

1	Inputs: Control space $\mathcal{X}$, kernel k, β, δ_1, δ_2, ρ^{min}, d^{max}
2	Initialize $y_0^{(i)}$ $\forall i$ Z_0 as empty sets.
3	For $t = 1,2,\ldots$ do
4	Observe context c_t
5	Compute $\mu_{t-1}^{(i)}$, $\sigma_{t-1}^{(i)}$ $\forall i$, based on Eq. (1) and (2)

6	Estimate the safe set of actions $S_t = S_0 \cup \{ x \in \mathcal{X} \mid \mu_{t-1}^{(1)}(c_t, x) + \beta\sigma_{t-1}^{(1)}(c_t, x) \leq d^{max} \wedge \mu_{t-1}^{(2)}(c_t, x) - \beta\sigma_{t-1}^{(2)}(c_t, x) \geq \rho^{min} \}$
7	Select the control policy $x_t = argmax_{x \in S_t} \mu_{t-1}^{(0)}(\omega_t, x) + \sqrt{\beta_t}\sigma_{t-1}^{(0)}(\omega_t, x)$
8	Observe $d_t(c_t, x_t)$, $\rho_t(c_t, x_t)$, $p_t^s(c_t, x_t)$, and $p_t^b(c_t, x_t)$ at the end of decision period t
9	Compute the cost $u_t(c_t, x_t) = \delta_1 p_t^s(c, x) + \delta_2 p_t^b(c, x)$
10	Update $Z_t \leftarrow Z_{t-1} \cup z_t := [c_t, x_t]$
11	Update $y_t^{(0)} \leftarrow y_{t-1} \cup u_t(\omega_t, x_t)$
12	Update $y_t^{(1)} \leftarrow y_{t-1} \cup d_t(\omega_t, x_t)$
13	Update $y_t^{(2)} \leftarrow y_{t-1} \cup \rho_t(\omega_t, x_t)$
14	End for

The control x_t is selected from the safe set S_t based on the posterior distribution of the cost function and the acquisition function (line 7). At the end of the time period t, all the performance indicators are observed. Then, the cost function is computed. Finally, the new context-control pair z_t, the value of the cost function $u_t(c_t, x_t)$ and the value of the constraint functions ($d_t(c_t, x_t)$ and $p_t(c_t, x_t)$) are added to their respective vectors to generate Z_t and $y_t^{(i)}$ $\forall i$ (lines 10-13).

Note that our solution does not expand explicitly the safe set like in other works such as [67] [66]. These works propose an explicit expansion of the safe set by intentionally exploring controls in the boundary. The objective is to converge to the true safe set and therefore to reach the optimal safe control. However, we found that our acquisition function can both minimize the cost function and expand the safe set.

The reason is that control policies with lower values of power consumption are usually in the boundary of the constraint (e.g., they are associated with higher service delay). Hence, when the acquisition function explores lower power controls it is indirectly exploring the boundaries of the constraint, reducing its uncertainty and thus expanding the safe set. In other words, the acquisition function exploits the problem structure to efficiently expand the safe set.

Practical Issues:

It is interesting to note that, if the performance bounds (constraints) are very tight and the problem is infeasible, the safe set will converge to the initial safe set, that is, $\lim_{t \to \infty} S_t = S_0$ (since S_0 is always included in S_t, Algorithm 3-6, line 5). This might happen only for certain contexts, e.g., for very low channel quality. In any case, our solution will select control policies from the initial safe set S_0, which are intentionally selected to be the ones with the lowest delay, the highest mAP and, therefore, the highest consumed power. On top of that, the proposed algorithm is robust to changes in the constraint settings, and hence can adapt if, for example, the operator decides to relax them during the system runtime in order to avoid such infeasibilities. We demonstrate this in the next section. Finally, it is worth mentioning that the computation of the posterior distribution in eq. (1)-(2) is $O(N^3)$. However, we found in our experiments that this does not introduce any delay since we have a wide enough time window to update the control policy, according to O-RAN specifications.

3.7 Intelligence Plane validation

In this section, we report the initial validation of the Intelligence Plane, which is based on the demonstration performed at the 2024 EuCNC & 6G Summit (EuCNC'24). The main objective of this validation was to assess the baseline architecture and operations of the Intelligence Plane, focusing on two key components: the AI

Engine and the Non-RT RIC. In particular, we considered the energy-efficient 5G carrier on/off switching use case presented in Section 3.3, developing and integrating the required ML models and AIA rApps.

3.7.1 Solution design and use case

As introduced in Section 2, the AI Engine, which leverages MLOps framework, hosts the BeGREEN ML models, whose outputs are exposed to control rApps through the AIA rApps. In this initial validation, we considered the exposure of two ML models, 5G sector energy and load predictor introduced in Section 3.3.2, and the Energy Score function presented in Section. Each of the models/functions were served through specific real-time pipelines of MLRun using the Nuclio serverless frameworks. At the Non-RT RIC domain, the specific AIA rApps were deployed and registered as data producers using the OSC's ICS component, which implements the R1 interface. Finally, the control rApp subscribed to the outputs of these functions and used them to drive the decisions of the on/off switching control loop. In the demonstration performed at the EuCNC'24, we exposed the outputs from the functions hosted in MLRun through a Prometheus exporter embedded in the control rApp, visualizing it in a Grafana dashboard. Future work is to generate the A1 Energy Saving (ES) policy, according to the definition presented in 2.1.2, and send it to the Near-RT RIC and the associated Energy Saving xApp. In addition, we will also incorporate RAN telemetry producer rApps processing and exposing online data obtained from the RAN; in this demo, this data was obtained from the offline dataset presented in section 3.3.1.

Figure 3-47 illustrates the components and interfaces involved in this validation.

3.7.2 Initial evaluation

This subsection presents the initial validation of the Intelligence Plane according to the abovementioned use case. The validation is mainly reported in the form of screenshots. Interested readers can also refer to the published video showcasing the demonstration[24].

The Non-RT RIC was implemented as a Kubernetes cluster in an Intel NUC9i7QNX with the following specs: i7-9750H CPU, 64 GB RAM, and 1 TB SSD. The AI Engine was deployed in an Intel NUC10i7FNH with the following specs: i7-10710U CPU, 64 GB RAM, and 1 TB SSD.

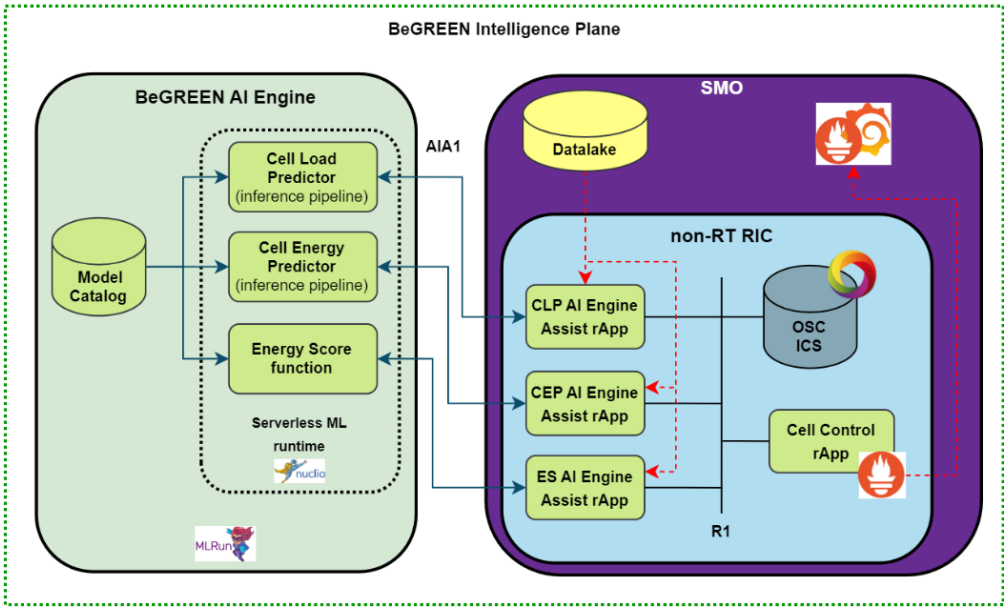


Figure 3-47: Intelligence Plane - EUCNC'24 demo and initial validation

[24] https://youtu.be/_N0JY0Sepgc?si=5cdNPETnWiqZtvHK

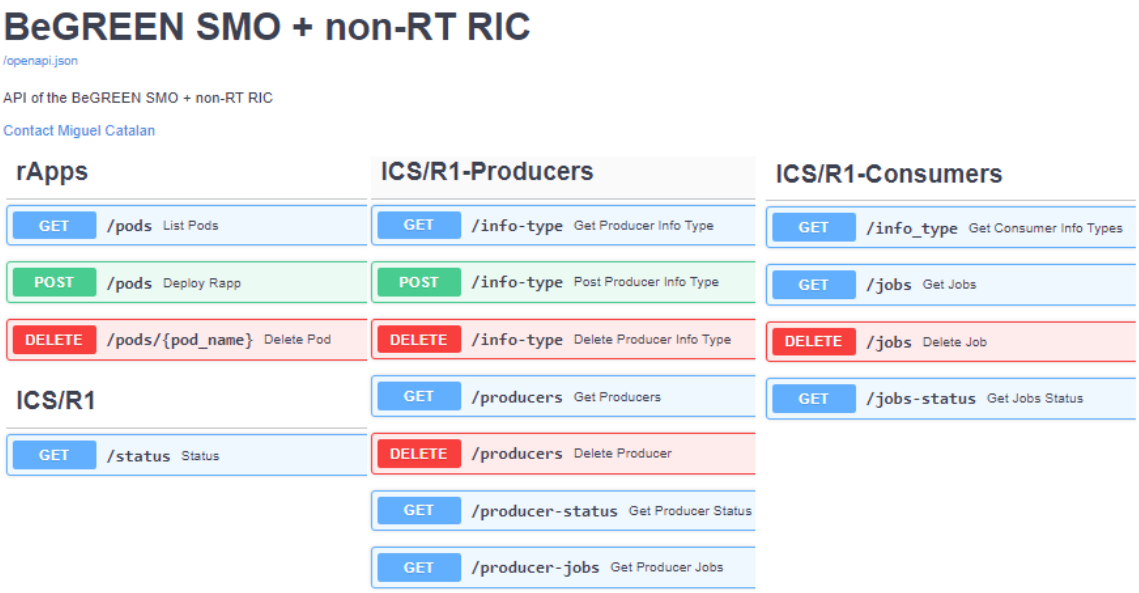


Figure 3-48: Intelligence Plane - BeGREEN SMO and Non-RT RIC REST API - initial validation

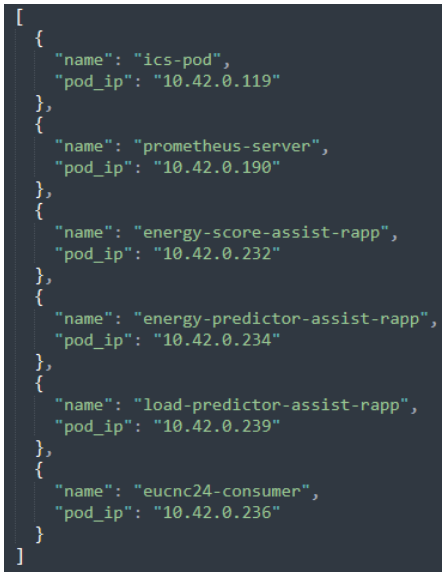


Figure 3-49: Intelligence Plane - Pods active in the Non-RT RIC k8s cluster during validation

3.7.2.1 Non-RT RIC functions

Non-RT RIC components, such as the ICS and the Apps, are deployed as pods and services in a Kubernetes cluster. In this validation, the focus was on the management through the SMO/Non-RT RIC REST API and the ICS component of the data types and the associated producer and consumer rApps. Figure 3-48 shows the main endpoints of the Non-RT RIC API, while the ICS API definition can be found in the OCS online documentation[25]. As future work, we plan to integrate and validate the management of A1 policies and the communication through A1 with the Near-RT RIC interface.

First, the REST API of the SMO allows to manage the lifecycle of the rApps, through POST (deploy) and DELETE (undeploy) methods. During deployment, the docker images of the rApps are retrieved from a registry and instantiated as pods according to the specified environment variables.

[25] https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric-plt-informationcoordinatorservice/en/latest/ics-api.html

Code | Details

200

Response body


```
{
  "status": "hunky dory",
  "no_of_producers": 3,
  "no_of_types": 3,
  "no_of_jobs": 3
}
```


Figure 3-50: Intelligence Plane - ICS/R1 status during validation

In addition, an associated Kubernetes service is created to facilitate communication between producers, consumers, and the ICS. Figure 3-49 illustrates the active pods during the validation of the use case.

As detailed in Section 2.1.2, OSC's ICS works as in implementation of the DME functions of the R1 interface. Through its API, it can be obtained the number of active producers, information or data types, and jobs or subscriptions, as illustrated in Figure 3-50.

Information types are used to identify the data that is generated by the producer rApps and consumed by the consumer rApps. The Non-RT RIC API implements POST, GET, and DELETE methods to manage them. The POST method allows to define the required information to generate the data ("job_definition") and the output that will be generated ("job_data"), following the model that was illustrated in 2.1. Note that once one or more producers are associated with a specific information type, this type cannot be deleted before the producers get undeployed. As an example, Figure 3-51 depicts the created information types defining the cell load predictor (top) and the energy score (bottom). As an example, Figure 3-51 depicts the created information types defining the cell load predictor (top) and the energy score (bottom).

Figure 3-52 depicts the workflow followed for the registration of the required information types for the ML models and the energy score.

```
{
  "info_job_data_schema": {
    "job_definition": {
      "cell_ids": "list of cells ids (list of str)",
      "period": "frequency of the predictions in seconds (int)"
    },
    "job_data": {
      "cell_id": "cell id (str)",
      "prediction": "prediction value (float)",
      "accuracy": "prediction value if available (float)"
    }
  },
  "info_type_information": {
    "description": "Prediction of the load consumed by a cell according to different KPIs"
  }
}
```

```
{
  "info_job_data_schema": {
    "job_definition": {
      "cell_ids": "list of cells ids (list of str)",
      "period": "frequency of the predictions in seconds (int)"
    },
    "job_data": {
      "cell_id": "cell id (str)",
      "score": "score value (float)"
    }
  },
  "info_type_information": {
    "description": "Energy score of a cell according to its load and consumed energy"
  }
}
```

Figure 3-51: Intelligence Plane - Example of information type definition

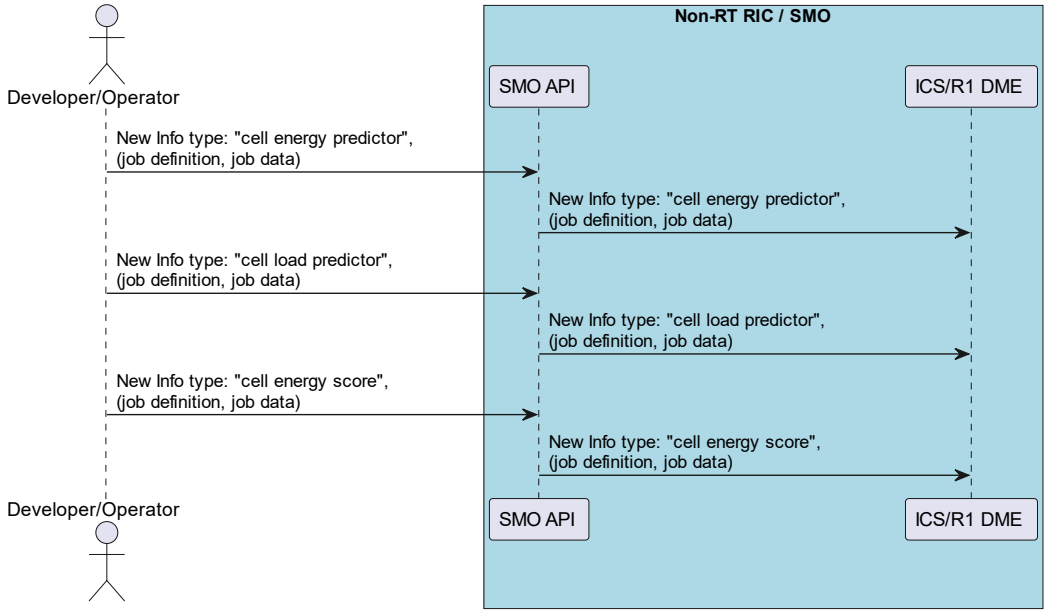


Figure 3-52: Intelligence Plane– Workflow for definition of information types for ML models

3.7.2.2 AIA rApps

AIA rApps mainly act as data producers, exposing the real-time pipelines of MLRun to the control rApps. In the case of ML models, MLRun also allows the creation of training and monitoring pipelines or the analysis of feature importance; the incorporation of these operations will be explored in BeGREEN D4.3. Nevertheless, pre-trained models or other non-ML functions can be easily incorporated into the framework and served through Nuclio. Figure 3-53 shows the ML functions view of MLRun which lists some of the models and functions being validated.

The command column shows the IP and port where the real-time function will be exposed and that will be triggered by the AIA rApp to get its outputs. Figure 3-54 shows how these functions are listed in the Nuclio framework.

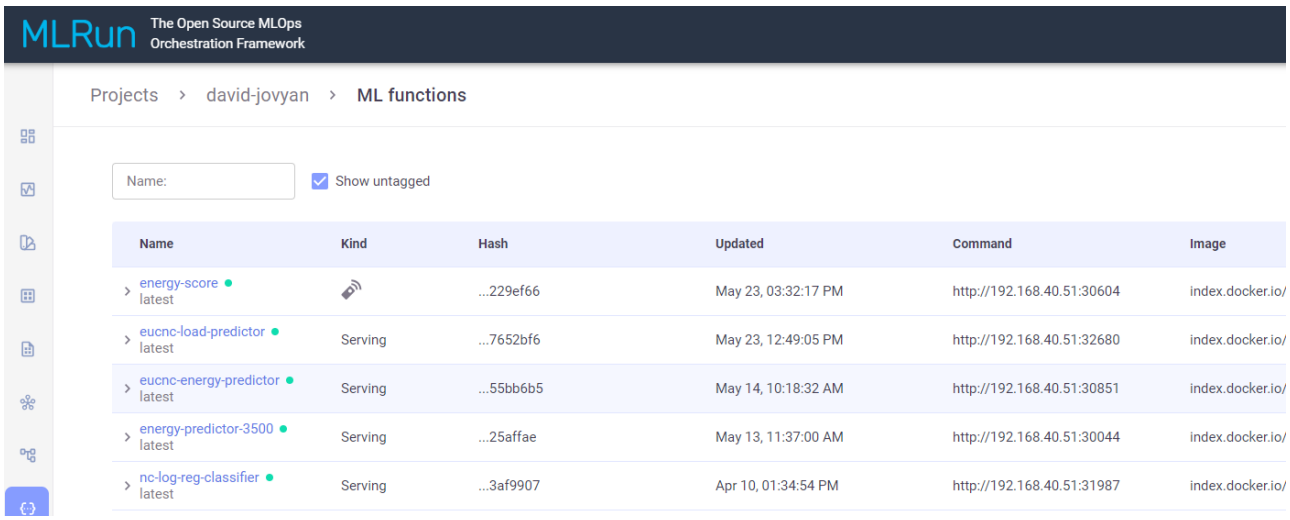


Figure 3-53: Intelligence Plane - MLRun: ML functions view

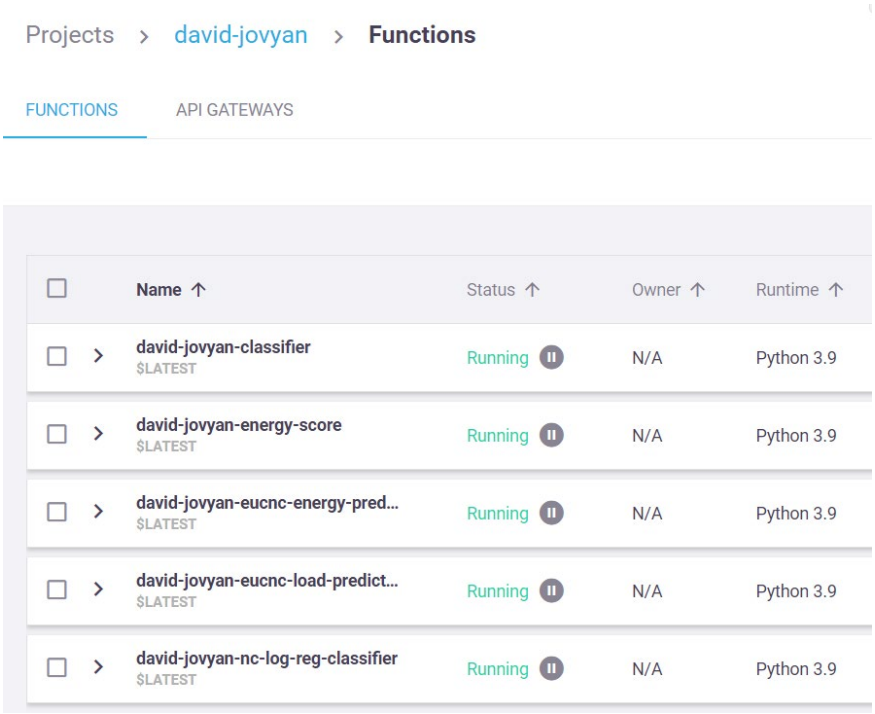


Figure 3-54: Intelligence Plane - Nuclio: Real-time functions view

```
{
  "name": "energy-predictor-assist-rapp",
  "image": "gitlab.i2cat.net:5050/areas/mwi/o-ran/rapps/predictor_energy_assist_rapp:latest",
  "ports": [
    "8092"
  ],
  "env": {
      "ICS_URL": "http://ics-service:8083",
      "NAME": "energy-predictor-assist-rapp",
      "PRODUCER_PORT":"8092",
      "ML_URL": "http://192.168.40.51:30851/v2/models/xgboost"
  }
}
```

Figure 3-55: Intelligence Plane - Energy Predictor AIA rApp deployment

200

Response body

```
{
  "supported_info_types": [
    "energy_cell_prediction"
  ],
  "info_job_callback_url": "http://energy-predictor-assist-rapp-service:8092/jobs",
  "info_producer_supervision_callback_url": "http://energy-predictor-assist-rapp-service:8092/supervision"
}
```

Figure 3-56: Intelligence Plane - Energy Predictor AIA rApp ICS registration

AIA rApps are deployed using the POST method of the Non-RT RIC, as shown in Figure 3-55 for the case of the Energy Predictor. Note that the serving endpoint of the ML model (or Nuclio function) is passed as an environmental variable, allowing to reuse the AIA rApp in case a different serving endpoint is available (e.g., a new model offering better accuracy).

Once deployed, the AIA rApp interfaces the ICS to register itself as producer of the associated information type, specifying the required endpoints for job creating and supervision as shown in Figure 3-56.

Then, the ICS/R1 will use the “info_job_callback_url” to register new jobs according to consumer subscription demands (past or new requests).

```
Load AIA rApp: Received job job='eucnc24-consumer-load_cell_prediction' type='load_cell_prediction'
data=JobData(cell_ids=['CATX0614P3'], period=5) target_uri='http://eucnc24-consumer-service:8093/datadelivery'
owner='eucnc24-consumer' last_updated='2024-06-21T13:17:49.037112Z'
```

Figure 3-57: Intelligence Plane - Registration of a new job in the Load Predictor AIA rApp

```
Load AIA rApp - Sending job: {'producer_name': 'load-predictor-assist-rapp', 'info_type': 'load_cell_prediction',
'job_data': {'result_list': [{'cell_id': 'CATX0614P3', 'prediction': 8.836968421936035, 'accuracy': 0.9235445553322382}]}}
Load AIA rApp - Sending job: {'producer_name': 'load-predictor-assist-rapp', 'info_type': 'load_cell_prediction',
'job_data': {'result_list': [{'cell_id': 'CATX0614P3', 'prediction': 9.919061660766602, 'accuracy': 0.9448294169383542}]}}
```

Figure 3-58: Intelligence Plane - Load Predictor AIA rApp – Data delivery

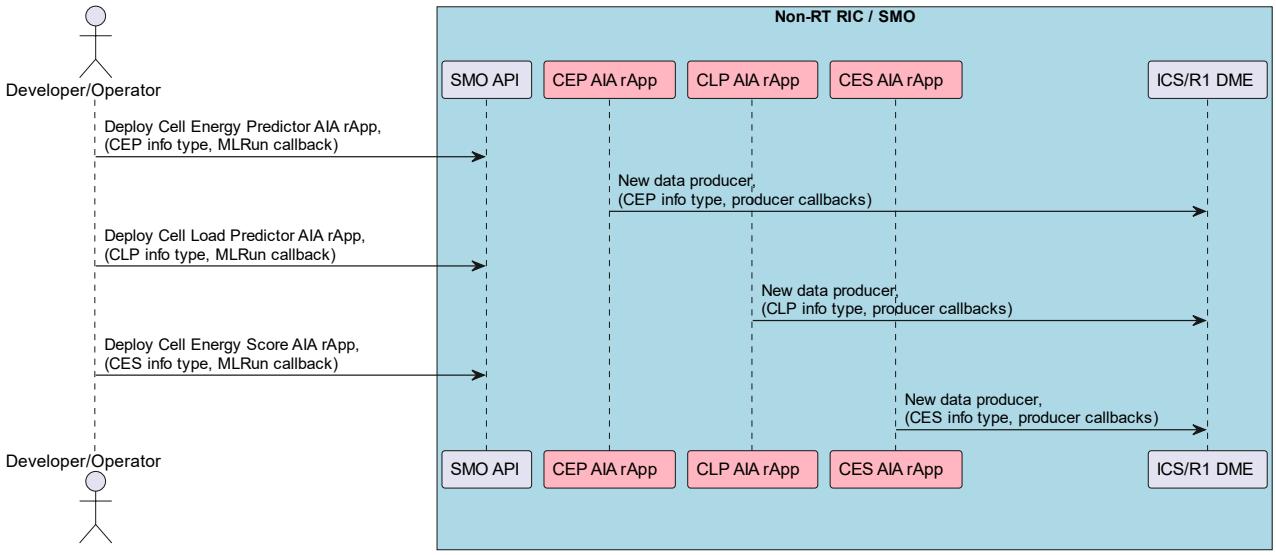


Figure 3-59: Intelligence Plane – Workflow for the deployment of AIA rApps

Figure 3-57 depicts an example of the automated job registration in the case of the Load Predictor AIA rApp, obtaining the required parameters for the job creation as defined in the "job_definition" field of the information type (see Figure 3-51) and endpoint of the consumer rApp for the data delivery.

According to this information, the AIA rApp starts delivering the data to the consumer as obtained from the serving endpoint in the MLRun/Nuclio and the data delivery format specified in the "job_data" field of the information type (see Figure 3-51). In the case of the initial validation scenario, the input data required by the models (e.g., last load value of the cell or number of active UEs) was obtained from a datalake storing the offline MNO dataset. Future work will include the online exposure of these KPIs through a RAN Telemetry rApp. Figure 3-58 shows an example of data delivery in the case of the Load Predictor AIA rApp, which includes the cell id, the predicted value, and the accuracy of the prediction or model.

Figure 3-59 depicts the workflow required to deploy the AIA rApps and register them as data producers through the ICS/R1 interface.

3.7.2.3 Control rApp

The control rApps are also deployed through the POST method in the SMO. In the case of the rApp being used for the initial validation of the Intelligence Plane, shown in Figure 3-60, we specified as environment variables the information required by the different information types to define the job (i.e., cell id and period).

Once deployed, the Control rApp creates the different jobs through the ICS, which interfaces with the required producers as introduced in Section 3.7.1. Note that in case a producer is not available when creating the job, the ICS will store this request and create the subscription once available. Figure 3-61 depicts the process of job generation done by the validated Control rApp, while Figure 3-62 illustrates the created job in the case of the energy cell prediction information type.

```
{
  "name": "eucnc24-consumer",
  "image": "gitlab.i2cat.net:5050/areas/mwi/o-ran/rapps/eucnc24_consumer_rapp:latest",
  "ports": [
    "8093", "9998"
  ],
  "env": {
     "NAME": "eucnc24-consumer",
     "ICS_URL": "http://ics-service:8083",
      "PORT_CONSUMER":"8093",
      "CELLS": "CATX0614P3",
      "JOB_PERIOD": "5",
      "EXPORTER_PORT": "9998"
   }
}
```


Figure 3-60: Intelligence Plane - Control rApp deployment


```
Creating job {'info_type_id': 'energy_cell_prediction',
'job_result_uri': 'http://eucnc24-consumer-service:8093/datadelivery', 'job_owner': 'eucnc24-consumer',
'job_definition': {'cell_ids': ['CATX0614P3'], 'period': 5},
status_notification_uri': 'http://eucnc24-consumer-service:8093/status'}
Creating job {'info_type_id': 'load_cell_prediction',
'job_result_uri': 'http://eucnc24-consumer-service:8093/datadelivery', 'job_owner': 'eucnc24-consumer',
'job_definition': {'cell_ids': ['CATX0614P3'], 'period': 5},
'status_notification_uri': 'http://eucnc24-consumer-service:8093/status'}
Creating job {'info_type_id': 'energy_score',
'job_result_uri': 'http://eucnc24-consumer-service:8093/datadelivery', 'job_owner': 'eucnc24-consumer',
'job_definition': {'cell_ids': ['CATX0614P3'], 'period': 5},
'status_notification_uri': 'http://eucnc24-consumer-service:8093/status'}
```


Figure 3-61: Intelligence Plane - Control rApp jobs generation

200

Response body


```
[
  {
    "info_job_identity": "eucnc24-consumer-energy_cell_prediction",
    "info_type_identity": "energy_cell_prediction",
    "info_job_data": {
      "cell_ids": [
        "CATX0614P3"
      ],
      "period": 5
    },
    "target_uri": "http://eucnc24-consumer-service:8093/datadelivery",
    "owner": "eucnc24-consumer",
    "last_updated": "2024-06-01T08:37:43.6528922"
  }
]
```


Figure 3-62: Intelligence Plane - Control rApp – job information

Finally, as was presented in Section 3.7.1 and illustrated in Figure 3-63, the AIA rApps start generating and delivering the data according to the job definition and the job data format.

As abovementioned, in the case of this initial validation, the control rApp just exposed the predictions and accuracy of the models, plus the energy score, through the Prometheus and Grafana frameworks, as shown in Figure 3-64. The graphs show the evolution of the data and the predictors during a week, depicting every 5 seconds (i.e., job period) a value related to a real 15-minute measurement (i.e., dataset granularity).

Finally, Figure 3-65 depicts the workflow required to deploy the control rApp and register it as data consumer, and the generated non-RT control-loop with the input data obtained from the ML models through the AIA rApps.

```
INFO:     10.42.0.239:51908 - "POST /datadelivery HTTP/1.1" 200 OK
{'producer_name': 'energy-predictor-assist-rapp', 'info_type': 'energy_cell_prediction',
'job_data': {'result_list': [{'cell_id': 'CATX0614P3', 'prediction': 168.22881062825522, 'accuracy': 1.0216324532080276}]}}
energy_cell_prediction prediction and accuracy for cell CATX0614P3:168.22881062825522 1.0216324532080276
INFO:     10.42.0.234:57620 - "POST /datadelivery HTTP/1.1" 200 OK
{'producer_name': 'energy-score-assist-rapp', 'info_type': 'energy_score',
'job_data': {'result_list': [{'cell_id': 'CATX0614P3', 'score': 8407.972555040395}]}}
Energy score prediction for cell CATX0614P3:8407.972555040395
INFO:     10.42.0.232:56102 - "POST /datadelivery HTTP/1.1" 200 OK
{'producer_name': 'load-predictor-assist-rapp', 'info_type': 'load_cell_prediction',
'job_data': {'result_list': [{'cell_id': 'CATX0614P3', 'prediction': 61.97632598876953, 'accuracy': 1.014294773619123}]}}
load_cell_prediction prediction and accuracy for cell CATX0614P3:61.97632598876953 1.014294773619123
INFO:     10.42.0.239:60714 - "POST /datadelivery HTTP/1.1" 200 OK
```

Figure 3-63: Intelligence Plane - Control rApp - data obtention

Figure 3-64: Intelligence Plane - - EUCNC'24 demonstration: Predictors view

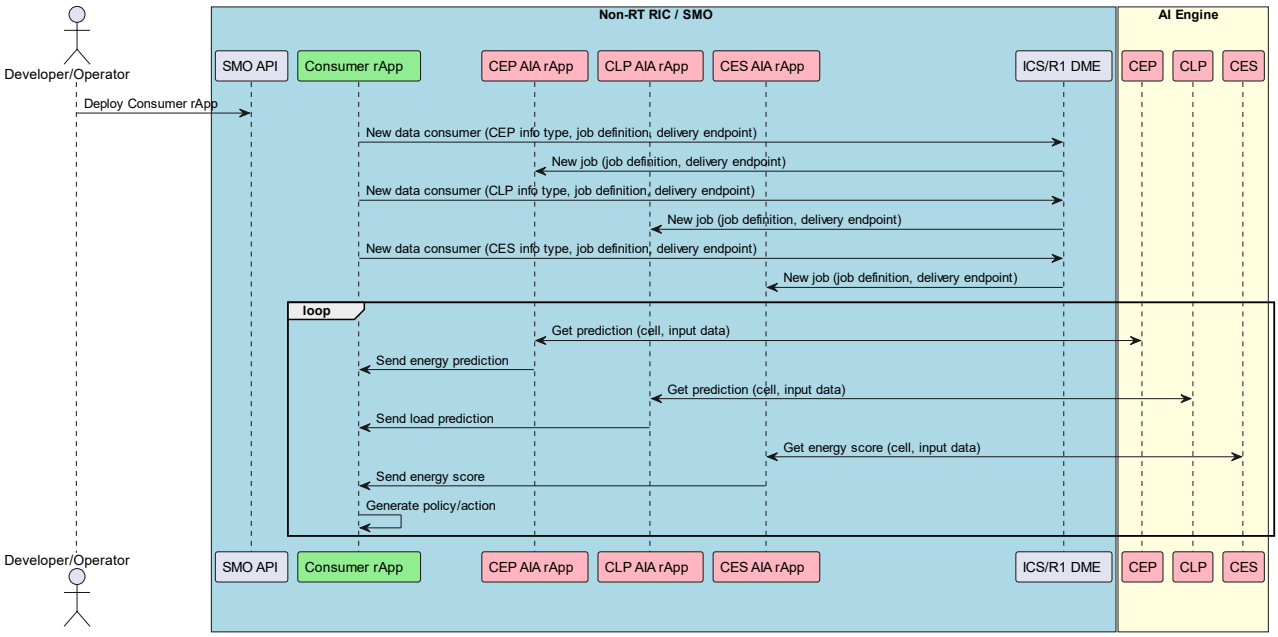


Figure 3-65: Intelligence Plane – Workflow for the deployment of control rApp and non-RT control-loop

3.7.2.4 Initial benchmark of the Intelligence Plane

This section reports an initial benchmark of the capabilities of the Intelligence Plane to provide data to the control rApps with a non-RT granularity (i.e., greater than 1 second). Starting from the previous use case, i.e. a control rApp and three available AIA rApps deployed in the Non-RT RIC, we evaluated the variation of the excess delay according to the number of consumers, the number of jobs (i.e., the number of simultaneous subscriptions of the consumer rApp) and the interval required by the control rApp to get all the predictions (i.e., the period of the control-loop). We calculated the excess delay as the difference between the required period and the measured period. During the evaluation, a new instance of the control rApp was deployed every 60 seconds until we reached the maximum number of consumers (i.e., 45, as shown in Figure 3-66 and Figure 3-67).

We initially compared the results of having a single job versus three simultaneous jobs per control rApp, respectively labelled as 1-X and 3-X in Figure 3-66. To identify the source of delays, we also compared cases where the AIA rApps triggered the inference of the ML models in the AI Engine (denoted as 1-1 and 3-1) with cases where the AIA rApps just send a random value (denoted as 1-0 and 3-0). Results shown in Figure 3-66 show that for a single job, the excess delay was maintained under 1 second during the whole evaluation, although it started to increase linearly after reaching approximately 30 consumers. In the case of 3 simultaneous jobs per consumer, where we measured the delay needed until the consumer obtained a new measurement from each job, the excess delay started early to increase linearly, around 15 consumers. This led to the impossibility of obtaining measurements according to the required interval of one second. According to the results without requesting the AI Engine, denoted as 1-0 and 3-0 in the figure, which didn't experience this linear increase, we can conclude that the excess delay was indeed caused by congestion in the AI Engine endpoints due to too many requests per second.

Nevertheless, we also evaluated the experienced excess delay in the case of 3 jobs when increasing the interval or control-loop period of the consumers. As shown in Figure 3-67, higher intervals alleviated the congestion of the AI rApps caused by the communication with the AI Engine. For instance, with a period of 5 seconds, the delay increase with the number of consumers was almost imperceptible. Note that this period will be enough for the vast majority of non-RT control-loops, since they are usually designed according to periods of tens of seconds or minutes.

Finally, note that in this initial validation the AI Engine was deployed in a single server (Intel NUC); therefore, in an operational deployment with the AI Engine Kubernetes cluster involving multiple nodes, the serverless operation of Nuclio will prevent the experienced delay by applying horizontal pod scaling strategies. We will evaluate this approach in following validations during the project.

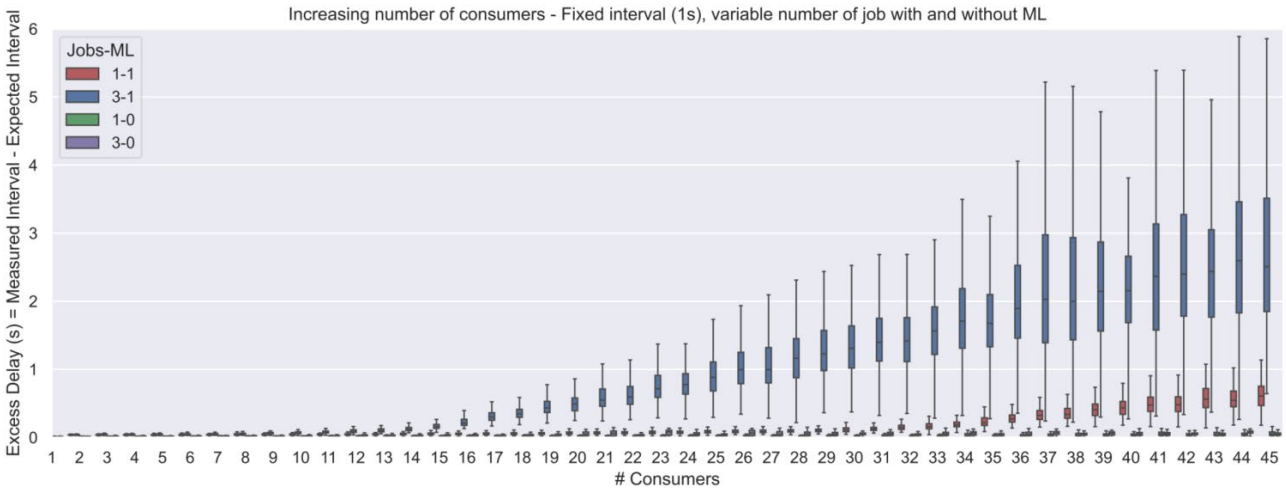


Figure 3-66: Intelligence Plane - Jobs latency with an increasing number of consumers and fixed interval

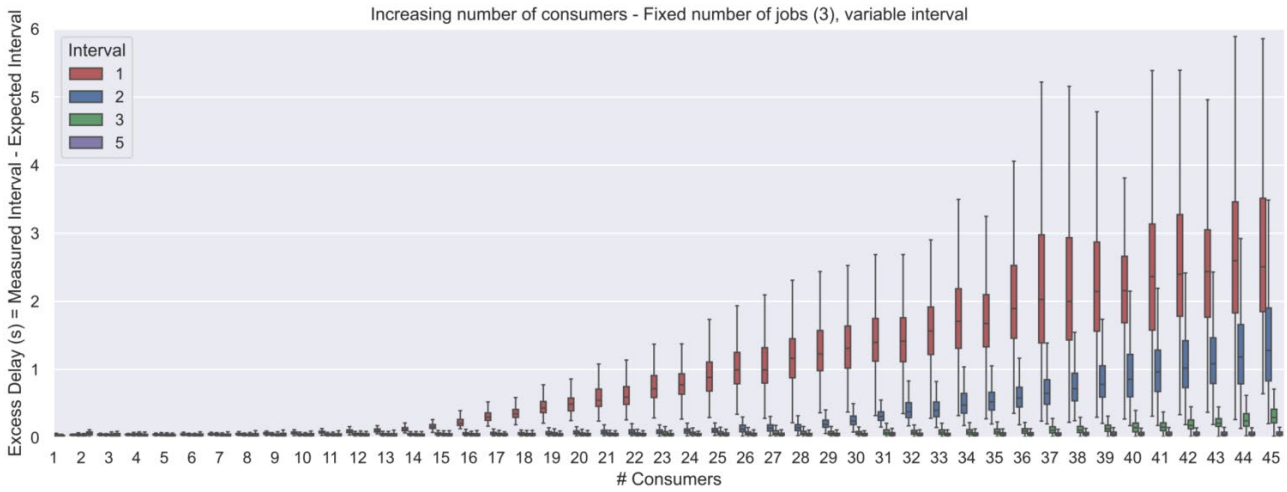


Figure 3-67: Intelligence Plane - ML-based jobs latency with an increasing number of consumers and fixed interval

3.8 Summary of Chapter 3

This section provides a summary of the presented uses cases (Table 3-11) and the associated AI/ML-Assisted Procedures to enhance energy efficiency (Table 3-12).

Table 3-11: Summary of Use Cases

Use Case	Main Objective and Targeted KPIs	AI/ML-Assisted Procedures	Baseline Scenario	Results of the Initial Validation
Dimensionality Reduction	To reduce data volume and CPU cycles in training predictive models	Supervised learning XGBoost Regressor	Assessing when a model with less features can be used without losing significant accuracy.	Data volume and processing minimization in specific retraining scenarios.
Computing Resource Allocation for vRAN	Prediction of the computing resources needed by the vRAN with the objective of reducing energy consumption.	Reinforcement learning. Specifically, the formulation is customized for this specific use case as a contextual bandit problem.	We compare against the optimal configuration of the system, (exhaustive search in the solution space) and SoTA benchmarks in the literature.	This deliverable provides an exhaustive experimental evaluation of the proposed solution. We evaluate the convergence, inference time, and the performance w.r.t. SoTA benchmarks and realistic traffic traces. We measured a 17% of computing resource savings in realistic scenarios.
Energy-Efficient 5G Carrier on/off Switching	Reduce energy consumption (5G carriers) and increase energy efficiency (4G carriers) without impacting QoS (average UE rate).	Load and energy predictors based on XGBoost Regressor. Logistic-Regressor classifier to drive on/off decision.	Measured energy consumption of an actual network according to MNO's dataset.	Restricted to a high-loaded site during a week. Up to 200 kWh of energy savings during a week (off ~50% of the time). Impact on QoS not evaluated.

Use Case	Main Objective and Targeted KPIs	AI/ML-Assisted Procedures	Baseline Scenario	Results of the Initial Validation
Coverage Hole Detection and Relay Placement	Identification of geographical regions with high traffic demands and poor propagation conditions (e.g. low RSRP). These regions may be addressed by energy efficient solutions based on the deployment of relays.	Unsupervised learning algorithms. In particular, a clustering algorithm based on DBSCAN is proposed to group geographical locations with problems in different clusters.	The coverage hole characterization is used as input for a relay placement algorithm to address the identified coverage holes. The energy consumption reduction with the placement of relays is compared with respect to the case where no relays are deployed.	The placement of a fixed relay to address a specific coverage hole avoids increasing the BS transmitted power. The coverage hole can be addressed with a power consumption reduction in the range between 35%-70% depending on the BSand relay power consumption model.
Relay Activation/Deactivation Process	Take adequate decisions of relay activation to serve UEs with poor propagation conditions and adequate decisions of relay deactivation to save energy based on the number of UEs served by the relay.	DQN (Deep Q-Network) that combines Reinforcement Learning (based on Q-learning) and Neural Networks.	We compare the energy consumption reduction by deactivating the relay with respect to the case that the relay is not deactivated.	The energy reduction that can be obtained for the considered relay is around 100Wh each day. It depends on the time the relay is activated. The maximum energy consumption that can be obtained is in the order 150Wh for each relay and each day.
Computing Resource Allocation for UPFs	Dynamically adapt the allocation of CPU resources to UPF load in order to enhance energy efficiency.	Load predictors based on XGBoost Regressor or Prophet forecasting. Additional AI/ML methods to be decided.	Consumption of a VPP-DPDK UPF server with realistic traffic and without energy efficient mechanisms.	Compared to an UPF configured in performance mode to lead with peak traffic demands, energy savings could reach 30%-45% depending on data load.
Joint Orchestration of vRANs and Edge AI Services	Orchestrate the virtualized BS and the AI serviced running at the edge of the network in a joint manner. The objective is to minimize the energy consumption subject to a minimum value of the QoS for the users.	Bayesian Online Learning, combining Gaussian processes used as surrogate function, and tailored acquisition function to handle the exploration-exploitation trade-off.	We compare against the optimal configuration of the system, (exhaustive search in the solution space) and SoTA ML solution in the literature.	We provide a experimental characterization of the problem under study, showing the trade-offs among the different involved variables and performance metrics.

Table 3-12: Summary of AI/ML-based Methods

Method	AI/ML type	Training	Inference
Dimensionality Reduction	Supervised Learning	Real data obtained from live networks. Not publicly available	Required dimensionality of feature set for predictive model.
Computing Resource Allocation for vRAN / RAN optimization and control	Reinforcement Learning	Real data obtained from our experimental platform. The dataset if publicly available at https://ieee-dataport.org/documents/o-ran-experimental-evaluation-datasets	The inference time is analysed in Sec 3.2.2.2. Suitable to run in the O-RAN Non-RT RIC.
Energy-efficient 5G carrier on/off switching	Supervised Learning, Time series regression	Real data from a Spanish MNO (70 sites, 2 months). Private dataset.	Energy consumption of a carrier, non-RT domain
Energy-efficient 5G carrier on/off switching	Supervised Learning, Time series regression	Real data from a Spanish MNO (70 sites, 2 months). Private dataset.	Load of a carrier, non-RT domain
Energy-efficient 5G carrier on/off switching	Supervised Learning, Classification	Real data from a Spanish MNO (70 sites, 2 months). Private dataset.	5G carrier on/off switching decision, non-RT domain
Coverage Hole Detection and Relay Placement.	Unsupervised learning, clustering algorithm.	No training is required. Real data that characterises the time/space distribution of the users. An example and details of the dataset is published in [47]	Coverage hole characterization, non-RT domain.
Relay activation/deactivation process	Reinforcement Learning.	Real data that characterises the time/space distribution of the users. An example and details of the dataset is published in [47]	Decision of relay activation/deactivation, non-RT domain.
CPU management for enhancing UPF Energy Efficiency	Supervised Learning, Time series forecasting	Real data from a Spanish MNO (70 sites, 2 months). Private dataset.	Load of the UPF, non-RT domain
Joint orchestration of vRANs and Edge AI services / RAN optimization and control	Reinforcement Learning	Real data obtained from our experimental platform. The dataset if publicly available at https://ieee-dataport.org/documents/o-ran-experimental-evaluation-datasets	Suitable to run in the O-RAN Non-RT RIC.

4 Summary and Conclusions

The growing demand for 5G networks has brought significant challenges, particularly in managing the energy consumption of network infrastructure to achieve sustainability objectives without impacting the quality of the service and experience. In Work Package 4, the BeGREEN project addresses these challenges by integrating AI/ML within the O-RAN framework, aiming at enhancing the decision-making of automated control-loops focused on energy efficiency optimisation. In this context, this BeGREEN D4.2 has presented the initial evaluation of the BeGREEN Intelligence Plane and of the proposed AI/ML methods to enhance the energy efficiency in the RAN and Edge domains.

Before presenting the initial validation of the Intelligence Plane in Section 3.7, Chapter 2 described its architecture, extending the initial design provided in BeGREEN D4.1 and highlighting the implementation choices. The main conclusions of this deliverable regarding one of its key elements, the AI Engine, are summarised as follows:

- The implementation of the AI Engine relies on two open-source frameworks: MLRun, which provides the required AI/ML services and hosts the trained ML models, and Nuclio, which implements the serverless serving of the ML models.
- ML models are decoupled from control rApps and xApps but exposed to them through associated Assist AI Engine (AIA) rApps/xApps. This approach allows ML model developers to focus on the model implementation and optimisation, while rApp/xApp developers can focus on the control logic and optimisation objectives. It also facilitates ML model reuse by several control rApps/xApps. The reference model of the AIA rApps/xApps is introduced, highlight the implementation choices and the integration with O-RAN interfaces.
- The exposure of ML models through AIA rApps in the non-RT RIC is done by exploiting the DME capabilities of the R1 interface. Particularly, the developed implementation relies on the ICS component provided by the OSC. This way, model outputs are exposed as information types, while AIA rApps, which communicate with the model serving endpoints in the AI Engine, perform as data producers. Finally, control rApps work as data consumers.
- The evaluation of the Intelligence Plane focused on the mentioned points, detailing the demonstration performed at the 2024 EuCNC & 6G Summit. Particularly, it comprehends the definition of different information types related to ML models and serverless functions developed within BeGREEN and hosted in the AI Engine, the deployment of AIA rApps as data producers and the deployment of a control rApp periodically consuming the outputs of the models. A video showcasing the demonstration can be found in BeGREEN YouTube channel[26]. The evaluation also includes a benchmarking of the capabilities of the implementation to provide data to the control rApps with a non-RT granularity, highlighting a trade-off between the number of data consumers and the periodicity of the control loops.
- Future work regarding the exposure of ML models will mainly consist of the integration of the AI Engine with the near-RT RIC to demonstrate near-RT serverless serving. Additionally, the implementation and demonstration of additional AI/ML services such as monitoring and (re)training.

The integration of the non-RT and near-RT RICs towards the management of energy saving optimisations is another of the key aspects being addressed by the Intelligence Plane. This deliverable presented the design choices regarding the implementation of energy saving A1 policies. Also, it introduced the main xApps that will be considered to achieve energy efficiency in cell management, e.g. to perform intelligent on/off

[26] https://www.youtube.com/watch?v=_N0JY0Sepgc

switching: the Energy Saving xApp, which manages the operation status of the cells according to the policies, and the Handover Manager xApp, which optimizes the handover process allowing to apply load balancing strategies. The independent management of these two xApps could lead to conflicts, for instance between energy saving policies and QoS or load-balancing policies. Therefore, a collaborative conflict mitigation approach is also presented in this deliverable. Future work will focus on implementing and demonstrating the coordination between RICs to manage energy saving optimisations and associated conflicts.

The benefits of integrating AI/ML and O-RAN architecture are exploited by two main solutions presented in Chapter 3, whose key findings and future work is summarised as follows:

- Compute resource allocation in vRAN: Considers the problem of allocating resources to vBS in vRAN scenarios under shared computing infrastructure. To this end, a RL-based solution is proposed, which considers the trade-off between channel quality, network demand, the CPU resources being assigned to the pool of vBS and the interference among vBS processes (noisy neighbour problem). Experimental evaluation demonstrates the feasibility of the proposed solutions, which can be applied as a non-RT control loop due to its low inference time, and the energy savings gains, achieving up to 17% reduction in overall computing resource usage without sacrificing throughput. Future work will consider an alternative approach, based on optimising the utilisation of assigned resources by the vBSs.
- AI/ML and data-driven strategies for energy-efficient 5G carrier on/off switching: Explores energy saving opportunities in a 5G NSA deployment by switching off 5G cells and offloading UEs to 4G cells. The presented analysis is based on real data from an MNO, considering the cases where 5G PRBs can be offloaded to the 4G cells of the same site and sector. In the case of a specific high-loaded site in the city centre, this approach could result in the 5G cell being deactivated 56% of the time during the week. According to this scenario, different data and ML-driven approaches are considered to rule the cell on/off switching decision. Future work will extend the analysis of energy saving opportunities to all the sites of the dataset, also analysing the impact on the QoS of the UEs. Additionally, more advanced ML-assisted methods will be proposed.

The integration of new technologies, which could enhance the energy efficiency of the network, but which are currently not being addressed by O-RAN architecture, is also being addressed in the BeGREEN project. In particular, this deliverable discussed the integration of RIS, ICAS and Relays. The integration of RIS will require extending E2 and O1 interfaces, denoted as E2+ and O1+ in BeGREEN architecture, to allow near-RT and non-RT management by the RIS Actuator. In the case of E2, new SMs are proposed to enable real-time control and monitoring of the smart surfaces. Regarding ISAC, possible options to process the radio signals at different levels (i.e., at the O-RU, O-DU, O-CU or near-RT RIC) are presented. Future BeGREEN deliverables will further elaborate the implications on the Intelligence Plane and RAN domain integration.

In the case of the Relays, it is proposed an integration based on O1 interfaces extensions, denoted as O1+ in the BeGREEN architecture, and a control component at the SMO which allows to apply non-RT optimisations. The proposed solution covers the full spectrum of relay-related procedures and their integration within the Intelligence Plance, starting with the detection of coverage holes, following with the identification of Relay UEs or placement for fixed relays, and concluding with the activation and deactivation of relays according to network conditions. In all the cases, specific AI/ML-based algorithmic solutions are presented and evaluated. Initial results, which are based on simulations according to a realistic scenario in a University Campus, characterize a specific coverage hole and show an energy consumption reduction in the range between 35%-70% depending on the BS and relay energy consumption model. Future work will consider addressing all the identified coverage holes with fixed relays and/or RUEs.

Regarding the Edge domain, and in a similar way to O-Cloud management, the BeGREEN architecture proposes interfaces and components between the SMO and the Edge controller which will allow to monitor

edge resources and apply dynamic resource allocation policies. Two AI/ML-assisted methods related to this area are proposed:

- Traffic-aware compute resource management to enhance UPF energy efficiency: Addresses the dynamic management of the edge compute resources allocated to UPF according to the variation of the traffic demand. The work presented in BeGREEN D4.1 is extended by incorporating a high-performance and more realistic, open-source implementation of the UPF using VPP and DPDK. The energy consumption characterization of this implementation reveals the necessity for energy-saving approaches, as it exhibits high CPU usage even in low-load scenarios. To this end, methods based on CPU frequency and thread scaling are evaluated. According to results in an experimental testbed and compared to an UPF configured in performance mode to lead with peak traffic demands, energy savings could reach 30%-45% depending on data load. Future work will integrate these methods with an ML-based decision-making process, utilizing UPF traffic forecasting to optimize resource management.
- Joint orchestration of vRANs and Edge AI services: Based on the experimental characterization presented in D4.1, it is analysed the intertwined relationships between Edge AI services performance, the resources allocated to vBSs and Edge AI services, and the energy consumption of the RAN and Edge domains. A Bayesian online learning algorithm is proposed to tackle this challenge, formulated as a contextual bandit problem. Future work will include the experimental evaluation of the proposed algorithm and its comparison with state-of-the-art ML approaches in terms of data efficiency and adaptability.

The Intelligence Plane also incorporates specific metrics to characterize the energy efficiency of the individual managed entities: the Energy Score and the Energy Rating. These metrics help to identify areas where energy consumption is highest and where optimizations can have the greatest impact. They not only aid in monitoring the effectiveness of energy-saving strategies but also enable dynamic adjustments through the proposed AI/ML-assisted control loops. The computation of these two metrics has been implemented as built-in serverless function in the Intelligence Plane through the Nuclio framework, and the exposure of the Energy Score function was demonstrated in the Intelligence Plane initial validation.

Additionally, BeGREEN also addresses the energy consumption and efficiency of the ML models themselves. In Chapter 3, a method is presented to reduce the dimensionality of the training data without compromising the accuracy of models, such as predictors. The method evaluates the importance of the model features and gradually discards them until accuracy is affected. An initial evaluation, using an energy consumption predictor, is presented, showing that this method can reduce CPU cycles during retraining by up to 85%. Finally, note that in this deliverable the energy consumption of specific models is also reported, particularly the ones related to the relay-enhanced control use case. Future work will extend this analysis to other relevant ML models being proposed in BeGREEN, evaluating the trade-off between the achieved energy saving gains and the required energy consumption to train and serve the models.

5 Bibliography

[1] BeGREEN D4.1, "State-of-the-Art Review and Initial Definition of BeGREEN O-RAN Intelligence Plane", December 2023, (Online) Available: https://www.sns-BeGREEN.com/deliverables

[2] ITU-R, "Future technology trends of terrestrial International Mobile Telecommunications systems towards 2030 and beyond", Report ITU-R M.2516-0, November 2022.

[3] O-RAN Alliance Working Group 1 (Use Cases and Overall Architecture), "O-RAN Network Energy Saving Use Cases Technical Report 2.0", O-RAN.WG1.Network-Energy-Savings-Technical-Report-R003-v02.00, June 2023.

[4] O-RAN Alliance Working Group 1 (Use Cases and Overall Architecture), "O-RAN Architecture Description", O-RAN.WG1.OAD-R003-v12.00, June 2024.

[5] BeGREEN D2.1, "BeGREEN Reference Architecture", July 2021, (Online) Available: https://www.sns-BeGREEN.com/deliverables

[6] MLRun, "Configuring runs and functions", https://docs.mlrun.org/en/latest/runtimes/configuring-job-resources.html#cpu-gpu-and-memory-requests-and-limits-for-user-jobs, Accessed June 2024

[7] MLRun, "Node affinity", https://docs.mlrun.org/en/v1.1.1/concepts/node-affinity.html, Accessed June 2024.

[8] NGMN Alliance, "NGMN 5G White paper", 5G Initiative Team, February 2015, Available online: https://ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf

[9] O-RAN Alliance Working Group 2 (Non-real-time RAN Intelligent Controller and A1 Interface Workgroup), "R1 interface: General Aspects and Principles", O-RAN.WG2.R1GAP-R003-v08.00, June 2024.

[10] O-RAN Software Community, "Non-RT RIC Information Coordination Service", https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric-plt-informationcoordinatorservice/en/latest/, Accessed June 2024

[11] O-RAN Alliance Working Group 2 (Non-real-time RAN Intelligent Controller and A1 Interface Workgroup), "O-RAN A1 interface: General Aspects and Principles", O-RAN.WG2.A1GAP-v03.03, June 2024.

[12] O-RAN Alliance Working Group 2 (Non-real-time RAN Intelligent Controller and A1 Interface Workgroup), "O-RAN A1 interface specification: Type Definitions 8.0", O-RAN.WG2.A1TD-v08.00, June 2024.

[13] E. Zeljković, "Proactive mobility management of software defined wireless networks", Doctoral dissertation, University of Antwerp, 2022.

[14] O-RAN Alliance Working Group 3 (Near-real-time RIC and E2 Interface Workgroup), "Near-RT RIC Architecture", O-RAN.WG3.RICARCH-R003-v05.00, October 2023.

[15] BeGREEN D3.2, "Initial Developments and Evaluation of the Proposed Enhancements and Optimization Strategies" July 2024 (Online) Available: https://www.sns-BeGREEN.com/deliverables

[16] O-RAN Alliance Working Group 4 (Open Fronthaul Interfaces WG), "Management Plane Specification", O-RAN.WG4.MP.0-R003-v15.00, June 2024.

[17] O-RAN Alliance Working Group 5 (Open F1/W1/E1/X2/Xn Interface Workgroup), "O-RAN O1 Interface specification for O-DU", O-RAN.WG5.O-DU-O1.0-R003-v09.00, February 2024.

[18] O-RAN Alliance Working Group 3 (Near-real-time RIC and E2 Interface Workgroup), "O-RAN E2 Service

Model (E2SM) Cell Configuration and Control", O-RAN.WG3.E2SM-CCC-R003-v04.00, June 2024.

[19] O-RAN Alliance Working Group 6 (Cloudification and Orchestration Workgroup), "O-RAN Study on O-Cloud Energy Savings", O-RAN.WG6.O-Cloud-ES-v01.00, June 2024.

[20] O-RAN Alliance Working Group 6 (Cloudification and Orchestration Workgroup), "O-RAN O2 Interface General Aspects and Principles", O-RAN.WG6.O2-GA&P-R003-v07.00, June 2024.

[21] BeGREEN, D2.2, "Evolved architecture and power enhancement mechanisms", July 2024, (Online) Available: https://www.sns-BeGREEN.com/deliverables.

[22] Marco Rossanese, et al., "Designing, building, and characterizing RF switch-based reconfigurable intelligent surfaces", in Proceedings of the 16th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization (WiNTECH '22), October 2022, https://doi.org/10.1145/3556564.3558236

[23] O-RAN Alliance Working Group 3 (Near-real-time RIC and E2 Interface Workgroup)," E2 Service Model (E2SM)", O-RAN.WG3.E2SM-R003-v05.00, February 2024.

[24] O-RAN Alliance Working Work Group 1 (Use Cases and Overall Architecture), "O-RAN Operations and Maintenance Interface 4.0", O-RAN.WG1.O1-Interface.0-v04.00, February 2021.

[25] 3GPP Rel-18, "Management and orchestration; 5G Performance Measurements", TS 28.552, June 2023.

[26] 3GPP Rel-15, "Management and orchestration; 5G end to end Key Performance Indicators (KPI)," TS_28.554, September 2018.

[27] 3GPP Rel-16, "Radio measurement collection for Minimization of Drive Tests (MDT)", TS 37.320, December 2021.

[28] O-RAN Alliance Working Group 6 (Cloudification and Orchestration Workgroup), "O-RAN Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN 7.0", O-RAN.WG6.CADS-v07.00, June 2024

[29] Open Network Foundation, "SMaRT-5G project: Sustainable Mobile and RAN Transformation (SMaRT)", White paper, June 2024.

[30] A. Kelkar and C. Dick, "NVIDIA Aerial GPU Hosted AI-on-5G", 2021 IEEE 4th 5G World Forum (5GWF), Montreal, Canada, October 2021, doi: 10.1109/5GWF52925.2021.00019.

[31] O-RAN Alliance Working Work Group 1 (Use Cases and Overall Architecture), "Use Cases Analysis Report", O-RAN.WG1.Use-Cases-Analysis-Report-R003-v14.00, June 2024.

[32] Mnih, Volodymyr, et al., "Playing atari with deep reinforcement learning." arXiv preprint, arXiv:1312.5602, 2013.

[33] Raposo, David, et al., "Discovering objects and their relations from entangled scene representations." International Conference on Learning Representations (ICLR 2017), 2017.

[34] Ayala-Romero, Jose A., et al. "vrAIn: A deep learning approach tailoring computing and radio resources in virtualized RANs." The 25th Annual International Conference on Mobile Computing and Networking. 2019.

[35] Ayala-Romero, Jose A., et al. "Orchestrating energy-efficient vRANs: Bayesian learning and experimental results." IEEE Transactions on Mobile Computing 22.5 (2021): 2910-2924.

[36] Ayala-Romero, Jose A., et al. "EdgeBOL: A Bayesian Learning Approach for the Joint Orchestration of vRANs and Mobile Edge AI." IEEE/ACM Transactions on Networking (2023).

[37] 3GPP, “Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation,” Link, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.211, 06 2022, version 17.2.0.

[38] Jaiantilal, Abhishek, Yifei Jiang, and Shivakant Mishra. "Modeling CPU energy consumption for energy efficient scheduling." Proceedings of the 1st Workshop on Green Computing, 2010.

[39] Shahid, Arsalan, et al. "Energy of computing on multicore CPUs: Predictive models and energy conservation law." arXiv preprint arXiv:1907.02805 (2019).

[40] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[41] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." arXiv preprint arXiv:1607.06450 (2016).

[42] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[43] Thimm, Georg, and Emile Fiesler. "High-order and multilayer perceptron initialization." IEEE Transactions on Neural Networks 8.2 (1997): 349-359.

[44] Tripathi, Sharda, et al. "Fair and scalable orchestration of network and compute resources for virtual edge services." IEEE Transactions on Mobile Computing (2023).

[45] T. Chen, C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). https://doi.org/10.1145/2939672.2939785

[46] 3GPP TR 38901, Study on channel model for frequencies from 0.5 to 100 GHz, (Release 18) v18.0.0 (2024-03).

[47] Olga Ruiz, Juan Sánchez-González, Jordi Pérez-Romero, Oriol Sallent, Irene Vilà, "Space and time user distribution measurements dataset in a university campus", Computer Networks, Volume 243, 2024, https://doi.org/10.1016/j.comnet.2024.110329.

[48] G. Auer et al., "How much energy is needed to run a wireless network?", in IEEE Wireless Communications, vol. 18, no. 5, pp. 40-49, October 2011, doi: 10.1109/MWC.2011.6056691.

[49] R. Fantini, D. Sabella, M. Caretti, “An E3F based assessment of energy efficiency of Relay Nodes in LTE-Advanced networks”, PIMRC, 2011.

[50] 3GPP TS 23.288, Architecture enhancements for 5G System (5GS) to support network data analytics services, (Release 18), v18.6.0, (2024-06

[51] 3GPP TS 26.531, Data Collection and Reporting; General Description and Architecture (Release 18), v18.1.0, (2023-12)

[52] B. Debaillie, C. Desset, and F. Louagie. A flexible and future-proof power model for cellular base stations. IEEE VTC, 2015.

[53] Arch Linux, “CPU frequency scaling”, https://wiki.archlinux.org/title/CPU_frequency_scaling, Accessed June 2024

[54] G. Lando, L. A. F. Schierholt, M. P. Milesi, and J. A. Wickboldt, “Evaluating the performance of open source software implementations of the 5g network core,” in NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium, 2023

[55] Ericsson Technology Review, “Energy-efficient packet processing in 5G mobile systems”, June 2022

[56] Shigeru Ishida, “Simple Measurement of UPF Performance”,

https://github.com/s5uishida/simple_measurement_of_upf_performance, Accessed June 2024

[57] D. Barach, L. Linguaglossa, D. Marion, P. Pfister, S. Pontarelli, and D. Rossi, "High-speed software data plane via vectorized packet processing," IEEE Communications Magazine, vol. 56, no. 12, pp. 97–103, 2018.

[58] FD.io, "PP/How To Optimize Performance (System Tuning)", https://wiki.fd.io/view/VPP/How_To_Optimize_Performance_(System_Tuning)#Hyperthreading, Accessed July 2024

[59] Intel, "Guidelines for optimizing power consumption of vctms deployments on intel xeon scalable processor" https://www.intel.com/content/www/us/en/content-details/778903/guidelines-for-optimizing-power-consumption-of-vcmts-deployments-on-intel-xeon-scalable-processors.html, Accessed July 2024.

[60] Everingham, Mark, et al. "The pascal visual object classes challenge: A retrospective." International journal of computer vision 111 (2015): 98-136.

[61] Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." Proceedings of the 19th international conference on World wide web. 2010.

[62] Filippi, Sarah, et al. "Parametric bandits: The generalized linear case." Advances in neural information processing systems 23 (2010).

[63] Williams, Christopher KI, and Carl Edward Rasmussen. Gaussian processes for machine learning. Vol. 2. No. 3. Cambridge, MA: MIT press, 2006.

[64] Duvenaud, David. Automatic model construction with Gaussian processes. Diss. 2014.

[65] Bull, Adam D. "Convergence rates of efficient global optimization algorithms." Journal of Machine Learning Research 12.10 (2011).

[66] Sui, Yanan, et al. "Stagewise safe bayesian optimization with gaussian processes." International conference on machine learning. PMLR, 2018.

[67] Berkenkamp, F., A. Krause, and Angela P. Schoellig. "Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics." arXiv." arXiv preprint arXiv:1602.04450 (2016).

[68] Sui, Yanan, et al. "Safe exploration for optimization with Gaussian processes." International conference on machine learning. PMLR, 2015.

[69] Fiducioso, Marcello, et al. "Safe contextual Bayesian optimization for sustainable room temperature PID control tuning." arXiv preprint arXiv:1906.12086 (2019).

[70] Krause, Andreas, and Cheng Ong. "Contextual gaussian process bandit optimization." Advances in neural information processing systems 24 (2011).