6G BLUR

6G BLUR: The Blurring RAN

SMART

Grant No. TSI-063000-2021-56

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 SMART 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, 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
1.0	30/05/2025	Jorge Baranda Hortigüela (CTTC)	Initial ToC and initial content (KC and PoCs incorporating work of CTTC, Ericsson, SRS, TID, KEY)
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 6G BLUR 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

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 Figures

List of Tables

List of Acronyms

AGV – Automated Guided Vehicle

CG – Cloud Gaming

CU – Centralised Unit

DL – Downlink

DPDK – Data Plane Development Kit

DU – Distributed Unit

E2E – End-to-End

FFS – Flexible Functional Split

FFT – Fast Fourier Transformation

FH – Fronthaul

KC – Key Concept

KPM – Key Performance Measurements

LLM – Large Language Model

LSTM – Long Short-Term Memory

MAC – Medium Access Control

MANO – Management and Orchestration

MPSoC – Multiprocessor System-on-Chip

NIC – Network Interface Card

NPN – Non-Public Network

PL – Programmable Logic

PoC – Proof of Concept

PPO – Proximal Policy Optimization

PRACH – Physical Random-Access Channel

PRB – Physical Resource Block

RAN – Radio Access Network

RC – RAN Control

RFSoC – Radio Frequency System-on-Chop

RIC – Radio Intelligent Controller

RU – Radio Unit

SIMD – Single Instruction Multiple Data

TW – Transmission Window

UC – Use Case

UL – Uplink

XR – Extended Reality

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-SMART aims (i) to design and to validate a RAN control architecture adapting to all the required decision timescales, from policy definition to scheduling, and (ii) to design and validate smart algorithms for efficient end-to-end resource management, including decision-making at all timescales, with real-time and non-real time decisions.

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 SMART 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].

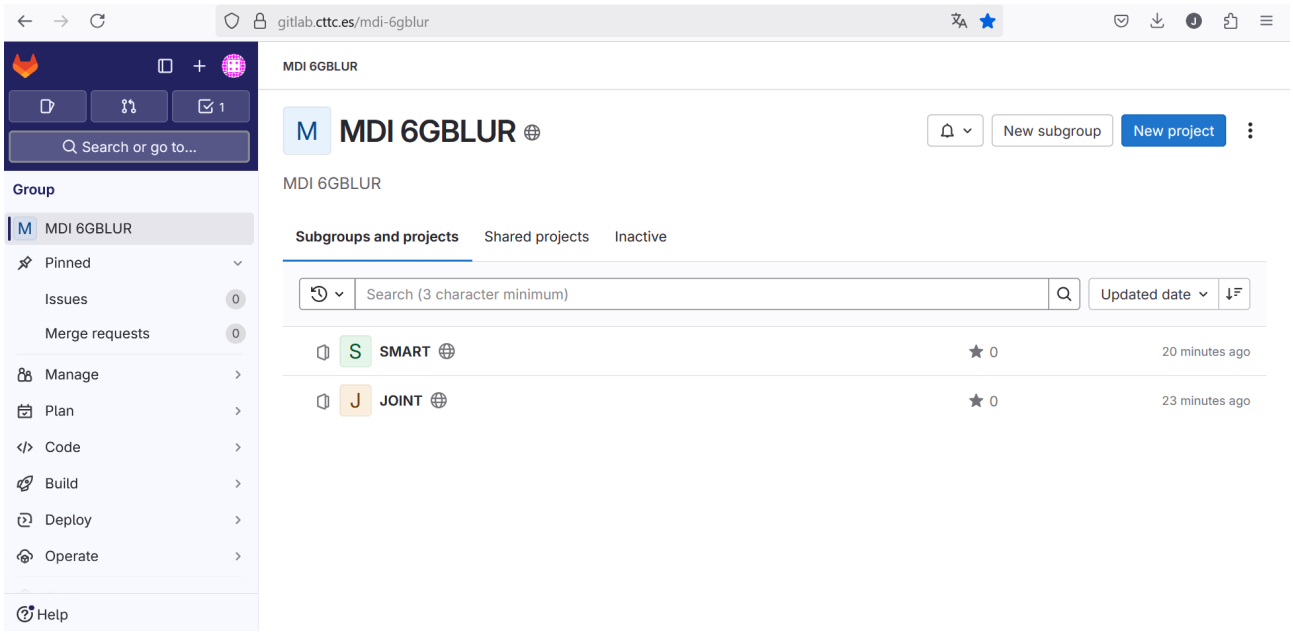

FIGURE 1. 6GBLUR GITLAB REPOSITORY FRONTPAGE

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.

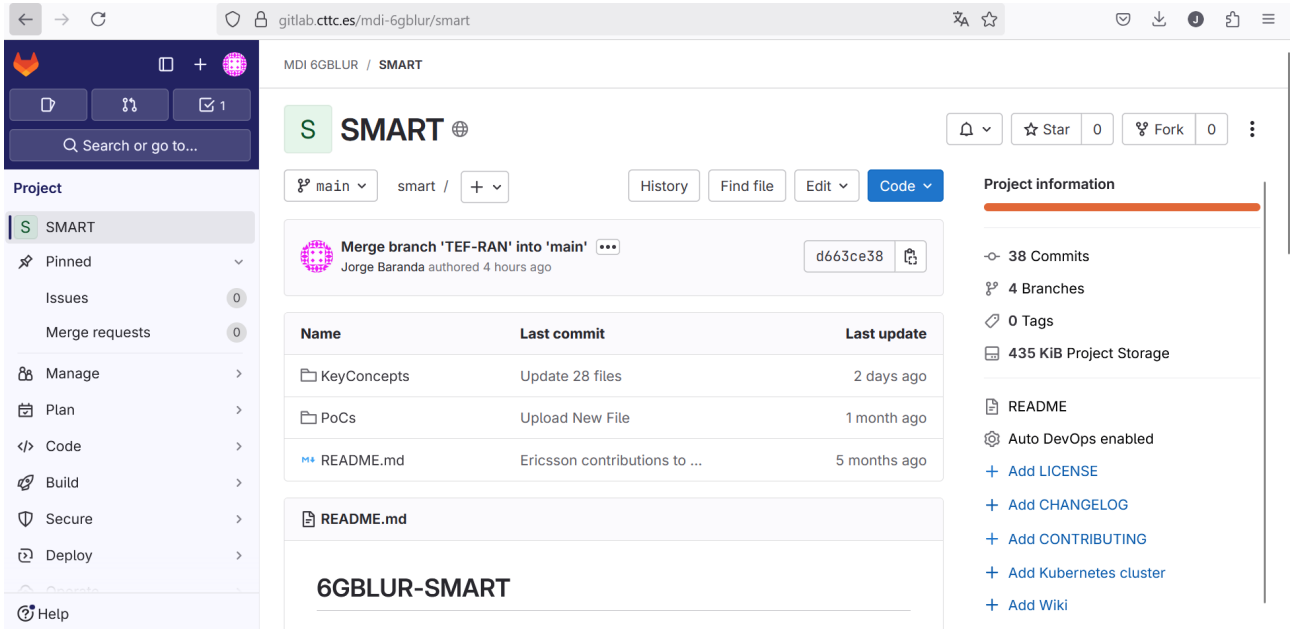

FIGURE 2. 6GBLUR SMART GITLAB REPOSITORY FRONTPAGE

Indeed, this brief report gives an overview of such outcomes and installation notes and useful links to support the installation of the components in third party environments and the re-use of its components in 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 between KCs and PoCs developed in SMART subproject based on the aspect they tackle, either RAN or MANO. 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 SMART KEY CONCEPTS INTO RAN AND MANO ASPECTS

Key Concept	RAN (E5)	MANO (E6)
K1.1: CU and DU implementation	X	
K1.2 : RAN Orchestration	X	

K1.4: Micro-orchestration of of virtual network functions at DU-RU level	X	X
K2.1: Development and characterization of FH and MH	X	
K2.2: Characterization of the FH and MH transport	X	
K2.3: Flexible Functional Split	X	
K2.5: QoS Management for time-critical immersive applications	X	X
K2.6: QoS strategies at the transport for multi-service aggregation		X
K3.1: AI/ML Workflows in O-RAN	X	
K3.2: AI/ML Platform	X	

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

UCPoC	RAN	MANO
UC2PoC1. Joint Fronthaul Capacity Control/Placement, Qos management and Flexible Functional Splits	X	X
UC2PoC2. NPN X-validation/optimization with Digital Twin	X	X
UC3PoC1. Data-Driven RAN optimization for NPN deployments	X	X
UC3PoC2. Optimization of DU-RU VNFs	X	X
UC3PoC3. Data-Driven O-RAN optimization based on xApps and rApps	X	X

2 SMART Key Concepts

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

2.1 SMART-K1.1: CU and DU implementation

2.1.1 New RLC E2 Agent

SRS has carried out several improvements to the E2 interface within the srsRAN CU/DU solution. We recall here that E2 is the O-RAN interface that allows the near-RT RIC to connect to the CU and the DU, extract metrics and control procedures and functionalities, thus enabling an ML-driven RAN. All the code is available from the srsRAN Project repository (more specific pointers are provided below).

The specific E2 components that have been developed in the context of 6GBLUR-SMART are listed next.

- The E2 agent at the RLC layer inside the DU, that allows the collection and exposition of metrics about data volume at the RLC SDU level (see https://github.com/srsran/srsRAN_Project/tree/main/lib/rlc).
- The metrics
    - *DRB.RlcSduTransmittedValumeUL_Filter*
    - *DRB.RlcSduTransmittedValumeDL_Filter*

for measuring the UL (DL, respectively) volume of data counted at the RLC SDU level (see https://github.com/srsran/srsRAN_Project/blob/main/lib/e2/e2sm/e2sm_kpm/e2sm_kpm_du_meas_provider_impl.cpp).

- E2SM-RC Control Style 2, Action 6, that controls the maximum number of PRBs that can be allocated to a UE (see https://github.com/srsran/srsRAN_Project/blob/main/lib/e2/e2sm/e2sm_rc/e2sm_rc_control_action_du_executor.cpp).

2.1.2 srsRAN for Disaggregated Deployments on Kubernetes via Helm Charts

SRS has worked on small adjustments that allow the srsRAN software to be deployed on Kubernetes via Helm Charts, as envisioned in JOINT-K1.2 and showcased in UC1PoC2. In particular, the srsRAN configuration file now offers the option to configure bind addresses for the F1-U and N3 interfaces that are different from the real addresses. This is required in virtual environments since the real address may not be accessible from the outside. The relevant code can be found here and here.

2.1.3 Slice-aware MAC Scheduler

SRS has developed a novel slice-aware MAC scheduler, which allows the operator to bound the RAN resources assigned to each network slice, thus contributing to the end-to-end isolation of the slices.

The example configuration below will activate two slices, namely (sst=1, sd=1), and (sst=1, sd=2). The minimum and maximum number of PRBs available for each slice is expressed as a percentage of the total available number of PRBs. The scheduler policy (round robin, proportional fair, guaranteed bitrate) can also be set independently for each slice.

TABLE 3. SAMPLE SLICE FILE CONFIGURATION SECTION IN SRS SOFTWARE

```
cell_cfg:
  slicing:
    -   sst: 1
        sd: 1
        sched_cfg:
           min_prb_policy_ratio: 40
           max_prb_policy_ratio: 80
        policy_sched_cfg:
           pf_sched:
             pf_sched_fairness_coeff: 5.0
    -   sst: 1
        sd: 2
        sched_cfg:
          min_prb_policy_ratio: 0
          max_prb_policy_ratio: 20
        policy_sched_cfg:
          pf_sched:
            pf_sched_fairness_coeff: 5.0
```

The code implementing this new functionality is available as part of the srsRAN Project suite. The design and functionality of the scheduler are described in detail in the project Deliverable E4, Section "*SMART-K1.1: CU and DU Implementation.*" The srsRAN configuration files for the experiments can be found here (it requires UE terminals capable of emulating the transmission over a ZMQ virtual radio, as explained in this tutorial).

2.2 SMART-K1.2: RAN Orchestration

CTTC developed this key concept presents various Fronthaul Control mechanisms that optimize the fronthaul resource utilization, along with a study related to the impact they have on applications generating downlink XR and CG traffic, has been included in the open-source public repository of 5G-LENA. In particular, it was included in the release:

- 5G-LENA release version v3.3 The development work starts with the following commit: 713528aac675d61c310d99056af3567791a331be

Also, the following example was developed and used to extract the results: cttc-nr-fh-xr.

Summary: 5g-lena-v3.3 includes Fronthaul Control mechanisms, that allow to simulate a limited-capacity fronthaul (FH) link based on the Fronthaul Capacity (m_fhCapacity) set by the user in the example script, and to apply FH Control methods (m_fhControlMethod) in order to restrict user allocations, if these they do not fit in the available FH capacity. O-RAN 7.2x functional split is assumed. The methods supported are the Dropping, Postponing, Optimize MCS, Optimize RBs and AdjustRBs, however it can be easily extended to apply additional methods. The Fronthaul Control takes decisions based on the model applied, the available fronthaul capacity, the applied modulation compression and the number of active UEs in each cell (either with new or HARQ data). This last information is updated after each scheduling process is finalized, through the MAC layer. The interaction of the Fronthaul Control with the MAC and high-PHY layers takes place through a set of SAP interfaces that allow the bidirectional exchange of information. The cttc-nr-fh-xr example shows how to use and configure the Fronthaul Control in a scenario employing XR, CG and VoIP traffic. For a more detailed description see the NR manual and Doxygen.

A detailed description can be found in project Deliverable E4, Section "*SMART-K1.2: RAN Orchestration*".

2.3 SMART-K1.4: Micro-orchestration of virtual network functions at DU-RU level [ADAPT]

Within the key-concept SMART K1.4, a framework for the adaptive offloading of low-PHY Radio Unit (RU) functions (e.g., Fast Fourier Transformation (FFT), Inverse FFT (iFFT), channel estimation, Physical Random Access Channel (PRACH)) between the Programmable Logic (PL) and the Application Processing Unit (APU) in AMD Radio Frequency System-on-Chip (RFSoC)/Multiprocessor System-on-Chip (MPSoC) devices has been developed. This part of the repo contains a notebook implementing a Reinforcement Learning (RL) simulation using Proximal Policy Optimization (PPO) to learn an *intelligent hybrid-offloading strategy* between: APU (CPU-like software execution) and PL (FPGA/Hardware acceleration).

This is done in a synthetic environment that models:

PRB utilization	Hazard/risk level	Latency	Thermal Evolution
Safety Penalty	Energy Consumption	Switching Cost (Implicitly measured)	

The agent learns when to run FFT-like tasks on the APU or offload to PL depending on dynamic network conditions and thermal state.

Environment (HybridOffloadEnv)

The environment exposes:

- **State**: [PRB utilization, hazard level]
- **Actions**: 0 = APU, 1 = PL
- **Internal thermal model**:

  temp += alpha * power - beta * (temp - ambient)
- **Reward**: negative of

  energy + latency + safety_penalty + thermal_penalty
- The episode lasts 100 steps.

This is a lightweight proxy for a real 6G RAN compute offloading scenario.

2.3.1 PPO Implementation

It includes:

- Actor–Critic network (2-layer MLP)
- Categorical policy for discrete actions
- PPO clipped objective:

  L = -min(ratio * adv, clipped_ratio * adv) + 0.5 * value_loss

The code correctly:

- Computes discounted returns
- Computes advantage = returns – baseline
- Uses 4 PPO update epochs per batch

Training Loop

For each seed:

- Run epochs × 100 steps

- Collect reward + telemetry:
    - Energy; latency; safety penalty; entropy; temperature; switching frequency
- Every 10 epochs → compute a policy heatmap over the 2D state space.

Returns structured logs for plotting.

Visualizations

The notebook finally plots:

1. PPO reward convergence (mean ± std)
2. Energy–latency Pareto scatter
3. Policy entropy over time
4. Cumulative switching count
5. Policy heatmap evolution across epochs

These give a nice interpretation of how the RL agent learns and stabilizes.

File Structure

```
HybridOffloadEnv    # Simulation environment
PPOAgent            # Actor–Critic + PPO update
train_agent()       # Training loop with metrics
plotting section    # Visualization utilities
```

Dependencies

Install via pip:

```
pip install torch numpy matplotlib pandas
```

2.4 SMART-K2.1: Development and characterization of FH and MH

The key-concept SMART K2.1 spurred two significant improvements in the O-FH library of srsRAN Project.

The adoption of **DPDK** allows the O-FH library to manage network packets directly at the NIC, without the need for a context switch to the kernel space or for intermediate copies of large amounts of data. The code is available **at this location** within the **srsRAN Project repository**.

The **SIMD** implementation of the O-RAN O-FH compression/decompression algorithms has provided a great processing gain, thanks to instruction parallelization. Most of the source files can be found in **this folder** of the srsRAN Project repository.

New configuration options (ru_ofh.cells.vlan_tag_cp and ru_ofh.cells.vlan_tag_up) allow the user to configure different vLANs for the control-plane and the user-plane O-FH traffic, thus achieving traffic segmentation even with a single NIC.

More details about the implementations and a preliminary performance evaluation can be found in the project Deliverable E4, Section "*SMART K2.1: Development and Characterization of FH and MH.*" A detailed measurement campaign showing the performance improvements produced by the introduction of DPDK and SIMD-based compression is reported **here**.

2.5 SMART-K2.2: Characterization of the FH and MH transport

This part of the repo contains the source code files for a Matlab environment consisting of the FH timing simulator described below[1].

2.5.1 FH TIMING SIMULATOR TOOL v1.0

MATLAB-based software application designed to simulate and analyse fronthaul (FH) timing conditions within centralized radio access networks. Its main purpose is to evaluate and determine the optimal timing configuration between the O-DU (Distributed Unit) and the O-RU (Radio Unit) in Open RAN architectures.

The tool is built considering Telefónica de España's IP/Fusión transport network architecture, leveraging it as a reference model to provide realistic and operator-specific insights. By simulating the transport and processing delays across the fronthaul segment, the tool assesses critical timing parameters, which are crucial for configuring the O-DU Transmission Window (TW) in compliance with O-RAN Alliance specifications.

The software provides comprehensive outputs, including key numerical parameters and visual diagrams. These diagrams illustrate the FH timing windows for both downlink (DL) and uplink (UL), covering both the control plane and the user plane.

Ultimately, this software allows going beyond conventional analysis by enabling exploration of deployment scenarios that exceed the standard 20 km fronthaul distance limit defined by the O-RAN Alliance. Through accurate modelling and comprehensive data presentation, the tool supports advanced design decisions for extending centralized RAN deployments over longer distances while ensuring strict timing requirements are met.

The software is composed of the following functions:

- FH_timing_simulator_main_v1: is the main script, which must be launched to run the tool.

[1] This Matlab environment was also used to develop the SMART-K2.4: Radio Stack optimization. Further details available in E4 report [3].

- get_centralization_distance: return the centralization distance of the stage in the range chosen by the user.
- network_design: return the minimum and maximum network delays in DL and UL based on network topology.
- network_topology: create the aggregation network topology and display its simplified topology.
- reduce_to_T1a_max_up_O-DU: reduce the FH propagation delay under the O-DU capability.
- show_diagram_DL: displays DL visual outputs.
- show_diagram_UL: displays UL visual outputs.
- show_results: show the FH resulting timing (and the corresponding distances).
- time_distance_conversion: convert propagation time in fiber to linear distance and vice versa.
- timing_analysis_DL: perform timing analysis of FH delays in downlink.
- timing_analysis_UL: perform timing analysis of FH delays in uplink.
- user_inputs: collect user input to give values to inputs.

After running the main script, the user is prompted to select the number of simulations, the location of the O-DU, and the desired distance case. The simulation of transport conditions is based on a real operator's aggregation network. In this scenario, the O-DU is centralized from the cell site (HL5: Hierarchical Level 5) to two levels upwards, either HL4 or HL3, depending on the level of centralization required for the analysis. The distance cases are defined as follows:

- HL4:
    - Number of aggregated cell sites per vDU pool: approximately 9.
    - Shortest case: 10-15 km
    - Average case: 15-20 km
    - Longest case: 20-25 km
- HL3:
    - Number of aggregated cell sites per vDU pool: approximately 23.
    - Shortest case: 35-45 km
    - Average case: 45-55 km
    - Longest case: 55-65 km

Given the input parameters related to the O-RU and the characteristics of the transport network, the tool processes the scenario and adjusts the configuration to maximize the centralization distance of the O-DU while ensuring that all time-sensitive requirements of the O-FH are strictly met.

Once the evaluation is complete, and if the scenario is successful, the tool provides the results in multiple formats. These include detailed outputs in the command window, graphical visualizations that help interpret key metrics, and an automatically generated file summarizing all relevant parameters and outcomes. This allows for both high-level insights and in-depth technical analysis to support design decisions and further optimization.

2.6 SMART-K2.3: Flexible functional split

CTTC has extended the ns-3 5G-LENA simulator to evaluate various functional split options as a first step toward exploring the potential of Flexible Functional Splits (FFS). Building on the existing implementation, which already includes capacity constraints for the 7.2 split, this work adds support for functional split options 6, 7.1, and 7.3, each with their own specific fronthaul capacity requirements that limit packet transmission. Additionally, the 7.2 split has been considered without applying modulation compression techniques. This contribution is the first step towards the development of FFS in the ns-3 5G-LENA simulator to select the most appropriate functional split based on traffic demand, network load, and scenario conditions.

The code is available in the 5G-LENA repository, in the merge request number !278 (*https://gitlab.com/cttc-lena/nr/-/merge_requests/278*) Currently it is inline with ns-3.45 version and nr v4.0. The code that enables or disables modulation compression is available in commit [f49ec5b4c63ea6508d91bd1656c1a1ecc9b39fa0] on the master branch of 5G-LENA.

The specific components developed as part of the 6GBLUR-SMART project are listed below:

- The *GetFhThr* function within the nr-fh-control module has been extended to include a new variable, m_*funcSplit*, which specifies the functional split to be used in the current simulation. This enables accurate calculation of the fronthaul throughput based on the selected FS.
- To support the calculation for different FS options, *GetFhThr* now also receives information about number of allocated OFDM symbols (*numSym*) and number of antenna ports (*antennaPorts*). These parameters are necessary for computing throughput for splits 6 and 7.3 (using *numSym*), and for split 7.1 (using *antennaPorts*).
- In the nr-fh-control, a new attribute named *FunctionalSplit* has been added. This attribute allows users to configure the desired FS during simulation through the cttc-nr-fh-xr example. It can be set either directly in the example or passed externally through a script or command line, enabling automated simulations. An internal mapping function converts the input string into a predefined value that the simulator understands.
- Finally, a new attribute has been added to enable or disable modulation compression for split 7.2 (*EnableDynamicModComp*). The throughput calculation for this split has been modified accordingly: if modulation compression is disabled, the effective modulation order is set to 32; otherwise, it matches the actual modulation order.

The design and analysis of different functional splits are described in detail in the project Deliverable E4, Section "*SMART-K2.3: Flexible functional split.*"

2.7 SMART-K2.5: QoS Management for time-critical immersive applications

CTTC work developed in KC SMART K2.5, where we proposed and implemented a new generalized QoS MAC scheduler, capable of managing properly various traffic types, while guaranteeing the requirements of time-critical traffic, has been included in the open-source public repository of 5G-LENA. In particular, it was included in the release:

- 5G-LENA release version v2.5 The development work starts with the following commit: 5f6ea7b2e2dd7b22f49f2040e44686ef35ac01c3

Also, the following examples were developed and used to extract the results:

- cttc-nr-simple-qos-sched.cc
- cttc-nr-multi-flow-qos-sched.cc

Finally, we have created a system test that ensures the correct functionality of the scheduler:

- system-scheduler-test-qos.

Summary: 5g-lena-v2.5 includes new QoS schedulers that perform scheduling by taking into account the QoS requirements of QoS flows. NrMacSchedulerTdmaQos and NrMacSchedulerOfdmaQos classes are responsible for setting the scheduler and access mode types when desired by the user and updating the DL and UL metrics of each UE. NrMacSchedulerUeInfoQos performs the sorting of the UEs (based on DL and UL metrics). Moreover, a new design for LC bytes assignment is included that allows the implementation of various algorithms. A new base class is added, known as NrMacSchedulerLcAlgorithm that allows the implementation of various algorithms for the LC byte assignment. Two algorithms are implemented. NrMacSchedulerLcRR that includes the original implementation of assigning bytes to LCs in RR fashion and NrMacSchedulerLcQos that shares bytes among the active LCs by taking into account the resource type and the e_rabGuaranteedBitRate of a flow. In addition, two examples are included, the cttc-nr-simple-qos-sched.cc with main purpose to test and validate the correct functionality of the new QoS MAC schedulers under XR traffic, and the cttc-nr-multi-flow-qos-sched that allows testing the performance of the QoS schedulers in conjunction with the LC QoS Assignment versus other schedulers, such as the RR and PF in conjunction with the LC RR scheduler. Finally, the system test system-scheduler-test-qos is used to verify the implemented QoS MAC schedulers by checking that the obtained throughput is as expected for the QoS scheduling logic. For a more detailed description see the NR manual and Doxygen.

In addition, CTTC and University of Cantabria implemented a new Lyapunov-based QoS MAC scheduler available in 5glena-lyapunov-mac-scheduler. This repository contains the scheduler implementation over 5G-LENA v2.5. In a nutshell, the scheduler is designed to stabilize traffic flows, guarantee the required throughput for each flow according to their QoS requirements, and promote efficient use of RBs.

The NrMacSchedulerOfdmaDPPA class is responsible for setting the scheduler and updating the DL metrics of each UE. RB allocation is handled by NrMacSchedulerUeInfoDPPA, which sorts UEs based on DL metrics, including a virtual throughput queue. This queue reflects the deviation between the achieved and the target throughput for the UE.

A detailed description can be found in project Deliverable E4, Section "*SMART-K2.5: QoS Management for time-critical immersive applications.*"

2.8 SMART-K3.1: AI/ML Workflows in O-RAN

The work developed in the workflow of this Key Concept is organized around the coordinated operation of two complementary xApps that exploit the O-RAN Key Performance Measurements (KPMs) and RAN Control (RC) capabilities developed by srsRAN.

2.8.1 KPM xApp

The first xApp (kpm_and_log.py) is dedicated to continuously monitoring traffic levels and PRB utilization across the radio access network. Using the standardized O-RAN KPM service, it gathers fine-grained performance indicators such as throughput, load distribution, PRB occupancy, and temporal fluctuations in demand. These measurements are processed in real time to build an accurate and up-to-date representation of network conditions, providing essential situational awareness regarding the behaviour of the radio interface. This xApp effectively acts as the sensing layer of the system, supplying the detailed operational data needed for higher-level decision making.

2.8.2 RC xApp

The second xApp (prb_controller_ul_from_kpm_log.py) consumes these KPMs and focuses on analysing traffic conditions to determine whether the current allocation of PRBs is optimal. By applying decision logic and policies enabled through the srsRAN implementation of the O-RAN RC framework, the xApp evaluates potential imbalances, congestion scenarios, or resource underutilization. When necessary, it formulates and issues RC control actions to modify PRB assignments dynamically, adapting resource allocation to better match the observed traffic patterns. This may involve increasing PRBs for heavily loaded cells, reducing allocations where demand is low, or balancing resources across the network to improve overall performance and spectral efficiency. Together, these two xApps establish a continuous, closed-loop process that follows the observe–analyze–act paradigm. The monitoring xApp ensures constant visibility into real-time network behaviour, while the decision-making xApp transforms these insights into targeted control actions aimed at optimizing the radio resources. This workflow demonstrates how O-RAN-compliant xApps,

supported by the srsRAN implementation of the KPM and RC specifications, can enable intelligent, autonomous, and responsive RAN management capable of adapting proactively to dynamic traffic conditions.

2.9 SMART-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 packet_loss_n6: Number of packet losses measured at the N6 interface
- Column packet_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 SMART UC PoCs

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

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

3.1.1 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.

3.1.1.1 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.

3.1.1.2 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.

3.1.1.3 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.

3.1.1.4 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.

3.1.1.5 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.

3.1.2 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-JOINT subproject. You can refer this 6GBLUR-JOINT project repository path - https://gitlab.cttc.es/mdi-6gblur/joint. The video was recorded during Face to Face November 2024 plenary meeting where Ericsson presented this PoC to partners.

3.2 UC3: RAN Control Architectures and Smart Algorithms

3.2.1 PoC1: Data-Driven RAN optimization for NPN deployments

This video file, named "UC3PoC1 Data-Driven RAN optimization for NPN.mp4", is related to UC3/PoC1: Data-Driven RAN optimization for NPN deployments.

This PoC focuses on data-driven optimization of Radio Access Network (RAN) within non-public network (NPN) deployments, leveraging new capabilities in 5G network management.

Please refer to UC3/PoC1: Data-Driven RAN optimization for NPN deployments 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.

3.2.2 PoC2: Optimization of DU-RU VNFs

This repository contains the **RFSoC Micro-Orchestrator** used in UC3PoC2, a complete research platform implementing:

- Predictive (proactive) orchestration using an LSTM forecasting model.
- Reactive (adaptive) orchestration using Proximal Policy Optimization (PPO) reinforcement learning.
- Dynamic offloading of low-PHY functions (FFT, iFFT, PRACH detection, channel estimation) between:
  - RFSoC PL (Programmable Logic),
  - RFSoC PS/APU (Arm cores),
  - optional Edge compute node.
- Context awareness from computer-vision triggers (e.g., street hazard detection).
- Online KPI monitoring for:
  - Latency
  - Energy consumption
  - Reliability
  - FFT/OFDM configuration (scaling FFT points)

The package includes runnable Python demos, hardware stubs, Vivado TCL skeletons, training notebooks, and a full PDF report with figures.

3.2.2.1 Repository Structure

```
rfsoc_micro_orchestrator/
├── README.md
```

```
├─── run_demo.sh
├─── run_demo.bat
├─── orchestrator/
│    ├─── predictive_agent.py       ← LSTM proactive model
│    ├─── reactive_agent.py         ← PPO reactive controller
│    ├─── micro_orchestrator.py     ← Main orchestrator logic
│    ├─── offloading_manager.py     ← PL/PS offloading decisions
│    ├─── kpi_collector.py          ← KPI logging & processing
│    ├─── data/
│    │    ├─── demo_trace.csv
│    │    └─── results/
│    └─── figures/
│         ├─── fig5a_latency.png
│         ├─── fig5b_energy.png
│         ├─── fig5c_reliability.png
├─── notebooks/
│    ├─── train_lstm_predictive.ipynb
│    ├─── train_ppo_reactive.ipynb
│    └─── visualize_results.ipynb
├─── hardware/
│    ├─── zcu111_bd.tcl
│    ├─── pl_offload_stub.c
│    ├─── ps_controller_stub.c
│    └─── ethernet_udp_stub.c
├─── docs/
│    ├─── report_full.pdf           ← Full scientific report with equations & figures
│    └─── README_hardware.md
└─── utils/
   ├─── logger.py
   ├─── config.yaml
   └─── helpers.py
```

- **File:** *predictive_agent.py*. Implements an **LSTM model** that forecasts: PRB usage, latency, energy, and external hazard context score over a future time window. This enables **proactive pre-allocation** of PL/PS compute resources.
- **File:** *reactive_agent.py*. Uses **PPO reinforcement learning** to take instant corrective actions when KPIs deviate from expected levels. Actions include:

  PL → PS offload; PS → PL return; Scaling FFT points; Switching to edge execution
  Increasing sampling Windows; Using fallback modes during CV hazard events

- **File:** *micro_orchestrator.py*. Integrates predictive + reactive layers:

  forecast → proactive plan → observe state → PPO correction → offload decision

  This unified control loop allows the system to:

  (1) anticipate load changes, (2) react to unexpected deviations, (3) minimize energy consumption, (4) maintain low latency and reliability constraints.
- **File:** *offloading_manager.py*. Implements the **RFSoC offloading decision logic**, and integrates with:

  PL FFT accelerator stub (pl_offload_stub.c)
  PS FFT implementation (Python or C)
  DMA/OCM data movement model
  Optional edge execution

  Supports dynamic FFT-size reconfiguration.
- **File:** *kpi_collector.py*. Logs: end-to-end latency, PL & PS energy model outputs, task success/errors (reliability), and data-rate & FFT timing. Plots are written to: orchestrator/data/results/

3.2.2.2 How to Run Demos, Analyze KPIs & Train Models

Install Requirements

pip install -r requirements.txt or manually ensure that the following libraries are installed: numpy, pandas, matplotlib, torch, stable-baselines3, onnx, jupyter.

Run the Micro-Orchestrator Demo

- Linux/macOS: ./run_demo.sh (first you may need to make it executable with: *chmod +x run_demo.sh*)
- Windows: run_demo.bat

This runs:

python orchestrator/micro_orchestrator.py --simulate

and the output goes to folder: *orchestrator/data/results/* , where you will find:

- kpi_log.csv
- offloading_log.csv
- fig_latency.png
- fig_energy.png
- fig_reliability.png

Train the Predictive LSTM Model

jupyter notebook notebooks/train_lstm_predictive.ipynb

This notebook:

- loads historic KPI traces
- trains an LSTM
- saves the model as predictive_model.pt

Train the PPO Reactive Controller

jupyter notebook notebooks/train_ppo_reactive.ipynb

The PPO agent:

- interacts with a simulation environment
- learns to minimize latency & energy
- exports ppo_reactive_policy.zip

Visualize KPI Comparisons

jupyter notebook notebooks/visualize_results.ipynb

Generates plots comparing:

- PL-only
- PS-only
- static hybrid
- LSTM-only
- PPO-only
- combined orchestrator (best performance)

3.2.3 PoC3: Data-Driven O-RAN optimization based on xApps and rApps

This Proof of Concept (PoC) focuses on offering a closed-loop system for monitoring and controlling O-RAN networks. An adaptative monitoring schema with integrated intelligence into the O-RAN stack has been developed and tested by means of including LLMs (Large Language Models), to automate tasks and reduce human feedback. This work builds upon several of the key concepts described previously.

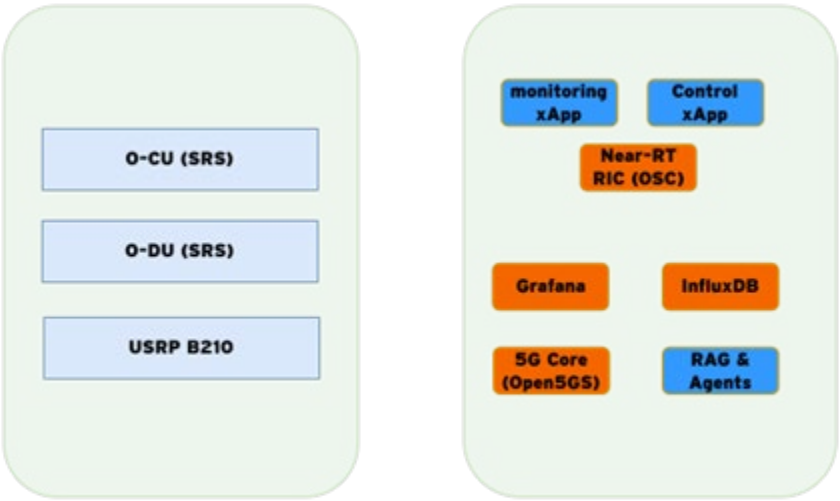

FIGURE 3. DIFFERENT ELEMENTS USED FOR THE UC3POC3

This work builds upon several of the key concepts described in detail in E4. 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.
- SMART-K3.1 – AI/ML Workflows in O-RAN: Implementation of monitoring and controlling xApps and integrated in an agentic based platform.
- JOINT-K3.1 – Monitoring Platform: Agentic based monitoring elements of RAN components.

Please refer to the UC3PoC3 section in the E4 deliverable for further details about the implementation of these PoC.

In this PoC, SRS contributed with several improvements in the E2 interface that will allow an ML-based monitoring of and control over the RAN, as detailed in SMART-K1.1.

A simple xApp, preliminary to the full PoC, has also been developed to showcase the possible interaction between E2SM-KPM and E2SM-RC: the xApp, running in the near-RT RIC, monitors the DL traffic of a UE by requesting updates via the *DRB.RlcSduTransmittedVolumeDL_Filter* metric and modifies the PRB quota of the UE via Action 6 after every 20MB of data. The components and the instructions of the demo can be found here. One can also refer to this article for further details. Even though his tutorial is about a different xApp, the set-up is exactly the same and one should only pay attention to run the following command to launch the simple_xapp xApp, instead of the kpm_mon_xapp xApp used in the tutorial.

```
docker compose exec python_xapp_runner ./simple_xapp.py
```

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 SMART 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/