6G BLUR: The Blurring RAN

JOINT

Grant No. TSI-063000-2021-57

E5-E6: RAN platform and MANO stack software report

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA
Centre Tecnològic de Telecomunicacions de Catalunya

Abstract

This report will provide a brief overview of the contents related to Radio Access Network (RAN) and Management and Orchestration (MANO) software developments done in the framework of the 6GBLUR JOINT subproject included in the 6GBLUR public repository. This report is complemented with E4, where more details and evaluation results for the different project innovations are included.

Document properties

Document number	E5-E6
Document title	RAN platform and MANO stack software report
Document responsible	Jorge Baranda Hortigüela (CTTC)
Document editor	Jorge Baranda Hortigüela (CTTC)
Authors	All partners institutions: CTTC, Ericsson, Telefonica, SRS, Unican, Keysight
Target dissemination level	Public
Status of the document	Final
Version	1.0
Delivery date	31 December 2025
Actual delivery date	31 December 2025

Document history

Revision	Date	Issued by	Description
0.1	30/05/2025	Jorge Baranda Hortigüela (CTTC)	Initial ToC and initial content (KC and PoCs incorporating work of CTTC, Ericsson, SRS, TID, KEY, UniCan)
0.2	30/11/2025	Jorge Baranda Hortigüela (CTTC)	Inclusion final KC and PoCs contributions from CTTC
0.3	15/12/2025	Albert Bel (CTTC)	Review
1.0	17/12/2025	Jorge Baranda Hortigüela	Final Version

Disclaimer

This document has been produced in the context of the 6GBLUR Project. The research leading to these results has received funding from the Ministerio Español de Asuntos Económicos y Transformación Digital (MINECO), under grant TSI-063000-2021-56/-57.

All information in this document is provided “as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

For the avoidance of all doubts, the MINECO has no liability in respect of this document, which is merely representing the authors’ view.

Contents

List of Figures

List of Tables

Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA
cttc Centre Tecnològic de Telecomunicacions de Catalunya

List of Acronyms

AGV – Automatic Guided Vehicle

API – Application Programming Interface

CP – Control Plane

CU – Central Unit

DRL – Deep Reinforcement Learning

DSCP – Differentiated Services Code Point

DU – Distributed Unit

FH – Fronthaul

GCN – Graph Neural Networks

GJN – Generalized Jackson Network

IGP – Interior Gateway Protocol

KC – Key Concept

LDP – Label Distribution Protocol

MANO – Management and Orchestration

ML – Machine Learning

NPN – Non-Public Network

NRP – Network Resource Partition

NSC – Network Slice Controller

OSM – Open Source MANO

PoC – Proof of Concept

RAN – Radio Access Network

RIC – Radio Intelligent Controller

RT – Real Time

SDP – Service Demarcation Point

UC – Use Case

UP – User Plane

VNE – Virtual Network Embedding Problem

Executive Summary and Key Contributions

The Blurring RAN (6GBLUR) is a coordinated project funded by UNICO MINECO program. 6GBLUR consists of two subprojects, 6GBLUR-SMART (smart decision-making algorithms for efficient end-to-end resource management) and 6GBLUR-JOINT (joint RAN and transport network control/orchestration mechanisms).

As part of 6GBLUR, 6GBLUR-JOINT aims (i) to design the 6G disaggregated mobile network, from RAN to core, according to service-based approaches in a way that it can be treated as any other virtual service that requires some specific hardware support, and (ii) to design and to validate of joint RAN and transport orchestration mechanisms that adapt to the needs of the disaggregated 6G network.

This document provides a brief overview of the software components and assets related to RAN and MANO aspects developed within the context of the 6GBLUR JOINT subproject. This content has been included in a repository and published as open source with installation notes to support the installation of the components in third party environments and the re-use of its components in future research activities. Furthermore, the developments described in this document resulted in contributions to open-source initiatives, like srsRAN project, 5G-LENA or Open-TAP. In such cases, the 6GBLUR repository contains descriptions and pointers to the main open-source projects where the contributions of 6GBLUR have been integrated.

Following the approach of E3 and E4 documents, the following description is organized into Key Concepts (KC) and related Proof of concepts (PoC) of the defined Use cases (UCs).

1 6GBLUR Repository

The outcomes of the development of the 6GBLUR proposed architecture, studied innovations through the Key concepts (KC) and its integration and validation in the Proof of concepts (PoC) of the different Use Cases (UCs) have been included into a public gitlab repository hosted by CTTC organization. Figure 1 presents the frontpage of this repository, which can be found at the following URL: https://gitlab.cttc.es/mdi-6gblur [1].

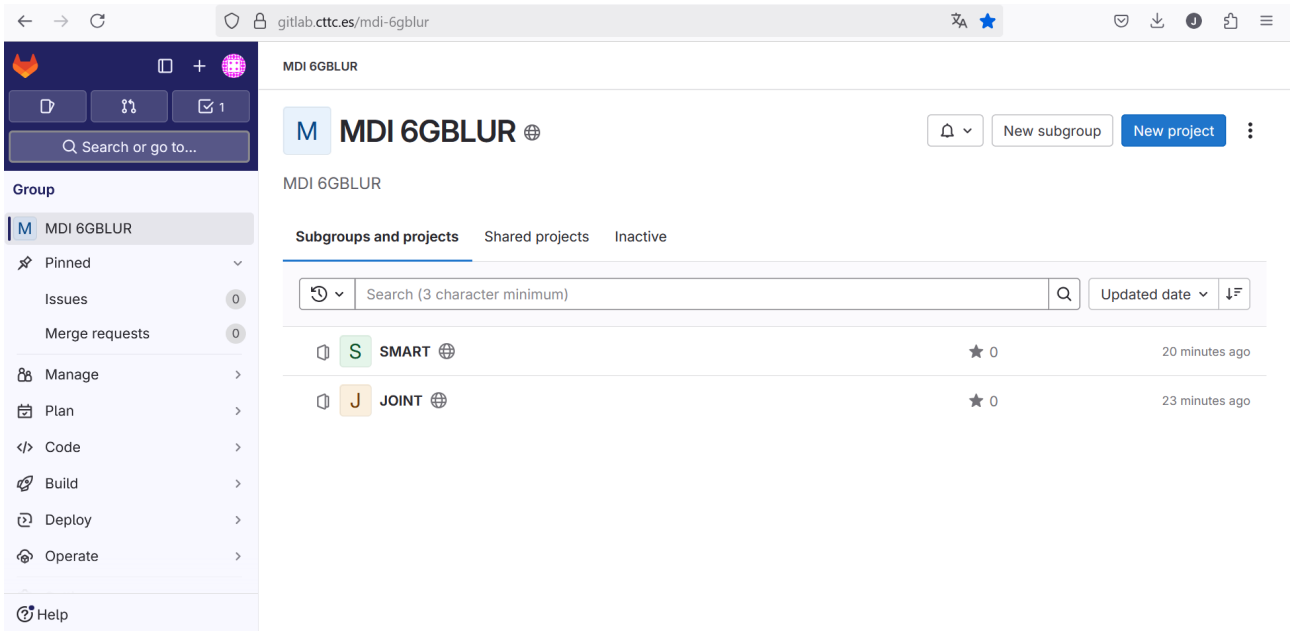


FIGURE 1. 6GBLUR GITLAB REPOSITORY FRONTPAGE

This repository is divided into 6GBLUR subprojects: SMART and JOINT and for each subproject, the outcomes are grouped among KCs and PoC, following the same organization of technical deliverables, like E3 [2] and E4 [3], as depicted in Figure 2.

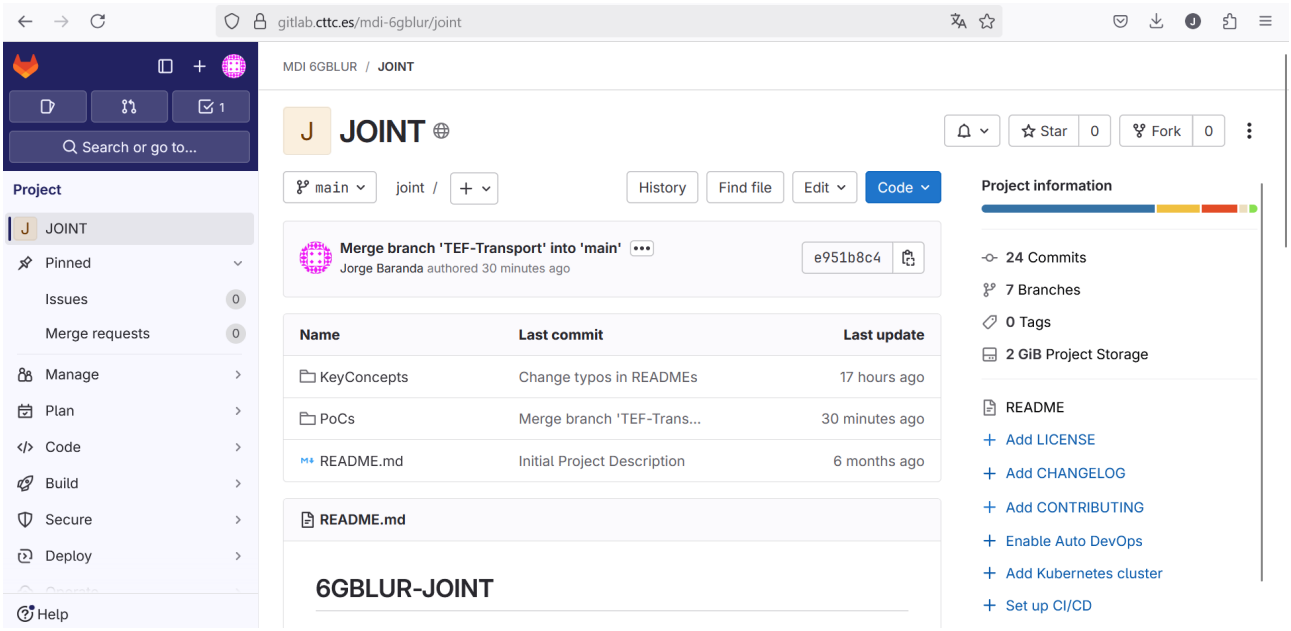


FIGURE 2. 6GBLUR JOINT GITLAB REPOSITORY FRONTPAGE

Indeed, this brief report gives an overview of such outcomes, installation notes and useful links to support the installation of the components in third party environments and the re-use of its components in ongoing and future research activities. The content in the repository includes different kind of material related with Radio Access Network (RAN) and Management and Orchestration (MANO) activities such as software simulation code, artefacts for the automated MANO of end-to-end (E2E) mobile network entities in cloud-native environments, videos explaining the different PoCs, datasets to train AI/ML models, description of the enhancements provided to Open Source Projects (i.e., like 5G-LENA [4], srsRAN Project [5] and OpenTAP [6]), some extracted results validating software enhancements to open-source projects. Table 1 and Table 2 present a classification of the KCs and PoCs developed in JOINT subproject based on the aspect they tackle, either RAN or MANO. Of course, due to the integration of KC into PoCs, we consider PoCs as MANO and RAN outcomes as they provide an end-to-end view. Full detailed explanations of algorithms and results can be found in the final deliverable E4 [3].

TABLE 1. MAPPING OF JOINT KEY CONCEPTS INTO RAN AND MANO ASPECTS

Key Concept	RAN (E5)	MANO (E6)
K1.1: Data plane reconfiguration and Zero-Touch		X
K1.2: Distributed deployment of e2e slices	X	X
K1.3: Non-Public Network		X

K1.4: Network Digital Twin		X
K2.1: Transport network optimization		X
K2.2: Entities' placement decisions	X	
K2.3: QoS policies and scheduling	X	
K3.1: Monitoring Platform	X	
K3.2: AI/ML Platform	X	

TABLE 2. MAPPING OF JOINT UCPOCS INTO RAN AND MANO ASPECTS

UCPoC	RAN	MANO
UC1PoC1. Zero-Touch distributed 3GPPP network for delivering Non-Public Network	X	X
UC1PoC2. Distributed O-RAN based mobile network	X	X
UC2PoC1. Joint Fronthaul Capacity Control/Placement, Qos management and Flexible Functional Splits	X	X
UC2PoC2. NPN X-validation/optimization with Digital Twin	X	X

2 JOINT Key Concepts

The following subsections present the description and overview of the contents uploaded to the corresponding KC space in the 6GBLUR JOINT repository.

2.1 JOINT-K1.1: Data plane reconfiguration and Zero-Touch

This part of the repo contains the artifacts corresponding to the different versions of the Open5Gs software used during 6GBLUR for the development of the JOINT KC1.1.

- Version v2.6.4 was used to develop a disaggregated deployment of the Control Plane (CP) and User Plane (UP) of the 5G mobile core. In this way, if properly configured the core, UP network functions can be activated on-demand.
- Version v2.7.1 is an evolution of the previous one decoupling the SMF function of the CP and with the possibility to configure the number of slices. Then, slice network functions (i.e., SMF and UPF) can be deployed on-demand when needed.

This folder contains three additional folders: i) dockerfiles, ii) Helms charts allowing the appropriate parametrization and deployment using Kubernetes, and iii) OSM descriptors (validated with OSM R13 and R14 version), allowing the deployment automation through OSM. An additional parameter file is added with OSM descriptor, as an example of configuration that can be passed during instantiation time to modify the values present in the corresponding HelmChart.

Please refer to K1.1 Data Plane Reconfiguration and Zero-Touch key concept section in the E4 deliverable for further details about the implementation of these artifacts.

2.2 JOINT-K1.2: Distributed deployment of e2e slices, including O-RAN

This part of the repo contains the resulting artifacts corresponding to the RAN and Near-Real Time (RT) RIC Controller used during 6GBLUR project for the development of the JOINT KC1.2. More specifically, RAN artifacts are based on open-source software provided by project partner SRS and Near-RT RIC on FlexRIC open-source software provided by Mosaic5G.

In every folder, we find i) dockerfiles, ii) Helms charts allowing the appropriate parametrization and deployment using Kubernetes, and iii) OSM descriptors (tested for OSM version R13 and R14), allowing the deployment automation through OSM.

Among RAN artifacts, we find artifacts for the complete gNB, the separated CU and DU (including UHD version for using with Split 8 and USRP as RU, and OFH version for using with Split 7.2 and LiteOn as RU in conjunction with Falcon Switch) and a PTP artefact to provide synchronization in the network interface of the cluster running the DU. Regarding the nearRT-RIC artifacts, we find them

for the controller and for a sample monitoring function allowing a variable list of monitoring parameters.

Please refer to K1.2 Distributed deployment of e2e slices, O-RAN key concept section in the E4 deliverable for further details about the implementation of these artifacts.

2.3 JOINT-K1.3: Non-public-network

This part of the repo contains a video file, named "ZERO TOUCH demo with CTTC and NPN Pelican.mp4", which is related to the key concept JOINT-K1.3 Non Public network to showcase how Non Public Network can be deployed and make it available for service using Zero touch mechanism. Please refer to JOINT-K1.3 Non Public network key concept section in the E4 deliverable for details about this key concept. The video was recorded while Ericsson was demonstrating this key concept to CTTC. The same was also demonstrated during the Face to Face November 2024 plenary session to partners.

2.4 JOINT-K1.4: Network Digital Twin

This part of the repo contains Postman collections and the Network Digital Twin Platform exposure API's YAML file resources. These resources showcase the APIs provided by the Network Digital Twin Platform and demonstrate how Application Functions, Communication Service Providers, or enterprise administrators can utilize the rich set of ML-based exposure APIs at different stages of the innovation cycle for connectivity use cases. The files included in the repository for this key concept are:

- **CTTC 5tonic-exposure.yaml** file is the Network Digital Twin Platform ML-based exposure open APIs specification document. Please refer to JOINT-K1.4 Network Digital Twin key concept section in the E4 deliverable for more details about the API description and example usages.
- **5G AGVs USE-CASE- DESIGN PHASE.postman_collection.json** file contains the Postman collection for the Design Phase API usage for the AGV real-world use case. Please refer to UC2PoC2: NPN X-validation/optimization with Digital Twin and UC3/PoC1: Data-Driven RAN optimization for NPN deployments sections in the E4 deliverable for more information about AGV real-world use cases and usage examples.
- **5G AGVs USE-CASE- PRE-PRODUCTION PHASE.postman_collection.json** file contains the Postman collection for the Pre-Production Phase API usage for the AGV real-world use case. Please refer to UC2PoC2: NPN X-validation/optimization with Digital Twin and UC3/PoC1: Data-Driven RAN optimization for NPN deployments sections in the E4 deliverable for more information about AGV real-world use cases and usage examples.

- **5G_AGVs_USE-CASE-_PRODUCTION_PHASE.postman_collection.json** file contains the Postman collection for the Production Phase API usage for the AGV real-world use case. Please refer to UC2PoC2: NPN X-validation/optimization with Digital Twin and UC3/PoC1: Data-Driven RAN optimization for NPN deployments sections in the E4 deliverable for more information about AGV real-world use cases and usage examples.
- **5G_AGVs_USE-CASE-_OPTIMIZATION_PHASE.postman_collection.json** file contains the Postman collection for the Optimization Phase API usage for the AGV real-world use case. Please refer to UC2PoC2: NPN X-validation/optimization with Digital Twin and UC3/PoC1: Data-Driven RAN optimization for NPN deployments sections in the E4 deliverable for more information about AGV real-world use cases and usage examples.

2.5 JOINT-K2.1: Transport network optimization

This key concept focuses on optimizing the transport network within an ORAN-based mobile network. The goal is to enable network slicing in the transport by developing a specialized controller. Currently, the industry lacks a standardized approach for defining a common Transport Network Slice Controller (NSC) component capable of facilitating the provision of connectivity services in the form of slices. This component would manage slice requests and the associated procedures.

The Network Slice Controller (NSC) is a component defined by the IETF to orchestrate the request, realization, and lifecycle control of network slices. It consists of two main modules: the mapper and the realizer.

2.5.1 Overview

The NSC handles end-to-end network slice requests originating from 5G customers. These requests are managed by the 5G end-to-end orchestrator, which configures RAN and Core Network elements accordingly and passes the request to the NSC for processing. The NSC then interacts with relevant network controllers to implement the network slice into the transport network.

2.5.2 Main Modules

Mapper

The mapper processes client network slice requests and correlates them with existing slices. When a slice request arrives, the mapper translates it by converting the request expressed in 3GPP NRM terms into the IETF NBI data model. This involves identifying the service demarcation points (SDPs) that define the connectivity in the transport network. Once these parameters are identified and mapped into the data model, the next step is to check the feasibility of implementing the slice request.

Realizing a slice requires an existing network resource partition (NRP) with the specified slice requirements, which may not be available at the time of the request. This information will be retrieved from an external module, which is beyond the scope of this definition. This module will provide a response regarding the feasibility of realizing the slice.

If there are no available NRPs for instantiating the slice, the mapper will request the realizer to create a new NRP. This involves interacting with the network controllers responsible for the transport network handled by the NSC. This process is iterative until the mapper determines that the slice realization is feasible. In the current version, it is assumed that there is only one available NRP corresponding to the entire network, and that it is always accessible to the user.

Realizer

The realizer module determines the realization of each slice by interacting with specific network controllers. This version is currently working with Teraflow SDN controller. It receives requests from the mapper and decides on the technologies to be used to instantiate the slice based on the selected NRP associated with the slice. For example, Layer 2 VPN is the technology employed to realize network slices in this version. To achieve this, the realizer generates a request for the network controller to establish a Layer 2 VPN between two SDPs with the requirements specified in the slice request.

2.5.3 Workflow

1. **Request Initiation**: Network slice request originates from a 5G customer and is managed by the 5G end-to-end orchestrator.
2. **Mapper Processing**: Converts the request into the IETF NBI data model, identifies SDPs, and checks feasibility.
3. **Realizer Action**: Determines technology (e.g., Layer 2 VPN) and interacts with network controllers to instantiate the slice.
4. **Implementation**: Network controllers configure the transport network as per the slice requirements.

2.5.4 Architecture

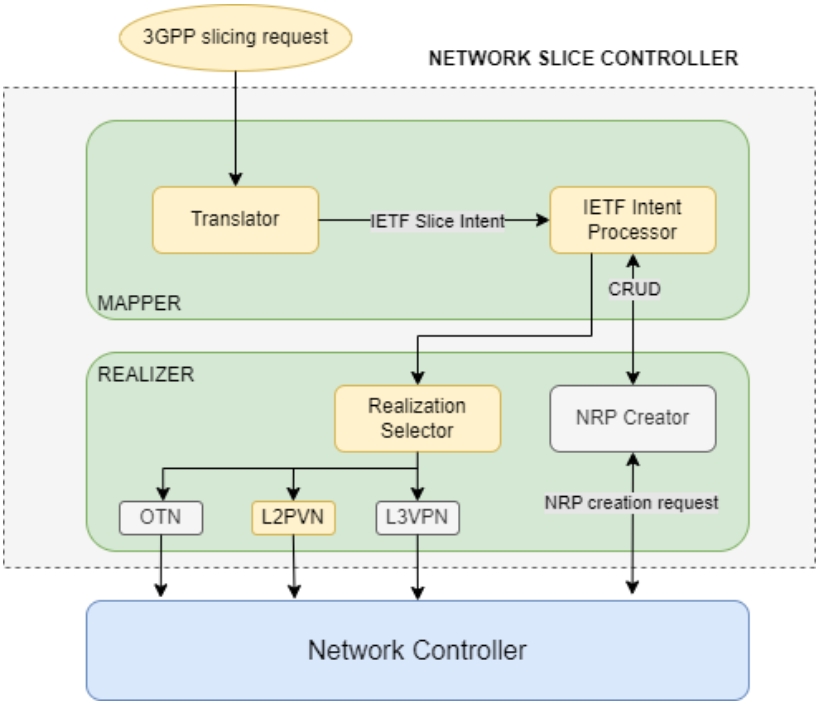


FIGURE 3. NSC ARCHITECTURE

2.5.5 Requirements

- Python3
- python3-pip
- python3-venv

2.5.6 Configuration Constants

In the main configuration file, several constants can be adjusted to customize the Network Slice Controller (NSC) behaviour:

Logging

- DEFAULT_LOGGING_LEVEL: Sets logging verbosity
  - Default: logging.INFO
  - Options: logging.DEBUG, logging.INFO, logging.WARNING, logging.ERROR

Server

- NSC_PORT: Server port
    - Default: 8081

Paths

- SRC_PATH: Absolute path to source directory
- TEMPLATES_PATH: Path to templates directory

Teraflow Configuration

- TFS_UPLOAD: Enable/disable uploading slice service to Teraflow
    - Default: False
- TFS_IP: Teraflow SDN controller IP
    - Default: "192.168.165.10"
- TFS_L2VPN_SUPPORT: Enable additional L2VPN configuration support
    - Default: False

This implementation assumes that a TeraflowSDN controller is deployed in the configured IP. In order to deploy it, you can follow the official deployment guide.

For operating over a topology, Cisco xrv9000 routers are required with the following pre-configurations:

- An Interior Gateway Protocol (IGP) activated. In this case, OSPF, for routing traffic through the routers.
- MPLS Label Distribution Protocol (LDP), required for enabling an MPLS-TE topology.
- RSVP activated in routers port interfaces.
- MPLS-TE topology, required for creating traffic engineering tunnels in the topology.
- Traffic engineering tunnels, through the possible paths for traffic between endpoints.

There are examples of topologies to onboard them on Teraflow in the files starting with "descriptor" in the src/templates folder.

2.5.7 Usage

To deploy and execute the NSC, follow these steps:

Preparation

1. git clone https://github.com/Telefonica/network_slice_controller.git
2. cd network_slice_controller

3. python3 -m venv venv
4. source venv/bin/actívate
5. pip install -r requirements.txt

Start NSC Server:

python3 app.py

Generate Slice Requests:

To send slice request, the NSC accepts POST request at the endpoint /slice. It is available in the swagger documentation panel at {ip}:{NSC_PORT}/nsc.

You can find templates ready for use with examples of 3GPP slice requests with different requirements in the folder **src/templates/6GBlur**. More detail about these templates can be found in KC JOINT 2.1 and UC1PoC 2 sections in E4 deliverable.

2.6 JOINT-K2.2: Entities' placement decisions

This key concept aims to optimize the placement of virtualized entities that belong to the disaggregated RAN into a shared physical infrastructure. Considering the implications of placement in terms of CAPEX/OPEX multiple placement policies can be adopted according to the specific preferences. This folder contains AI/ML solution based on Graph Neural Networks (GCN) and Deep Reinforcement Learning (DRL) that provides a solution to the underlying Virtual Network Embedding Problem (VNE). The AI/ML agent is configured to optimize the revenue to cost ratio. Nevertheless, the implemented framework can be adapted to multitude of policies. The implementation is based in the project FLAG-VNE.

2.6.1 Installation

The installation and execution have been tested in a machine with OS Ubuntu 24.04.2 LTS with Python version 3.12.3.

- Clone the repository move to the corresponding folder

```
git clone git@gitlab.cttc.es:mdi-6gblur/joint.git
cd KeyConcepts/JOINT-K2.2 Entities placement decisions
```

- Generate and activate a virtual Python environement

```
python3 -m venv myenv
source myenv/bin/activate
```

After activating the environment, the prompt should be preceded by *(myenv)*.

- Install the required packages

```
bash install.sh
```

2.6.2 Folder Structure

```
.
├── figs # folder where results figures are stored
├── save # folder with results
├── settings
    ├── p_net_waxman.yaml # random physical network
    ├── v_net_gnb.yaml # generator for virtual networks, each being a
base station
├── vne_simulator # simulator folder from FLAG_VNE
├── install.sh # Installation file
├── main_blur_joint_k22.py # experiments execution file
└── plot_blur_joint_k22.ypnb # plotting file
```

The *settings* folder contains the description of the physical and virtual networks as follows:

- *p_net_waxman.yaml*: physical nodes are characterized by their CPU capacity and number of antennas, which are used to allocate RUs; it is worth noting that only some physical nodes are configured to allocate RUs. Physical links are characterized by their bandwidth (communication capacity).
- *v_net_gnb.yaml*: each virtual network consists of 3 virtual nodes: RU, DU, CU. The first node, that represents the RU, only requires 1 antenna and no CPU. The other two nodes (DU and CU) require no antenna and a configurable amount of CPU. Virtual links are configured with bandwidth requirement.

The *save* folder already contains the results obtained for the tool validation: *~/save/a3c_gcn/w_d_[min]_[max]*, where *min* and *max* correspond to the minimum and maximum value of the CPU of the physical nodes. Each result folder contains:

- *log* folder with the log output of tensorboard.
- *model* folder with the model description in a pickle file.
- *record* folder with the output of the training and execution in .csv files.

- *config.yaml* with the concrete configuration with which the results were obtained.

2.6.3 Execution

The project is ready to execute the tool as follows:

```
python3 main_blur_joint_k22.py
```

The execution of the tool with generate a new folder into *save/a3c_gcn* as well as a *dataset* folder that contains the specific networks (physical and virtual) that have been generated in *.gml* files.

The file *plot_blur_joint_k22.ipynb* generates plots from the existing results (*~/save/a3c_gcn/w_d_[min]_[max]*). The file can be modified to generate similar plots from newly generated results.

2.7 JOINT-K2.3: QoS policies and scheduling

This repository contains the resulting artifacts from network simulation experiments. It includes scripts to automate the execution of ns-3 simulation scenarios, Jupyter notebooks for analysing the results and comparing them with analytical models, and the simulation scenarios themselves, implemented directly in ns-3. These resources were used during the 6G-BLUR project for the development of the JOINT KC2.3. This repository hosts the ns-3 implementation with particular focus on Differentiated Services Code Point (DSCP). The modelling enables performance evaluations of network slices with varying priority levels, as well as the convergence of technologies including backhaul and fronthaul.

2.7.1 ns-3 Environment

Scripts for simulation automation, post-simulation analysis, and data processing as well as ns-3 scenarios can be found in the following folders:

Folder	Description
Jackson-nets	Analysis using traditional Jackson network models assuming a stochastic model.
OpenGJN	Tools for Open Generalized Jackson Network used to analyse fronthaul performance.
UC2POC1	Code related to the UC2POC1 analysis approach.
Sched-analysis	Scheduling analysis in a bottleneck scenario.
HL3HL5nalysis	Centralization analysis between HL3 and HL5, considering different levels of aggregation: (i) O-DU allocated at HL3 managing a pool of 23 sites, (ii) O-DU

	allocated at HL4 handling a pool of 9 sites. Different centralization distances were evaluated based on Telefónica de España's IP/Fusión transport network architecture.

2.7.2 Getting Started

Run the following commands to build the simulator:

```
mkdir build
cd build
cmake ..
cd ..
CXXFLAGS="-W -Wall -g" ./ns3 configure --disable-werror --disable-python
--disable-tests --disable-examples --build-profile=release
./ns3 build
```

2.7.3 Key Modifications

All relevant modifications are made in:

- Traffic-Control The Traffic Control layer sits in between the NetDevices (L2) and any network protocol (e.g. IP). It is in charge of processing packets and performing actions on them: scheduling, dropping, marking, policing, etc.

2.7.4 Documentation

- ns-3 Manual
- ns-3 Model-Library

2.7.5 Additional Artifacts

In addition, the ofh-traffic-pcapanalysis folder contains several scripts for analyzing metrics from real Open Fronthaul (OFH) traffic captures.

Moreover, qnet1.1 includes a solver for obtaining performance metrics by modeling the system as a Generalized Jackson Network (GJN).

2.8 JOINT-K3.1: Monitoring platform

This repository contains artifacts to generate an Agentic monitoring platform.

Please refer to the project and simple-case folders to obtain more information on the different components developed, the architecture and the interactions between. Inside them you can find additional README files with detailed instructions on how to setup and run the agentic monitoring platform.

Please refer to the K3.1 Agentic Monitoring Platform key concept section in the E4 deliverable for further details about the implementation of these artifacts.

2.9 JOINT-K3.2: AI/ML Platform (AIMLP)

This dataset belongs to both 6GBLUR project under the key concept called K3.2 AI/ML Platform. You can refer to K3.2 AI/ML platform-related concepts in the E4 deliverable. This key concept is common for 6GBLUR SMART and 6GBLUR JOINT projects.

The AGV_Usecase_dataset.csv file contains a list of AGV use case experiments (refer to the UC2PoC2: NPN X-validation/optimization with Digital Twin and UC3/PoC1: Data-Driven RAN optimization for NPN deployments sections in the E4 deliverable) along with their execution details and related datasets.

Please refer to the following details to associate the datasets with a clear description:

- Column Id: Unique Experiment ID Column
- start_time: Execution start time of the experiment Column
- stop_time: Execution end time of the experiment
- Column duration: Experiment execution duration
- Column configuration_cell_name: RAN cell name
- Column configuration_bandwidth: RAN bandwidth configuration
- Column configuration_power: RAN transmission power configuration
- Column configuration_antenna: Antenna serial name
- Column configuration_tdd_pattern: TDD pattern configuration value
- Column energy: RAN energy consumption in Watts/hour for every 6 seconds
- Column energy_total: Total RAN energy consumption in Watts/hour for the experiment duration
- Column owd: One-way delay/latency measured in milliseconds for the experiment duration
- Column pocket_loss_n6: Number of packet losses measured at the N6 interface
- Column pocket_loss_ue: Number of packet losses measured at the UE probe interface
- Column predicted_kpis: Network digital twin predicted KPIs for the experiment

- Column traffic_definition: Traffic model details used for the experiment
- Column traffic_model: Traffic model name used for the experiment
- Column traffic_received: Traffic received, measured in Mbps, for the experiment
- Column traffic_sent: Traffic sent, measured in Mbps, for the experiment

3 JOINT UC PoCs

The following subsections present the description and overview of the contents uploaded to the corresponding UC PoCs space in the 6GBLUR JOINT repository.

3.1 UC1: 6G Disaggregated Zero-Touch Mobile Network as a Service

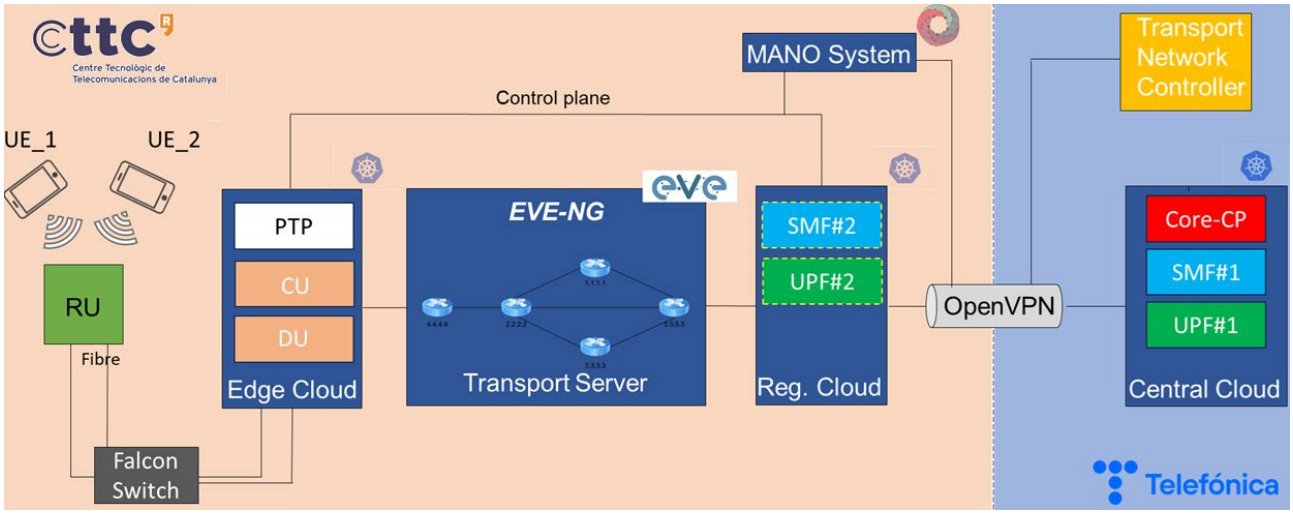
3.1.1 PoC1: Zero-Touch distributed 3GPP network for delivering Non-public Networks

This part of the repository contains a video file, named "6G-BLUR F2F Meeting Nov. 21-22-20241121_113701-UC1PoC1.mp4", which is related to UC1/PoC1: Zero-Touch Distributed 3GPP Network for Delivering Non-Public Networks.

The implementation pathway of this PoC has focused on the practical implementation of the vision concept for Zero-Touch mechanism in NPN. Please refer to UC1/PoC1: Zero-Touch Distributed 3GPP Network for Delivering Non-Public Networks Section in the E4 deliverable for details about this PoC. The video was recorded during Face to Face November 2024 plenary meeting where Ericsson presented this PoC to partners.

3.1.2 PoC2: Distributed O-RAN based mobile network

This Proof of Concept (PoC) demonstrates the deployment of a flexible and distributed end-to-end cloud-native mobile network based on the O-RAN architecture. Specifically, the PoC involves the deployment of a complete mobile network, including the configuration of network slices to provide differentiated connectivity services to users.



This work builds upon several of the key concepts described previously. More specifically, the involved key concepts are:

- SMART-K1.1 – CU and DU Implementation: Implementation of O-RAN Central Unit (CU) and Distributed Unit (DU) components using the srsRAN software.
- JOINT-K1.1 – Data Plane Reconfiguration and Zero-Touch: Distributed and Disaggregated cloud-native deployment of mobile core network entities (control plane and data plane).
- JOINT-K1.2 – Distributed Deployment of End-to-End Slices, Including O-RAN: Cloud-native deployment of all RAN and Core network elements.
- JOINT-K2.1 – Transport Network Optimization: Deployment and optimization of the transport network to support the mobile infrastructure.

Please refer to the UC1PoC 2 section in the E4 deliverable for further details about the implementation of these PoC. The software developed for the execution of this PoC is available in: Orchestrator GUI and NSC. You can also find a PoC demo in [7]

3.2 UC2: Joint RAN and Transport mechanisms for 6G Disaggregated Mobile networks

3.2.1 PoC1: Joint Fronthaul Capacity Control/Placement, QoS management and Flexible Functional Splits

This proof of concept explores and evaluates the various possibilities that emerge when implementing different distributed approaches within a mobile network infrastructure, specifically incorporating O-RAN components in the RAN domain.

3.2.1.1 FH characterization

An essential task undertaken was the creation of a MATLAB tool capable of calculating the most suitable O-DU timing based on predefined O-RU timing and specific fronthaul conditions. The files can be seen in SMART KC2.2.

FH characterization presents the results obtained through the use of the tool, including both numerical data and graphical visualizations of the extended centralization scenarios. It highlights the key configuration parameters involved, from DU and FH, as well as the specific timing adjustments performed by the software to maximize the achievable centralization distance under varying conditions.

3.2.1.2 QoS Policies and Scheduling

JOINT KC2.3 contains the resulting artifacts from network simulation experiments. It includes scripts to automate the execution of ns-3 simulation scenarios, Jupyter notebooks for analyzing the results and comparing them with analytical models, and the simulation scenarios themselves, implemented directly in ns-3.

This repository hosts the ns-3 implementation with particular focus on Differentiated Services Code Point (DSCP). The modelling enables performance evaluations of network slices with varying priority levels, as well as the convergence of technologies including backhaul and fronthaul.

3.2.1.3 FH experimental development

The new features of the srsRAN O-FH library (SIMD-based compression and DPDK, see SMART-K2.1, were driven by the PoC requirements at the fronthaul level.

3.2.1.4 FH control methods

Various FH control methods have been developed to control the downlink data bulks that go through a limited-capacity fronthaul link.

They have been implemented in ns-3 5G-LENA and evaluated through system-level simulations using mixed AR/VR/CG scenarios. The new features of this work are included in QoS schedulers FH control and flexible FS.

3.2.1.5 OpenTapPlugin

As part of the experimental activities, an OpenTAP plugin was developed to enable native integration of the srsRAN software with the automated testing framework, streamlining the execution and collection of experimental results. Additional information regarding the experimental setup and procedures can be found in the subsection Experimental Implementation and Assessment of UC2PoC1 in E4 document.

OpenTapPlugin is designed for native integration of srsRAN software used in the testbed. More information can be found in OpenTapPluginForAutomatedTesting.

3.2.2 PoC2: NPN X-validation/optimization with Digital Twin

This part of the repository contains a video file, named "UC2PoC2 NPN Optimization with NDT.mp4", which is related to UC2PoC2: NPN X-validation/optimization with Digital Twin.

The Proof of Concept for NPN X-validation/optimization using a Digital Twin illustrates the potential of using a Network Digital Twin platform for automated self-configuration and optimization in non-public networks. Please refer to UC2PoC2: NPN X-validation/optimization with Digital Twin Section in the E4 deliverable for details about this PoC.

This PoC is also shared with 6GBLUR-SMART subproject, you can refer this 6GBLUR-SMART project repository path - https://gitlab.cttc.es/mdi-6gblur/smart. The video was recorded during Face to Face November 2024 plenary meeting where Ericsson presented this PoC to partners.

4 Conclusions

Beyond technical deliverables E3 and E4, 6GBLUR has generated a set of additional outcomes, which have been made public through the 6GBLUR repository [1]. This repository is divided into the two 6GBLUR subprojects: SMART and JOINT. This document describes the outputs corresponding to the JOINT subproject.

The content in the repository is further organized into KC and PoCs, including the different kind of material related with Radio Access Network (RAN) and Management and Orchestration (MANO) activities such as software simulation code, artefacts for the automated MANO of E2E mobile network entities in cloud-native environments, videos explaining the different PoCs, datasets used to train AI/ML models, description of the enhancements provided to Open Source Projects (i.e., like 5G-LENA [4], srsRAN Project [5] and OpenTAP [6]), and some extracted results validating software enhancements to open-source projects.

The provided installation notes and links included in some of the KC and PoCs, and the available README files support the installation of the components in third party environments and promote its re-use in ongoing and future research activities.

5 References

[1] 6GBLUR Gitlab repository, [Online] Available at: https://gitlab.cttc.es/mdi-6gblur

[2] 6GBLUR E3 Deliverable, “Final RAN architecture design”, June 2024

[3] 6GBLUR E4 Deliverable, “Mechanisms and results report”, December 2025

[4] 5G-LENA, “The 5GNR module for the ns-3 simulator”, [Online] Available online at: https://5g-lena.cttc.es/

[5] srsRAN Project, “Open Source RAN”. [Online]. Available at: https://github.com/srsran

[6] OpenTap, “An Open Source Project for Test Automation”. [Online]. Available at: https://opentap.io/

[7] J. Baranda, J. Velázquez Martínez, A. Bel, D. Gregoratti, L. M. Contreras, J. Mangues-Bafalluy, “Full Orchestration of a Distributed 5G cloud-native Mobile Network: O-RAN, Core, and Transport”, in IEEE Network Softwarisation Conference (IEEE Netsoft’25), 23-27 June, Budapest, Hungary.