



Grant Agreement N°: 957317  
Topic: ICT-42-2020  
Type of action: IA



# AFFORDABLE 5G

## D4.2: 5G roll-out and system testing report

Work package	WP 4
Task	Task 4.2
Due date	30 <sup>th</sup> June 2022
Submission date	19th July 2022
Deliverable lead	CELLNEX
Version	1.1

## Abstract

The goal of this deliverable is to define the principles and procedures that have been followed to integrate the building blocks reported in the previous deliverables, mainly D1.2, D1.3, D2.2, D3.2 to become part of the two 5G-NPN based platforms used for the final demonstrations and trials in the project. The integration plan defines how the consortium managed to map the Affordable5G architecture components and equipment to Castellolí and Malaga 5G-NPN platforms. The focus is also on defining the test cases to ensure that the building blocks are properly working in the target platforms, while also test results and integration and unit tests are described. This document provides the second, and final, version of test cases that have been expanded and implemented during the second year of the Affordable5G project lifetime.

**Keywords:** 5G Non-Public Networks, O-RAN, NFV, AI/ML-based network optimization, System architecture

### List of Contributors

Partner	Short name	Contributor(s)
ATOS SPAIN SA	ATOS	Sergio González, Borja Otura and Josep Martrat
ADVA Optical Networking Israel Ltd	ADVA	Andrew Sergeev
RETEVISION I SA	CEL	Judit Bastida
ACCELLERAN	ACC	Simon Pryor
ATHONET SRL	ATH	Nicola di Pietro, Daniele Munaretto and Daniele Ronzani
THINK SILICON EREYNA KAI TECHNOLOGIA ANONYMI ETAIRIA	THI	Georgios Keramidas
RUNEL NGMT LTD	REL	Israel Koffman and Baruch Globen
NEMERGENT SOLUTIONS S.L.	NEM	Marta Amor, Eneko Atxutegi and Aarón Rodríguez
UBIWHERE LDA	UBI	Rita Santiago , Roni Fernades Sabença and Diogo Guedes
MARTEL GMBH	MAR	Gabriele Cerfoglio, Andrea Falconi and Giacomo Inches
EIGHT BELLS LTD	8BELLS	George Kontopoulos
NEARBY COMPUTING SL	NBC	Oscar Trullols and Angelos Antonopoulos
UNIVERSIDAD DE MALAGA	UMA	Pablo Herrera Díaz, Javier Andrés Jiménez Jiménez, Francisco Luque Schempp and Pedro Merino Gómez
ETHNIKO KAI KAPODISTRIAKO PANEPISTIMIO ATHINON	NKUA	Panagiotis Trakadas, Lambros Sarakis, Panagiotis Gkonis, Sotirios Spantideas and Anastasios Giannopoulos
FUNDACIO PRIVADA I2CAT, INTERNET I INNOVACIO DIGITAL A CATALUNYA	I2CAT	Juan Camargo, Wilson Ramirez
UNIVERSITAT POLITECNICA DE CATALUNYA	UPC	Jordi Pérez-Romero, Oriol Sallent
EURECOM	EUR	Navid Nikaein, Sofia Pison

### List of reviewers

Partner	Name
ATOS	Sergio Gonzalez, Rosana Valle (QA)
NKUA	Panos Gkonis
UMA	Pablo Herrera

### Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	11/11/2021	ToC	CELL
V0.2	12/05/2022	First round of contributions	ALL
V0.3	10/06/2022	First review version	NKUA
V0.3	15/06/2022	1st Review comment included	CELL, ALL
V0.4	17/06/2022	Second review version	UMA
V0.5	22/06/2022	Final refinements	ALL
V0.6	30/06/2022	Final version	ALL
V1.0	08/07/2022	QA check and submission	ATOS
V1.1	19/07/2022	Update of section 4.1.4 and QA check	ATH, ATOS

### Disclaimer

The information, documentation and figures available in this deliverable, is written by the Affordable5G (High-tech and affordable 5G network roll-out to every corner) – project consortium under EC grant agreement 957317 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

**Copyright notice:** © 2020-2022 Affordable5G Consortium

### Project co-funded by the European Commission in the H2020 Programme

<b>Nature of the deliverable:</b>		R	
<b>Dissemination Level</b>			
PU	Public, fully open, e.g., web		√
CI	Classified, information as referred to in Commission Decision 2001/844/EC		
CO	Confidential to Affordable5G project and Commission Services		

## EXECUTIVE SUMMARY

---

In the first part of this deliverable, an update of the architecture instantiation in each testbed describing the updated building blocks and the integrations that have been performed is provided.

In Malaga Campus platform, the main integrations are related to the OSM orchestration system, 5G Core updates and O-RAN extensions and proper integrations between the partners' components. The 5G Core integrations with the dRAX, the integrations performed with regards of the TSN over 5G and the OSM extension and integration and the description of the status of the integration between the RU and the DU are provided as well.

In Castellolí platform, the described integrations are between the NearbyOne orchestrator, the Slice Manager, the 5GCore, the NWDAF component and O-RAN components (followed by the developed xApps) that have been integrated in the platform. The new version of dRAX has also been implemented in Castellolí, and has been integrated with the Orchestrator, as well as the Slice Manager and the last version of the 5GCore. Moreover, the Nemergent MCS applications have been deployed as well, but their integration is an on-going work for the last part of the project.

Additionally, a revised timeline for each of the testbeds is presented, showing the updated strategy plan that the consortium has followed to achieve all the activities expected for the second year of the project and, specifically, redefined to meet the deadlines for the end of the project: perform end to end system test cases and pilot validation.

This deliverable also contains a detailed definition of the test cases: First of all, the individual test cases, used to verify the behavior of each component operating in a standalone fashion are described. After the component-individual test case definition, the integration test cases are defined in order to describe all the test cases that took place to review the correct interaction between affordable 5G components in each of the two platforms. The most advanced integrations present the results of their test cases as well as the respective summary tables describing the relevant KPIs.

Finally, Affordable 5G pilots are described in specific sections for TSN over 5G Proof of Concept, Smartcity with beyond state-of-the-art video streaming application and Mission Critical Services, covering several scenarios and requirements from the 5G network perspective. Each pilot describes its building blocks and indicates a good integration progress and provides current deployment status.

## TABLE OF CONTENTS

<b>EXECUTIVE SUMMARY .....</b>	<b>4</b>
<b>TABLE OF CONTENTS .....</b>	<b>5</b>
<b>LIST OF FIGURES .....</b>	<b>7</b>
<b>LIST OF TABLES.....</b>	<b>10</b>
<b>ABBREVIATIONS .....</b>	<b>11</b>
<b>1 INTRODUCTION.....</b>	<b>14</b>
<b>2 INTEGRATION AND TESTING IN MALAGA CAMPUS .....</b>	<b>15</b>
2.1 Malaga Campus Architecture instantiation .....	15
2.2 Malaga Campus Integrations.....	16
2.2.1 5G Core Integrations.....	16
2.2.2 OSM integrations .....	20
2.2.3 O-RAN Integrations.....	22
2.3 Integration and Testing Timeline definition .....	26
<b>3 INTEGRATION AND TESTING IN CASTELLOLÍ.....</b>	<b>27</b>
3.1 Castellolí's Architecture instantiation .....	27
3.2 Castellolí's Integrations .....	27
3.2.1 5G Core Integrations.....	28
3.2.2 Orchestrator Integrations.....	29
3.2.3 SMO and ORAN integration .....	30
3.2.4 O-RAN Fronthaul & Synchronization Integrations.....	33
3.3 Integration and Testing Timeline definition .....	34
<b>4 TEST CASE DEFINITION AND KPI MEASUREMENTS .....</b>	<b>35</b>
4.1 Individual Test Cases .....	35
4.1.1 OSM KNF Placement Test .....	35
4.1.2 Slice Manager .....	36
4.1.3 AI-ML .....	37
4.1.4 5G Core .....	44
4.1.5 Power Measurement Testbed for NEOX Accelerators.....	49
4.1.6 TSN over 5G .....	51
4.1.7 Orchestrator .....	55
4.1.8 Open 5G-RAN.....	60
4.1.9 Individual Test Cases KPIs Summary table.....	65
4.2 Integration Test Cases .....	66
4.2.1 RU-DU .....	66
4.2.2 DU-CU .....	69
4.2.3 CU – 5G Core .....	70
4.2.4 Slice Manager – Orchestrator.....	70
4.2.5 AIML – Telemetry - Orchestrator .....	73
4.2.6 Telemetry Execution in Far Edge Devices.....	80
4.2.7 Non-RT RIC (RIC manager) – dRAX.....	82
4.2.8 Orchestrator– 5G Core.....	90
4.2.9 Synchronization of TSN endpoint with ADVA master clock .....	90
4.2.10 Integration Test Cases KPIs Summary table .....	92
<b>5 TSN OVER 5G PROOF OF CONCEPT .....</b>	<b>94</b>
5.1 Introduction .....	94
5.2 Pilot Building Blocks .....	94
5.2.1 Translation from TSN domain to 5G .....	94

5.2.2	Prioritization over 5G.....	95
5.2.3	Synchronization.....	95
5.3	Integration and Deployment Status .....	96
<b>6</b>	<b>SMARTCITY .....</b>	<b>97</b>
6.1	Introduction .....	97
6.2	Pilot Building Blocks .....	98
6.2.1	Dynamic service.....	98
6.2.2	Static service.....	101
6.3	Integration and Deployment Status .....	103
6.3.1	Dynamic service.....	103
6.3.2	Static service.....	103
6.4	Integration and Deployment Status in THI Platform .....	104
<b>7</b>	<b>MISSION CRITICAL SERVICES .....</b>	<b>106</b>
7.1	Introduction .....	106
7.2	Pilot Building Blocks .....	107
7.3	Integration and Deployment Status .....	109
<b>8</b>	<b>CONCLUSIONS.....</b>	<b>113</b>
<b>9</b>	<b>REFERENCES .....</b>	<b>114</b>

## LIST OF FIGURES

Figure 1 High Level architecture Malaga.....	15
Figure 2 Some HEARTBEAT messages captured by Tcpdump during gNB and 5GC communication. ....	16
Figure 3 Illustration of the ITP-EU integration with partner's premises.....	17
Figure 4: Affordable5G/FUDGE-5G Open 5GS demo @ EuCNC 2022 .....	18
Figure 5: CU-CP creating CU-UP and DU attaching to demonstrate an active radiating gNB	18
Figure 6: UE attaching to gNB and authenticating with 5GC to establish active connection .....	19
Figure 7: 5G smartphone connected to and running over the Affordable5G Open SNPN	19
Figure 8 OSM Deployment .....	22
Figure 9: Network setup for O-RAN 7.2 in Eurecom.....	23
Figure 10: OAI O-DU integration with the Foxconn RU .....	24
Figure 11. Malaga TimePlan .....	26
Figure 12 High Level architecture Castellolí .....	27
Figure 13 Architecture of the 5G core and NWDAF integration.....	28
Figure 14 RIC manager proposed architecture.....	31
Figure 15 Endpoints of RIC Manager's API.....	31
Figure 16 dRAX 5G A1 Policy Type .....	32
Figure 17 Castellolí Fronthaul infrastructure (LLS-C3) .....	33
Figure 18 Castellolí Timeplan .....	34
Figure 19 Affordable5G AI/ML Framework architecture.....	38
Figure 20 GUI showing the AI/ML pipeline successfully loaded .....	39
Figure 21 CLI showing the AI/ML pipeline successfully loaded.....	39
Figure 22 GUI showing the AI/ML pipeline enabled.....	40
Figure 23 GUI showing the AI/ML pipeline graph view with all the pipeline components successfully executed.....	40
Figure 24 AMD logs showing the successful deployment of the model in the MSP .....	41
Figure 25 MSP configuration file automatically updated by the AMD .....	42
Figure 26 MSP REST response showing the availability of a test model that is being served correctly .....	43
Figure 27 MSP REST response to a model prediction (half_plus_two) .....	43
Figure 28 CDF of the prediction latency of a toy model (half_plus_two) and the CPU prediction model provided by I2CAT.....	44
Figure 29 Simulation testbed for measuring power consumption of NEOX accelerator	50

Figure 30 Latency and throughput for NEOX accelerator when the number of cores is modified.....	50
Figure 31 Power and energy of NEOX accelerator when the number of cores is varied 51	
Figure 32 E2E latency results .....	52
Figure 33 e2e jitter results.....	53
Figure 34 UP shortcut testbed – baseline .....	54
Figure 35 Concourse-ci screenshot showing the provisioning device tests. ....	57
Figure 36 Cypress video's screenshot showing the e2e-provision-vm2 in « provisioning » status at time 24.20s of the test. ....	57
Figure 37 Cypress video's screenshot showing the e2e-provision-vm2 in « Ready » status at time 895.30s of the test. ....	57
Figure 38 Concourse-ci screenshot showing the provisioning device tests. ....	59
Figure 39 Cypress video's screenshot showing the sample-deployment « ambient-al block » in « provisioning » status at time 9.90s of the test.....	59
Figure 40 Cypress video's screenshot showing the sample-deployment « ambient-al block » in « Ready » status at time 33.12 s of the test. ....	59
Figure 41 Cypress video's screenshot showing the sample-deployment « ambient-al block » in « Undeploying » status at time 34.90 s of the test.....	60
Figure 42: PCAP captura of PTP, C-Plane and U-Plane traffic.....	65
Figure 43 NearbyOne orchestrator logs showing a GET request to the Slice Manager	73
Figure 44 NearbyOne Orchestrator logs showing POST creating a ran slice in Slice Manager .....	73
Figure 45 Metric "container_cpu_usage_seconds_total" obtained from the Prometheus instance by the Message Broker .....	75
Figure 46 Metric "container_network_transmit_bytes_total" obtained from the Prometheus instance by the Message Broker .....	76
Figure 47 Message Broker logs showing the publication of the prediction in the queue 78	
Figure 48 Test script showing that the prediction is properly published in the queue	78
Figure 49 Prometheus dashboard showing the prediction data that was published in the Message Broker .....	79
Figure 50 Prometheus dashboard showing the rules defined monitoring the predictions .....	80
Figure 51 Prometheus dashboard showing the triggering alarms.....	80
Figure 52 Screenshot with the execution of test "Registration of Mocked Non-RT dRAX 5G" 83	
Figure 53 Screenshot with the execution of test "Registration of dRAX 5G" .....	84
Figure 54 Screenshot with the execution of test "Deployment of the Telemetry xApp" 85	
Figure 55 Screenshot with the execution of test "Creation of dRAX's Policy Type" ....	86



Figure 56 Screenshot with the execution of test "Creation of dRAX's Policy Type" ....	87
Figure 57 Screenshot with the execution of test "List all the information stored about dRAX" .....	88
Figure 58 Screenshot with the execution of the "Delete Policy test" .....	89
<i>Figure 59 Screenshot with the execution of test "Undeployment of the Telemetry xApp"</i> .....	89
Figure 60 Output of ptp4l command.....	91
Figure 61 Output of phc2sys command .....	92
Figure 62 TSN over 5G PoC final architecture .....	94
Figure 63 TSN over 5G PoC current architecture .....	96
Figure 64 Smartcity Pilot Architecture .....	97
Figure 65 Smartcity steps for re-identification .....	98
Figure 66 Services workflow .....	99
Figure 67 Smartcity association between different components .....	100
Figure 68 ML Algorithm workflow.....	102
Figure 69 Smartcity components involving service work.....	102
Figure 70 Simplified vision of Pilot1.....	106
Figure 71 Pilot1 in different PoP .....	106
Figure 72 MCS service in Affordable 5G network.....	107
Figure 73 Simplified interaction between modules in Pilot1 .....	107
Figure 74 Scalability use case workflow .....	108
Figure 75 Interactions between MCS service, monitoring and orchestrator .....	108
Figure 76 Dockerized MCS service.....	109
Figure 77 Nemergent MCS service deployed in Castellolí .....	110
Figure 78 Nemergent MCS service Pods deployed in Castelloli .....	111
Figure 79 Nemergent MCS service deployed in Castellolí with NodePort .....	111
Figure 80 Nemergent MCS service deployed in Castellolí with storage classes.....	112

## LIST OF TABLES

---

Table 1 Individual Test Cases KPIs Summary Table .....	65
Table 2 Integration Test Cases Summary Table .....	92
Table 3 Emergency Communications port exposure in NodePort mode.....	110
Table 4 Status update of end-to-end testing of deployed Pilot 1 in Castellolí.....	112

## ABBREVIATIONS

---

<b>3GPP</b>	Third Generation Partnership Project
<b>4G</b>	Fourth Generation
<b>5G</b>	Fifth Generation
<b>5GC</b>	5G Core
<b>AI</b>	Artificial Intelligence
<b>AF</b>	Application Function
<b>AGV</b>	Automated Guided Vehicle
<b>AMF</b>	Access and Mobility Function
<b>AMR</b>	Autonomous Mobile Robots
<b>API</b>	Application Programming Interface
<b>AUSF</b>	Authentication Server Function
<b>BC</b>	Boundary Clock
<b>BSS</b>	Business Support System
<b>CAPIF</b>	Common API Framework
<b>C-MDAF</b>	Centralized Management Data Analytics Function
<b>CN</b>	Core Network
<b>CNN</b>	Convolutional Neural Networks
<b>CP</b>	Control Plane
<b>CU</b>	Central Unit
<b>CPRI</b>	Common Public Radio Interface
<b>DA</b>	Data Analyzer
<b>DC</b>	Data Collector
<b>DL</b>	Deep Learning
<b>DN</b>	Data Network
<b>DNN</b>	Data Network Name
<b>DRL</b>	Deep Reinforcement Learning
<b>DSCP</b>	Differentiated Services Code Point
<b>DS</b>	Data Source
<b>DS-TT</b>	Device Side Translator
<b>DSF</b>	Data Semantic Fabric
<b>DP</b>	Data Plane
<b>DU</b>	Distributed Unit
<b>E2E</b>	End to End
<b>eMBB</b>	enhanced Mobile Broadband
<b>EMS</b>	Element Management System
<b>EMS-CM</b>	EMS - Configuration Management
<b>ENI</b>	Experiential Network Intelligence
<b>FEC</b>	Forward Error Correction
<b>FoF</b>	Factory of the Future
<b>gNB</b>	Next Generation NodeB
<b>gNMI</b>	gRPC Network Management
<b>IM</b>	Infrastructure Monitoring
<b>IMF</b>	Infrastructure Management Framework
<b>IoT</b>	Internet of Things
<b>IP</b>	Internet Protocol
<b>ITU</b>	International Telecommunication Union
<b>KDU</b>	Kubernetes Deployment Unit
<b>KPI</b>	Key Performance Indicator
<b>LiFi</b>	Light Fidelity
<b>LLS</b>	Lower Layer Split
<b>LSTM</b>	Long Short-Term mMemory

<b>MBO</b>	Mobile Backhaul Orchestrator
<b>MCS</b>	Mission Critical Service
<b>MCPTT</b>	Mission Critical Push-To-Talk
<b>MEC</b>	Multi-access Edge Computing
<b>MIB</b>	Master Information Block
<b>ML</b>	Machine Learning
<b>mMTC</b>	massive Machine Type Communications
<b>MNO</b>	Mobile Network Operator
<b>NBI</b>	Northbound Interface
<b>Near-RT RIC</b>	Near Real Time RIC
<b>NF</b>	Network Function
<b>NS</b>	Network Service
<b>NSD</b>	Network Service Descriptor
<b>NSMF</b>	Network Slice Management Function
<b>NSSMF</b>	Network Slice Subnet Management Function
<b>NFL</b>	Network Function Layer
<b>NFV</b>	Network Function Virtualization
<b>NFVI</b>	Network Function Virtualization Infrastructure
<b>NN</b>	Neural Network
<b>Non-RT RIC</b>	Non Real Time RIC
<b>NPN</b>	Non Public Network
<b>NR</b>	New Radio
<b>NST</b>	Network Slice Template
<b>NT</b>	Network Telemetry
<b>NWDAF</b>	Network Data Analytics Function
<b>NW-TT</b>	Network Side Translator
<b>OAM</b>	Operations, Administration and Management
<b>ONAP</b>	Open Network Automation Platform
<b>OS</b>	Operative System
<b>OSM</b>	Open Source Mano
<b>PCC</b>	Policy Charging and Control
<b>PCF</b>	Policy Control Function
<b>PCP</b>	Priority Code Point
<b>PDU</b>	Protocol Data Unit
<b>PLC</b>	Programmable Logical Controller
<b>PNF</b>	Physical Network Function
<b>PRB</b>	Physical Resource Block
<b>PSF</b>	Physical Security Function
<b>PLMN</b>	Public Land Mobile Network
<b>PNI-NPN</b>	Public Network Integrated – Non Public Network
<b>PM</b>	Performance Monitoring
<b>PPDR</b>	Public Protection and Disaster Relief
<b>P4</b>	Programming Protocol-independent Packet Processors
<b>QoS</b>	Quality of Service
<b>RAN</b>	Radio Access Network
<b>RAT</b>	Radio Access Technology
<b>REST</b>	REpresentational State Transfer
<b>RIC</b>	Radio Intelligent Controller
<b>RIS</b>	Reconfigurable Intelligent Surface
<b>RL</b>	Reinforcement learning
<b>RNN</b>	Recurrent Neural Network
<b>RoE</b>	Radio Over Ethernet
<b>RRM</b>	Radio Resource Management
<b>RU</b>	Remote Unit

<b>SA</b>	Standalone
<b>SBA</b>	Service Based Architecture
<b>SDN</b>	Software Defined Networking
<b>SEAL</b>	Service Enabler Architecture Layer
<b>SIB</b>	System Information Block
<b>SIM</b>	Subscriber Identity Module
<b>SLA</b>	Service Level Agreement
<b>SME</b>	Small of Medium Enterprise
<b>SMF</b>	Session Management Function
<b>SMO</b>	Service Management and Orchestration
<b>SON</b>	Self Organizing Network
<b>TCP</b>	Transmission Control Protocol
<b>TNE</b>	Transport Network Equipment
<b>TSC</b>	Technical Steering Committee
<b>TSN</b>	Time Sensitive Networking
<b>UAV</b>	Unmanned Aerial Vehicle
<b>UDM</b>	Unified Data Management
<b>UDR</b>	Unified Data Repository
<b>UE</b>	User Equipment
<b>UP</b>	User Plane
<b>UPF</b>	User Plane Function
<b>UE</b>	User Equipment
<b>UML</b>	Unified Modeling Language
<b>URLLC</b>	Ultra Reliable Low Latency Communication
<b>V2X</b>	Vehicle to Everything
<b>VDU</b>	Virtual Deployment Units
<b>vEPC</b>	Virtualized Evolved Packet Core
<b>VM</b>	Virtual Machine
<b>VIM</b>	Virtual Infrastructure Manager
<b>VNF</b>	Virtual Network Function
<b>VoMS</b>	Vertical-oriented Monitoring System
<b>VNFD</b>	Virtual Network Function Descriptor
<b>VPN</b>	Virtual Private Network
<b>VSNF</b>	Vertical Service Management Function
<b>VSF</b>	Virtual Security Function
<b>WDM</b>	Wavelength Division Multiplexing

## 1 INTRODUCTION

---

The development of an affordable 5G system, which is the main objective of this project, is based on the combination of many building blocks that, together, conform a full functional structure. To achieve this, one of the most critical steps is the integration and testing of the components, which is the main goal of this deliverable.

To this end, the two sites are separately described: In Section 2, the integration and testing performed in Malaga platform is defined, while Section 3 describes the same content structure but regarding Castellolí platform. In each section, the architecture of the platforms is presented, which are updated versions of the ones included in D4.1 [2] and the integrations that have taken place for the specific platform are described in detail. Finally, each section describes the Timeplan readjustment and the milestones to be achieved before the end of the project.

Section 4 is divided by the individual test cases and the integration test cases. The first part is a detailed section that contains the test cases performed to validate each of the building blocks before being integrated in the platforms. This individual test cases also specify the test results in the same section.

The second part of section number 4 explains the test cases that took place to validate the integrations between components, once deployed in the platform or in a preliminary stage in some cases. The integrations test cases also gather the results obtained at the end of the section.

The following sections describe the specific evolution of each pilot, explaining the pilot building blocks, the integrations that have been found necessary and its deployment status.

Section 5 describes the Time-Sensitive Networking (TSN) over 5G Proof of Concept. It is divided in three main sections: Translation from TSN domain to 5G, prioritization over 5G and time synchronization.

Section 6 explains the SmartCity pilot, that intends to demonstrate the usage of a 5G private network, and the advances provided by Affordable5G in an emergency scenario occurring at an indoor environment.

Finally, Section 7 provides a detailed explanation of the pilot of Emergency Communications, which main objective is to integrate, deploy and demonstrate 3GPP-compliant MCS services on top of the Affordable5G private network.

## 2 INTEGRATION AND TESTING IN MALAGA CAMPUS

5G testbed Malaga Campus is being upgraded with the installation and deployment of enhanced products, developed prototypes and open platforms to carried out complete end to end (E2E) system tests.

### 2.1 Malaga Campus Architecture instantiation

The Affordable5G high level architecture instantiation in Malaga is depicted in Figure 1. This figure is an update of the one presented in D4.1 [2]. Main building blocks to be integrated within Malaga platform are highlighted in red. These are: the Open-Source Mano (OSM), in charge of the orchestration of the 5G network; the O-RAN, which is an integration of different components developed each one by a different partner and which takes care about the 5G radio access; and the 5G Core (5GC), which has been updated to include new functionalities to support pilot deployments. It is important to note that pilot deployments will be explained further in Sections 5, 6 and 7.

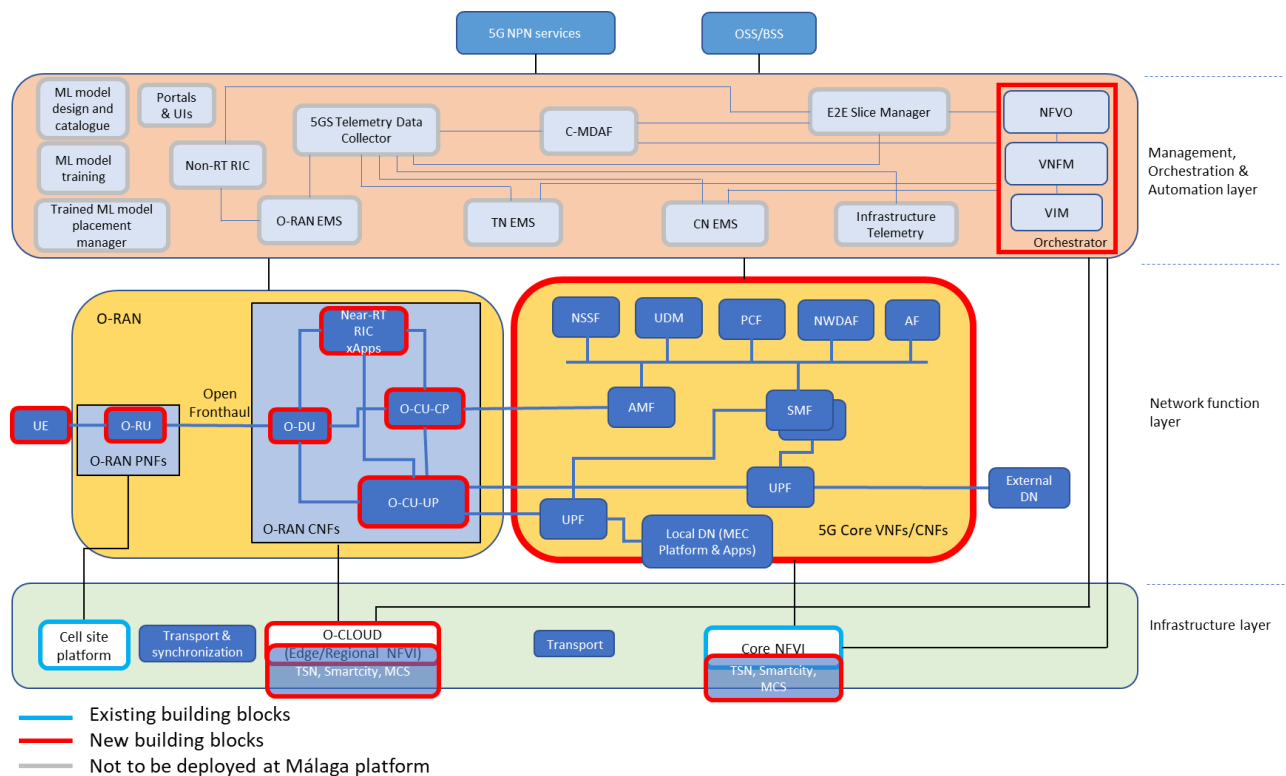


Figure 1 High Level architecture Malaga

In addition, an alternative O-RAN solution will be deployed in Málaga. This solution will allow us to test multivendor compatibility with the 5G core and the rest of the components of the 5G network, including the pilots deployed on the top. Thus, one the goals of the project can be tested, which is the interoperability of different equipment coming from different vendors leading to an affordable 5G network.

## 2.2 Malaga Campus Integrations

In this section, a detailed update of all integrations carried out at the Malaga platform is shown. Each subsection includes the explanation about the improvements in a specific component or building block, and their integrations with the platform. This also includes the management of the interfaces involved in this process. The building blocks to be described are the 5G Core, the OSM and the O-RAN.

### 2.2.1 5G Core Integrations

This section provides an overview of the Athonet 5G core installation, configuration, and test plan in Malaga (UMA) testbed, to provide the 5GC integration in the end-to-end (E2E) 5G infrastructure.

Two Virtual Machines (VMs) containing pre-configured 5GC functionalities were provided by Athonet and installed on the Malaga Network Functions Virtualization Infrastructure (NFVI). One VM contains a full 5GC Stand Alone (SA) instance (release 3.1 of Athonet's software), which acts as a main central core with complete control and user planes, whereas the second VM only contains a User Plain Function (UPF) and plays the role of a 5G edge node. This second UPF is installed on the same hardware of the first instance, but it is logically separated from the rest of the functionalities. This setting allows the creation of a second network slice with a separate dedicated user plane.

The provisioning and installation were performed remotely via Virtual Private Network (VPN) by dedicated Athonet support service.

The network plan was configured according to the testbed owner's requirements, in order to expose the 5GC interfaces as follows:

- N1, N2 between the Access and Mobility Management Function (AMF) and the gNB
- N3 between the UPF and the gNB
- N6 from the UPF to the Data Network (Internet)

At the moment of writing this document, a Nokia gNB is attached to the 5GC AMF; the attachment was verified with a set of validation tests, by checking the HTTP message traces and logs of the network elements, especially for the AMF.

No.	Time	Source	Destination	Protocol	Length	Info
42	3.373174792	10.182.32.32	10.182.32.41	SCTP	100	HEARTBEAT
43	3.373871696	10.182.32.41	10.182.32.32	SCTP	100	HEARTBEAT_ACK
167	9.261164612	10.182.32.32	10.182.32.41	SCTP	100	HEARTBEAT
168	9.261982942	10.182.32.41	10.182.32.32	SCTP	100	HEARTBEAT_ACK

```

<
> Frame 42: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 10.182.32.32, Dst: 10.182.32.41
▼ Stream Control Transmission Protocol, Src Port: 38412 (38412), Dst Port: 38412 (38412)
  Source port: 38412
  Destination port: 38412
  Verification tag: 0xaf3edc0
  [Association index: disabled (enable in preferences)]
  Checksum: 0x6bf60f04 [unverified]
  [Checksum Status: Unverified]
  > HEARTBEAT chunk (Information: 48 bytes)

```

Figure 2 Some HEARTBEAT messages captured by Tcpcdump during gNB and 5GC communication.

At the end of the project, the Nokia Radio Access Network (RAN) will be replaced with two O-RAN solutions, one of them directly coming from the Affordable5G project.



The central 5GC instance was provisioned with 10 Subscriber Identity Module (SIM) cards provided by Athonet; the associated User Equipments (UEs) were then successfully provisioned and attached to the core.

### 2.2.1.1 Integration with Accelleran O-CU (dRAX)

The integration of the Athonet 5GC with the O-CU (ACC dRAX) was performed in multiple stages, described below.

#### 2.2.1.1.1 First stage with Athonet ITP-EU & ACC Labs

A first preparatory stage of the 5GC's integration with the O-RAN solution developed by the project, was performed by connecting and testing the N1/N2 and N3 interfaces between the 5GC and Accelleran's gNB, obtaining the E2E IP connectivity. For this purpose, a cloud instance of the Athonet 5GC was utilized. The following section provides some details about this platform and its specific integration.

In order to perform these tests, Athonet provided an instance of AWS 5GC called Innovation Test Platform for European projects (ITP-EU). This cloud platform has been used for advanced tests in subsidized projects, like, in this case, Affordable5G.

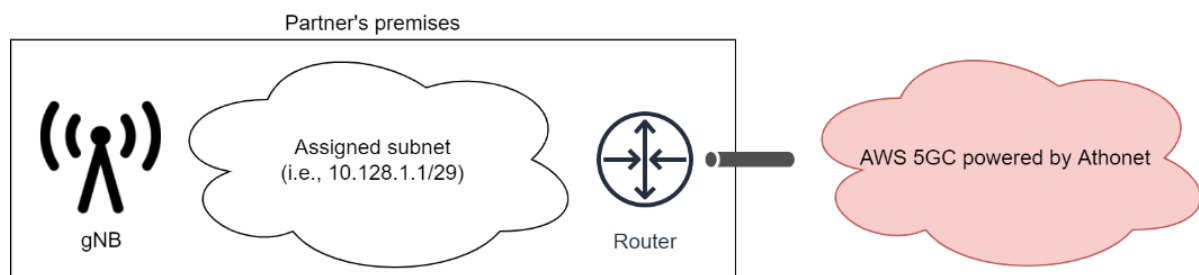


Figure 3 Illustration of the ITP-EU integration with partner's premises

The first integration test introduced above has been performed by Accelleran, to verify the connectivity of its gNB with the 5GC, testing UPF sessions and Internet traffic exchange. For that, a dedicated 5GC cloud instance has been set up on ITP-EU. The Athonet support team provided all the information regarding the PPTP VPN access to allow Accelleran to connect to the platform. Then, a specific subnet (e.g., /29) was assigned to give the proper IP address to the Accelleran's gNB.

Accelleran provided its SIMs information (i.e., IMSI, K and OPc) for the provisioning of the users into the 5GC Unified Data Management (UDM).

#### 2.2.1.1.2 Second stage for EuCNC Demo

A second stage of integration was performed in ACC labs in preparation for a demo video @ EuCNC 2022 in Grenoble. The video was positioned as a joint Affordable5G & FUDGE-5G demo (with Athonet participating in both projects), and its preparation was used as a second stage of integration.

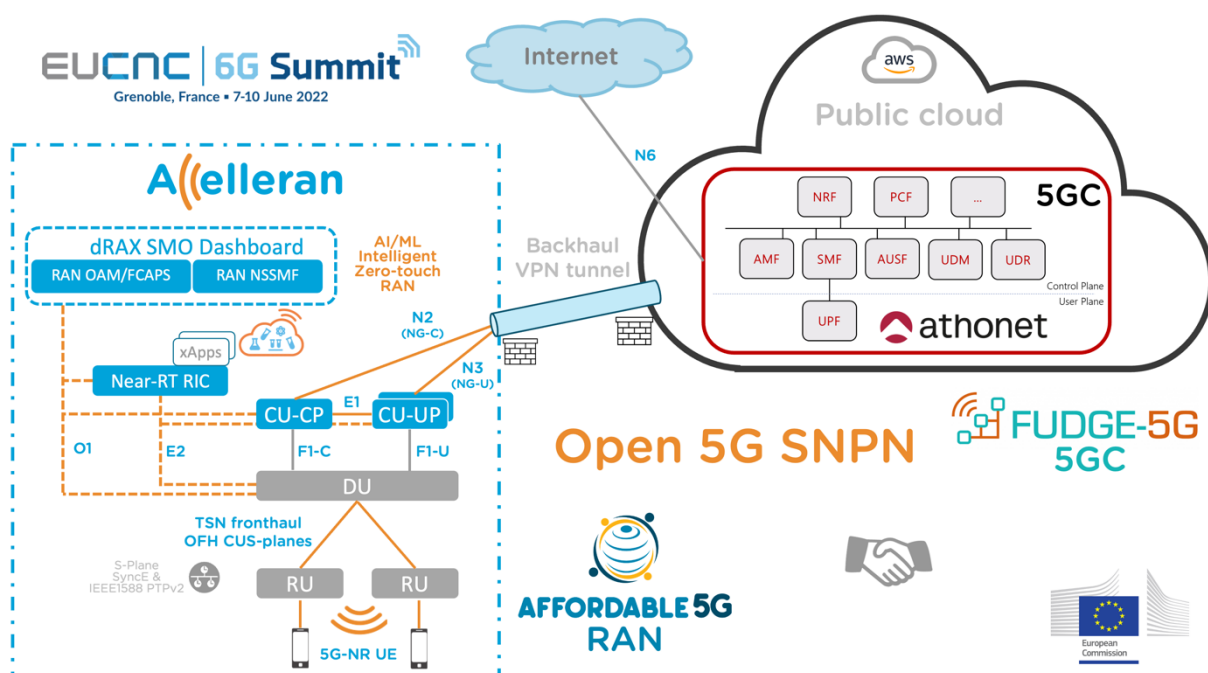


Figure 4: Affordable5G/FUDGE-5G Open 5GS demo @ EuCNC 2022

This demo was a full E2E integration of a 3GPP Release-15 (with Release-16 additions) Standalone 5G SNPN with an O-RAN aligned RAN. The UE was a commercial 5G smartphone. The DU and RU were external 3<sup>rd</sup> party network functions.

As part of the preparations for this video, the 5GC and disaggregated CU (CU-CP and CU-UP) N2/N3 interfaces were re-tested, with focus more on consistent RAN/5GC network slicing. Several different slice configurations were tested but for the demo, an eMBB slice was used to the 'default' UPF in the Athonet cloud, with default DN routed to the Internet.

The first screenshot (Figure 5) shows the internal configuration of the gNB with CU-CP creating the CU-UP instance and a DU attaching to CU, monitored by the RIC/SMO.

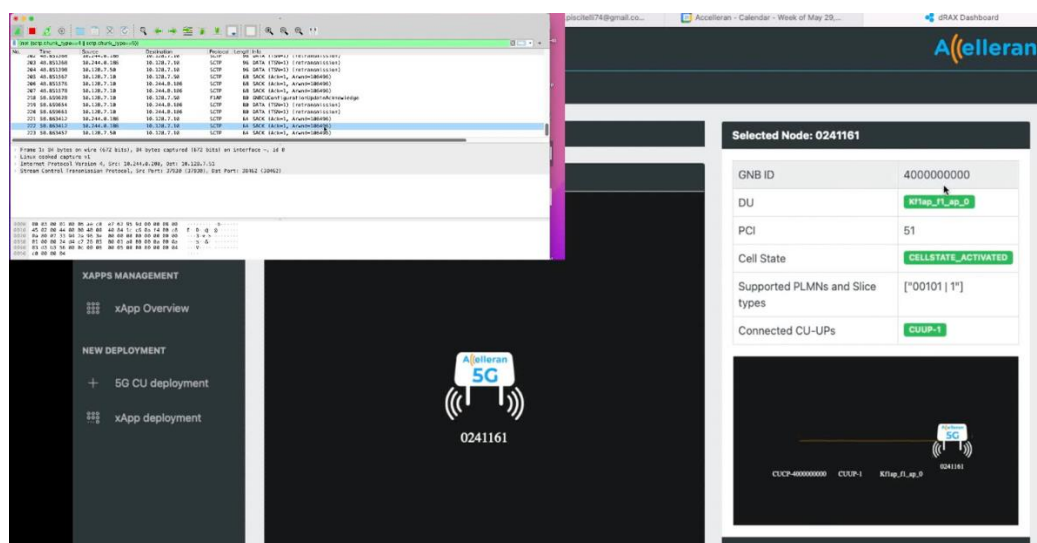


Figure 5: CU-CP creating CU-UP and DU attaching to demonstrate an active radiating gNB

When the UE (smartphone) is taken out of flight mode, it then attaches to the gNB and then 5GC, authenticating and then establishing an active Internet connection. Some of the resultant generated F1, N2 & N3 traffic is shown in Figure 6:

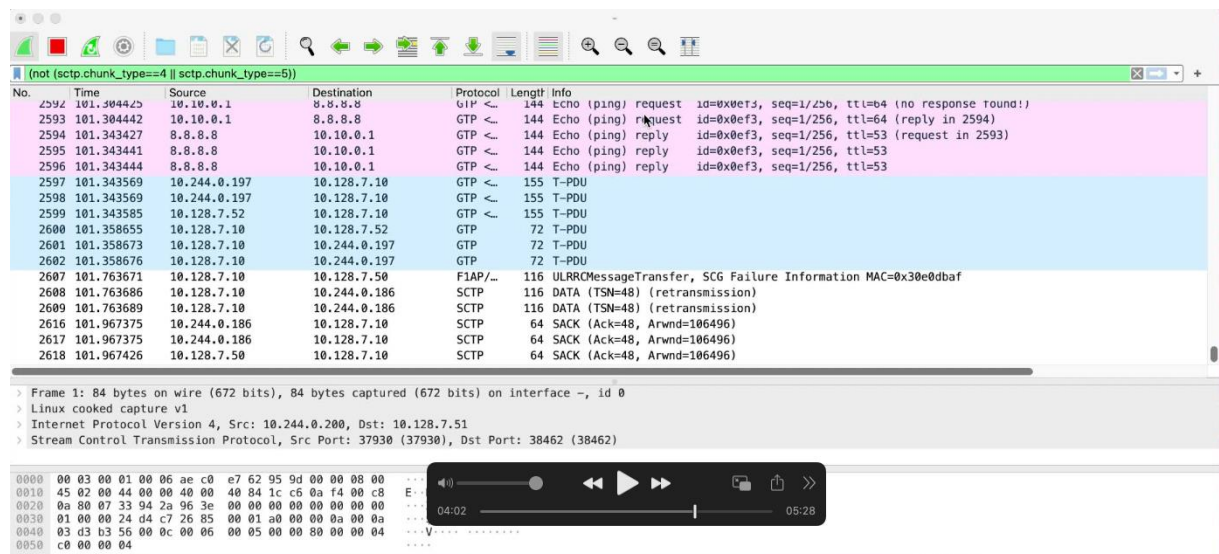


Figure 6: UE attaching to gNB and authenticating with 5GC to establish active connection

Finally, in Figure 7, the smartphone displays the Athonet tagged 5GS that it has connected to. An example IP service (in this case a speedtest) is then demonstrated. Note that delays and throughput reflect setups in the lab (with the VPN connection to the Athonet 5GC in the cloud and a public Internet speedtest server) and should not be used to infer achievable performance in Castellolí or Malaga (which depends on other factors like allocated 5G-NR spectrum bandwidth, UPF location, etc.).

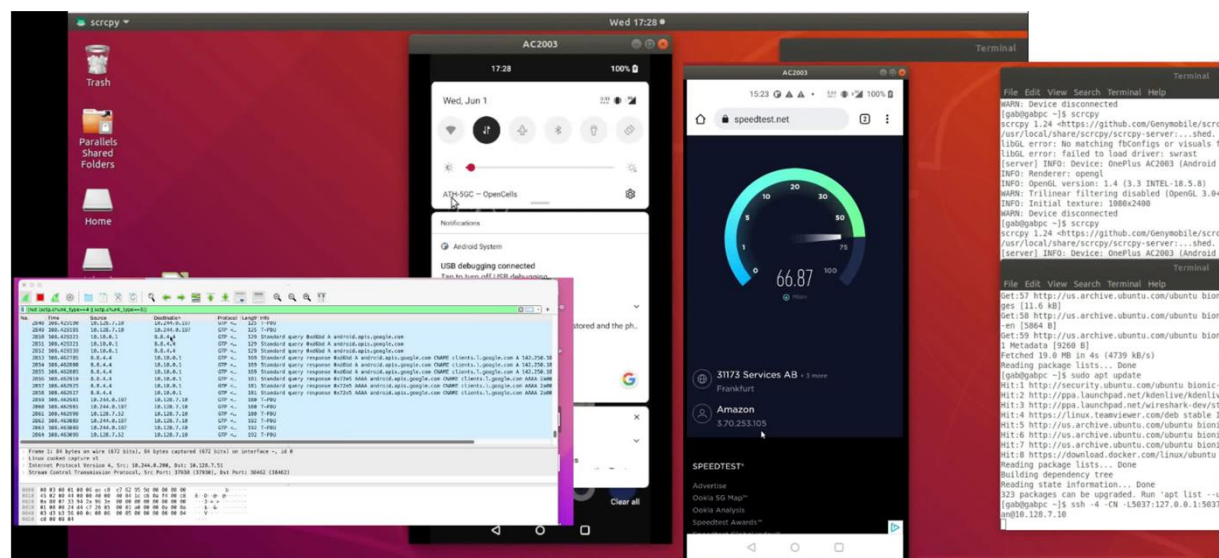


Figure 7: 5G smartphone connected to and running over the Affordable5G Open SNPN

The final integration will be then consolidated with the 5GC instance physically installed at Castellolí and UMA's site, focusing on further specific test cases.

### 2.2.2 OSM integrations

Martel has developed several software components that extend the current OSM functionality in two directions:

- **KNF Placement.** The ability to provide placement details for Kubernetes Network Functions (KNFs), in order to target specific nodes of a registered Kubernetes cluster. This is currently not possible in the official OSM releases, with the placement capabilities only limited to targeting entire clusters, with no way to target individual nodes.
- **Infrastructure as Code.** Presently OSM lacks built-in support for GitOps. Martel has developed OSM Ops, a distributed system to complement OSM's own deployment and operation tools with GitOps pipelines. The basic idea is to describe the state of an OSM deployment through version-controlled text files hosted in an online Git repository. Each file declares a desired instantiation and runtime configuration for some of the services in a specified OSM cluster. Collectively, the files at a given Git revision describe the deployment state of these services at a certain point in time. OSM Ops monitors the Git repository in order to automatically reconcile the desired deployment state with the actual live state of the OSM cluster.

#### KNF Placement

For the KNF placement functionality, the goal was to contribute this feature to OSM's main codebase, discussing the design and implementation details with the OSM Technical Steering Committee (TSC) for approval. The reasoning for this was not only to ensure that the functionality becomes part of an official OSM release, thus allowing for long term support for it, but also to provide a good contribution to OSM thanks to Affordable5G.

At this time, the feature's design was mostly finalized with help from the OSM's developers, however, implementation details were not fully discussed. This unfortunately due to slow communication with the OSM developers, and several back-and-forth discussions dealing with the issue of providing placement functionality for KNF descriptors without violating the vendor contract (e.g. not altering the descriptors directly, providing ways for vendors to specify whether or not to allow placement, etc.). As it stands, an implementation of the placement functionality is now possible, but only in the form of a custom lifecycle management module (LCM) that needs to be deployed in place of OSM's base LCM module (Release Ten).

Due to the current status of the KNF placement implementation functionality we opted against replacing the module in any installations of OSM that is being actively used for other tests. What we have is 3 separate VMs running in Malaga:

- A VM running OSM V.10 with the modified LCM module deployed, 8 cores, 8 GB memory, 100 GB disk space.
- A VM running a master node of a Kubernetes cluster, 2 cores, 4 GB memory, 40GB disk space.
- A VM running a master node of a Kubernetes cluster, 2 cores, 4 GB memory, 40GB disk space.

The OSM instance has the Kubernetes cluster registered through a dummy Virtual Infrastructure Manager (VIM), and a package containing a mission critical service for Pilot 2

ready to be deployed on it, with the ability to provide (at onboarding time) a list of labels that the target Kubernetes nodes must have in order for the KNF to be deployed on them.

Due to the nature of this placement feature, defining relevant Key Performance Indicators (KPIs) isn't an easy task. Metrics such as the resource usage of deployed services or their latency response aren't measuring performance in relation to the placement functionality introduced, but rather the performance of the OSM and Kubernetes themselves. Their performance in that regard doesn't relate to our work in that regard. This is a new functionality being introduced, and the advantages that come from it are mainly in the flexibility gained in placing services on the exact nodes where they are needed, and the ability to maintain consistency within OSM, which would be difficult to have if one was to start moving Kubernetes pods around outside of OSM.

A KPI we could use in this situation could be the average E2E deployment time for a given KNF on our setup. This to compare the time needed to deploy a KNF on a given node without the placement functionality (which requires manually moving the deployments/statefulsets on the Kubernetes cluster from one node to another) and with the placement functionality (providing the configuration directly at onboarding time from OSM). However, even in this case, there are several factors at play that could make the time vary by a large margin, as the complexity of the services (how many KDUs and corresponding Deployments/Statefulsets and pods are created) and how they are meant to be distributed across the nodes can impact the deployment times.

### Infrastructure as Code

The Infrastructure as Code solution was validated through a fully-fledged integration into the Malaga cluster. The Malaga rollout entailed deploying every OSM Ops component and then using the OSM Ops facilities to set up a GitOps pipeline to automatically deploy the Nemergent services. We outline deployment and test cases below, then conclude this section with a brief discussion of KPIs.

### Deployment

The OSM Ops deployment targets the virtual servers already set up for KNF Placement. OSM runs in its own virtual server (named "osm") and is configured with a VIM pointing to a Kubernetes cluster (MicroK8s distribution) made up of two nodes, named "node1" and "node2". The Kubernetes cluster hosts FluxCD's own Source Controller and the OSM Ops main service, both in the "flux-system" namespace. Source Controller monitors an OSM test repository on GitHub. The OSM Ops service connects both to Source Controller, within the same cluster, and to the OSM north-bound interface (NBI) running outside the Kubernetes cluster. The diagram below (Figure 8) illustrates the deployment.



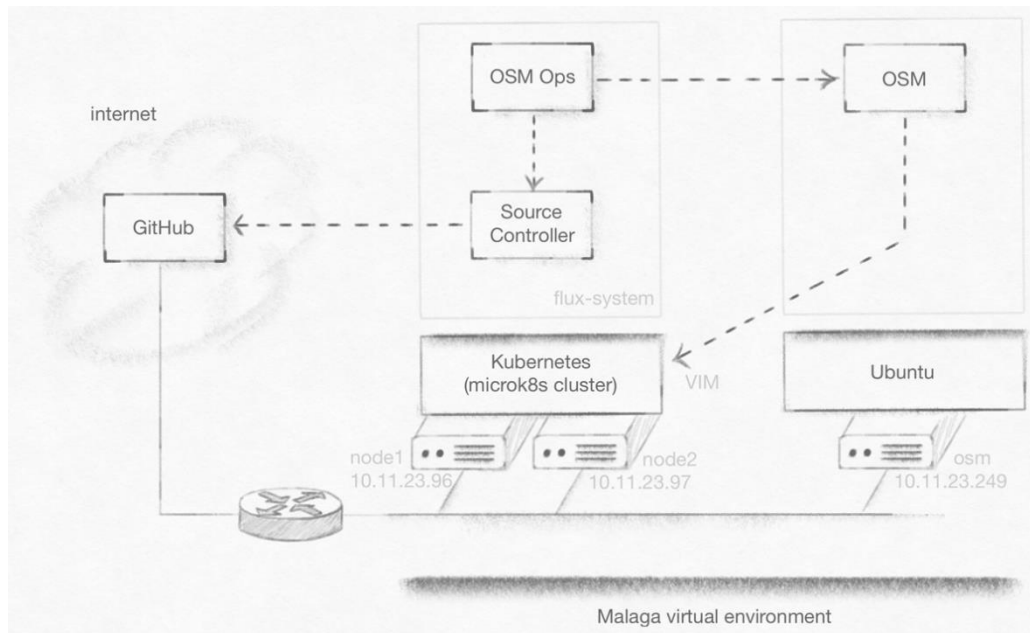


Figure 8 OSM Deployment

The OSM Ops deployment entailed several tasks. We first installed and configured the FluxCD CLI on node2, then deployed FluxCD and OSM Ops services to the Kubernetes cluster. Following that, we configured OSM Ops to be able to interact both with the Git demo repository through Source Controller and the OSM Northbound Interface (NBI). Finally, the OSM cluster had to be configured with a new repository containing the Nemergent Helm charts for the services to be deployed through the GitOps pipeline as well as suitable NSD and VNFD descriptors.

Note that, with the above setup, OSM creates Network Service (NS) instances using an Network Service Descriptor (NSD) named “affordable\_nsd” pointing to a Virtual Network Function Descriptor (VNFD) called “affordable\_vnfd”. The VNFD declares a Kubernetes Deployment Unit (KDU) named “nemergent” which references the Helm repository mentioned earlier.

## 2.2.3 O-RAN Integrations

### 2.2.3.1 Status of the O-DU - O-RU Integration in Eurecom Open5GLab

Eurecom Open5GLab network setup for the O-RAN 7.2 development is depicted in Figure 9 and it is composed of a FibroLAN switch that interconnects the O-DU server with different O-RAN compliant RUs including Mavenir, Foxconn and VVDN RUs. The network is synchronized using Qulsar PTP Grand Master that is either distributing the PTP to DU and RUs directly or via the FibroLAN switch.

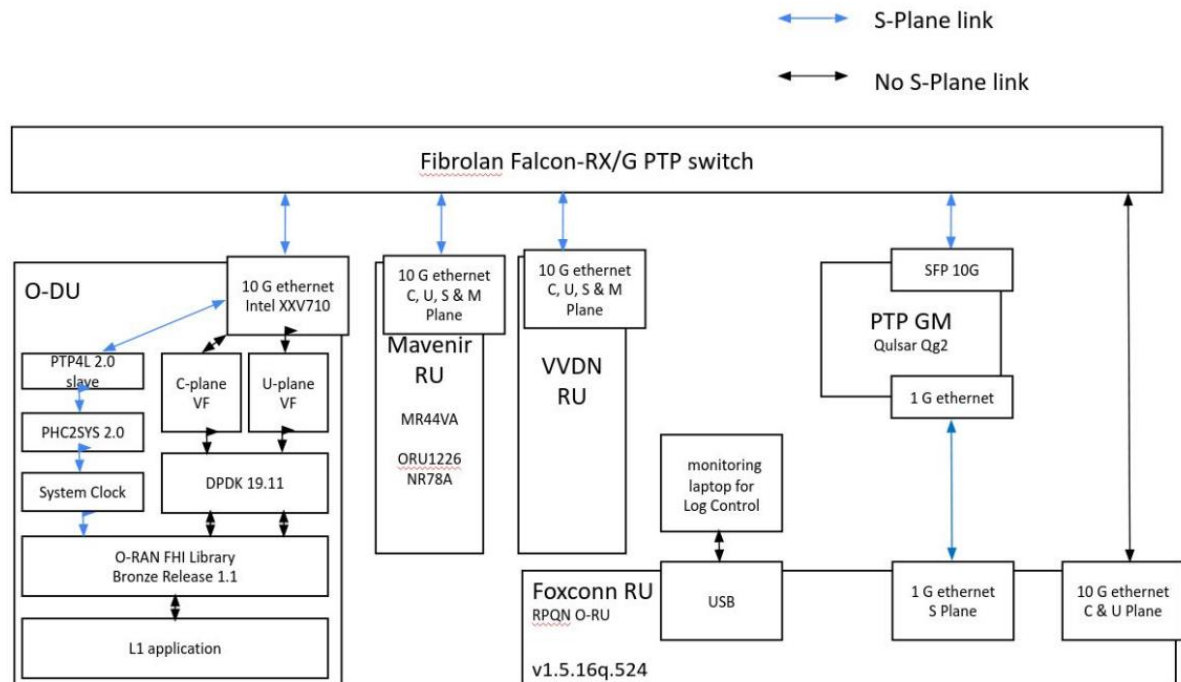


Figure 9: Network setup for O-RAN 7.2 in Eurecom

OAI O-DU integrates the O-RAN Front Haul Interface (FHI) software libraries, many steps to achieve the full integration with commercial O-RU have been validated:

- **S-Plane validation** both with:
  - Local master clock: OAI O-DU assuming grand master role and O-RAN sample app O-RU assuming the slave role.
  - Grand master in the network: PTP synchronization packets coming from the Qulsar Grand Master and OAI O-DU assuming the PTP slave role.
- **CP and UP validation** using both:
  - Testing OAI O-DU with respect to the O-RAN O-RU sample app.
  - Successful connection of OAI-DU to the Foxconn RU and proper exchange of O-RAN packets.

The connection to the commercial off-the-shelf (COTS) Foxconn RU required first, the validation of the S-plane for both the O-DU and O-RU, followed by the configuration of the Foxconn RU M-plane. A specific configuration of the network switch was also set to comply with the VLAN tagged packet of the CP and UP.

The wireshark capture in Figure 10 shows the successful connection of OAI O-DU to Foxconn RU using the O-RAN 7.2 interface. The UP packets flow in both directions, O-DU <-> Foxconn RU. The CP packets indicate the RU about the transmission of the UP packets.

Example: a CP message from OAI-DU will set up one section ID that spans from Physical Resource Block (PRB) 0 to 105, which indicates that all the subsequent UP packets will be transmitted over the selected PRB. Such CP may be sent at any time (e.g., when beamforming) to indicate the RU about the PRBs associated to the subsequent UP packets.

Time	Source	Destination	Protocol	Length	Info
1 0.000000000	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-C	60	C-Plane, Type: 1 (Most channels), Id: 0 (PRB: 0-105)
2 0.000000202	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-C	60	C-Plane, Type: 1 (Most channels), Id: 0 (PRB: 0-105)
3 0.000000265	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-C	60	C-Plane, Type: 1 (Most channels), Id: 0 (PRB: 0-105)
4 0.000000326	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-C	60	C-Plane, Type: 1 (Most channels), Id: 0 (PRB: 0-105)
5 0.000324230	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
6 0.000324305	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
7 0.000324380	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
8 0.000350927	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
9 0.000386758	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
10 0.000386845	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
11 0.000386918	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
12 0.000386988	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
13 0.000441717	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
14 0.000441784	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
15 0.000441847	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)

Time	Source	Destination	Protocol	Length	Info
1 0.000000000	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
2 0.000000063	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
3 0.000000127	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
4 0.000000190	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
5 0.000000253	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
6 0.000000314	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
7 0.000000379	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
8 0.000000441	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
9 0.000021731	6c:ad:ad:00:04:dc	66:44:33:22:11:00	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
10 0.000021791	6c:ad:ad:00:04:dc	66:44:33:22:11:00	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
11 0.000021855	6c:ad:ad:00:04:dc	66:44:33:22:11:00	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
12 0.000021918	6c:ad:ad:00:04:dc	66:44:33:22:11:00	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
13 0.000021981	6c:ad:ad:00:04:dc	66:44:33:22:11:00	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
14 0.000022044	6c:ad:ad:00:04:dc	66:44:33:22:11:00	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
15 0.000022108	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
16 0.000022170	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)

Figure 10: OAI O-DU integration with the Foxconn RU

Some additional steps are still required to achieve a full E2E solution able to allow COTS UE to connect to the 5G SA network using O-RAN FHI and they are mainly related to:

- Timing tuning for OAI O-DU threads with the aim to comply with O-RAN FHI library one. This will allow the precise O-RAN FHI buffer fill.
- Observation of the O-RU emitted spectrum and decoding of the principal cell signaling radio channels (e.g., Master Information Block (MIB) and System Information Block (SIBs))
- Integration of the PRACH procedures complying with the O-RAN FHI to enable the UE random access procedure for the connection.
- Connection with other commercial RUs like RunEL, Mavenir and VVDN.

### 2.2.3.2 Status of the Integration with RunEL RU

With the aim to help RUNEL to test the interoperability of their RU with a test DU, Eurecom delivered to RunEL a dedicated software that simulates the ORAN Interface protocol in March 2022 time frame. RunEL with Eurecom remote support successfully installed the Eurecom ORAN 7.2 Simulator SW and all other needed supporting SW tools on one of RunEL servers and the simulator is ready to start the initial integration between the DU and the RU. Eurecom will first finalize the integration of OAI-DU with the Foxconn RU, and then provide the software platforms to RUNEL.

Following several virtual meetings between Eurecom and RunEL the integration plan between Eurecom protocol stack (Core, CU and DU) and RunEL RU over ORAN PHY split interface Option 7.2 was created.

Following the completion of the initial ORAN integration, Eurecom will provide to RunEL the full stack of the 5G SA Core + CU + DU SW that will be also installed in the RunEL server and the E2E 5G standalone link that includes commercial 5G UE will be integrated and tested.



### 2.2.3.3 DU - CU Integration

As one of the goals of O-RAN is to support multi-vendor RAN solutions, for Accelleran, interoperability of the F1 interface between CU & DU is key, to allow the ACC dRAX (Near-RT RIC and disaggregated CU with CU-CP and CU-UP) to support multiple DU vendors and implementations.

As ACC have been integrated with multiple O-DUs (in fact, the OAI is the second integration and there are other integrations ongoing), ACC has created an integration specification and plan (primarily for F1 interface but also E2).

The integration plan is split into 4 phases of increasing complexity:

- **Phase 1: Basic F1 message exchange for setup, operation, and release:**
  - Defining a minimum configuration (1 UE per CU-CP, 1 cell, 1 RU with 1DU with 1CU, 1 IPv4 eMBB slice, 1 PDU Session)
  - Instantiate and configure 1 CU-CP with corresponding 1 CU-UP for the single slice + PDU Session.
  - Instantiate and configure DU with IP address of CU-CP.
  - Test F1-C Setup from DU to CU-CP & F1 Response.
  - gNB-CU Configuration Update and Acknowledge, to update DU with cells to be activated list.
  - Test F1 message exchange with UE initial connection (RRCSetup)
  - Test UE initial context setup.
  - Test RRC message transfer of NAS signalling.
  - Test PDU session setup (RRCReconfiguration) and F1-U setup of DU to CU-UP.
  - Test UE context release.
  - Test data transfer, UL & DL data transfer of 1 UE at full throughput.
- **Phase 2: Increasing UE attachment and data transfer:**
  - Increase number of UEs to connect, first 8 then 16.
  - Test data transfer, UL & DL data transfer of 8 then 16 UE at full throughput.
- **Phase 3: Specific feature testing of F1 & gNB states, RAN sharing, failure modes:**
  - Testing of QoS management (to setup QoS aware scheduling and GBR DRB configuration).
  - UE context modification, to add DRB to UE context (RRCReconfiguration)
  - QoS Admission control with (UE connect setup failure)
  - RAN sharing: gNB-CU configuration update with multiple PLMNlist, update of DU. Check broadcasted multiple PLMN by the Cells in SIB1.
  - F1 reset handling (both DU an CU initiated)
  - F1 error indication handling (both DU an CU initiated)
  - DU Netconf compliance testing and network management.
  - Other System Information (OSI) Support (SIB2 – SIB9), • Handling “gNB-CU System Information” containing “SIB type to Be Updated List”
  - Handling of Paging scheduling.
  - Testing of RRC Inactive (context release and re-setup)
  - Failure mode handling: F1 link failure, fronthaul link down, RU down, cell down
  - Handover testing.
  - E2 testing, E2 setup request/response.
  - E1 RIC subscription setup and delete, RIC indication.

- **Phase 4: Performance limit testing with flow control, increasing system capacity:**
  - Incrementally increasing number of Cells per DU, RUs per DU, DUs per CU-CP, CU-UP per CU-CP, UEs (so total throughput exceeds available bandwidth, retest at performance limits.
  - Test of PDU sessions of type IPv4, IPv6, IPv4v6.
  - Test of network slicing eMBB, URLLC, mMTC types.
  - Test of multiple PDU sessions per UE.
  - Test of multiple data radio bearers per UE.

These F1 (but not E2) tests have been performed between the OAI DU and ACC CU by installing an ACC CU in Eurecom labs. Additionally, all these tests have also been performed with the alternative O-DU being installed in Malaga.

## 2.3 Integration and Testing Timeline definition

The integration of all Infrastructure layer and Management layer elements in the two project sites involves many subtasks and dependencies. To better track the integration activity and identify possible bottlenecks, the consortium has developed and is maintaining a time plan in the form of a Gantt chart.

The time plan keeps track of all tasks, starting dates, durations, dependencies between tasks and involved partners. The network integration activity is tracked separately for the two project test sites (Castellolí and Málaga). Figure 11 shows the updated time plan for Málaga:

Taking as a reference the diagram already presented in D4.1 [2], the timeplan has suffered important deviations. This has made that final deadline in the diagram have been shifted to represent the real status. It is actually a blocking point that has delayed the full deployment of the 5G network at Malaga platform. The main issue is the integration between the RU-DU, which is a critical interface in O-RAN and whose new expected date for its fulfillment is early September.

UMA, as owner of Malaga platform, has acquired an alternative full O-RAN solution to be used in the meantime, which will be ready by early July and will allow us to continue with the rest of the tasks. In any case, once the issue is solved, the alternative solution will go to a second place, mainly to test multivendor interoperability, and the main O-RAN solution will be the one coming from Affordable5G project.

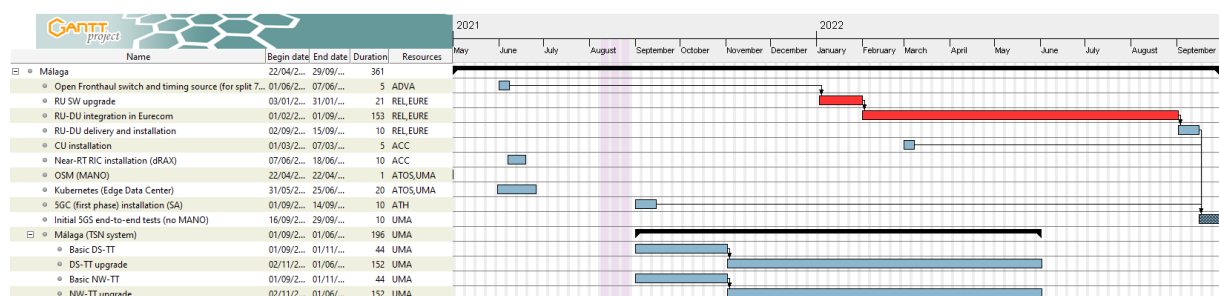


Figure 11. Malaga TimePlan

### 3 INTEGRATION AND TESTING IN CASTELLOLÍ

5G testbed circuit Castellolí is being upgraded with the installation and deployment of product releases, developed prototypes and open platforms to carry out complete E2E system tests.

#### 3.1 Castellolí's Architecture instantiation

The Affordable5G high level architecture instantiation in Castellolí is depicted in Figure 12. This figure presents an evolution of the architecture presented in D4.1 [2]. The building blocks to be integrated in Castellolí platform are highlighted in red. The building blocks that are already deployed are highlighted in blue.

The main building blocks that have been deployed in the Testbed are the NBYONE, in charge of the orchestration of the 5G network; the O-RAN, which will present important integration points of different components developed by the different partners, and finally, the 5G core, which has been updated to include new functionalities to support pilot developments. Note that pilot developments will be explained in another sections.

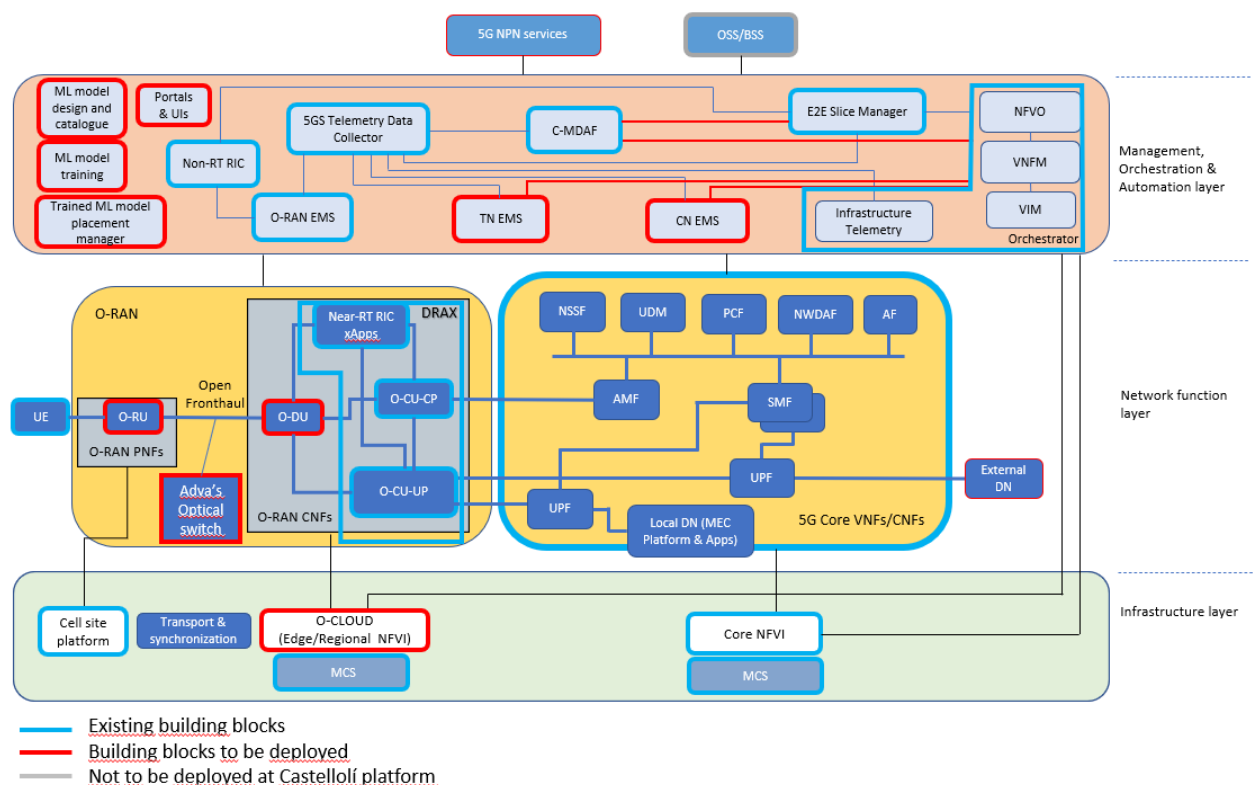


Figure 12 High Level architecture Castellolí

#### 3.2 Castellolí's Integrations

This section contains the detailed explanation of the integrations that have taken place in Castellolí platform. Each subsection describes the updates and detailed activities performed to succeed with the components' integrations in the platform.

### 3.2.15G Core Integrations

This section provides an overview of the Athonet 5G core installation, configuration and test plan in Castellolí testbed, to provide the 5GC integration in the end-to-end 5G infrastructure.

A full 5GC instance (upgraded to release 3.1) has been provided as a VM containing a pre-configured core, installed on the Castellolí NFVI.

The provisioning and installation have been performed remotely via VPN by dedicated Athonet support service. As in Malaga, the 5GC networking and licensing were configured and checked, in order to fully provide the required functionalities.

#### 3.2.1.1 Integration with the NBC Orchestrator

The Athonet 5GC exposes its Northbound APIs in order to be managed and monitored by an orchestrator. In this context, NearbyOne orchestrator (NBC) will leverage this set of APIs to retrieve some 5G system information. In particular, the focus is on the 5GC KPIs, that can be retrieved by the NearbyOne orchestrator to provide a single point of monitoring. More information about this integration is provided in Section 3.2.2.4.

#### 3.2.1.2 5GC KPI retrieving for NWDAF

A telemetry component is expected to be integrated with the 5GC. In this respect, a NWDAF instance is currently deployed in Castellolí testbed, with the purpose to connect to the 5GC and retrieve some relevant KPIs of its functional elements (e.g., number of sessions, number of UEs, traffic throughput, etc.)

The 5GC is equipped with a Prometheus component, which allows KPI shipping towards another Prometheus instance or another compatible monitoring tool, through a Remote Write functionality. The Remote Write was enabled to the 5GC Prometheus instance by setting the following HTTP URI:

`http:<PROM_IP:PORT>/api/v1/write`

where PROM\_IP and PORT are the endpoints of the receiving Prometheus instance, that from which the NWDAF retrieves some KPIs.

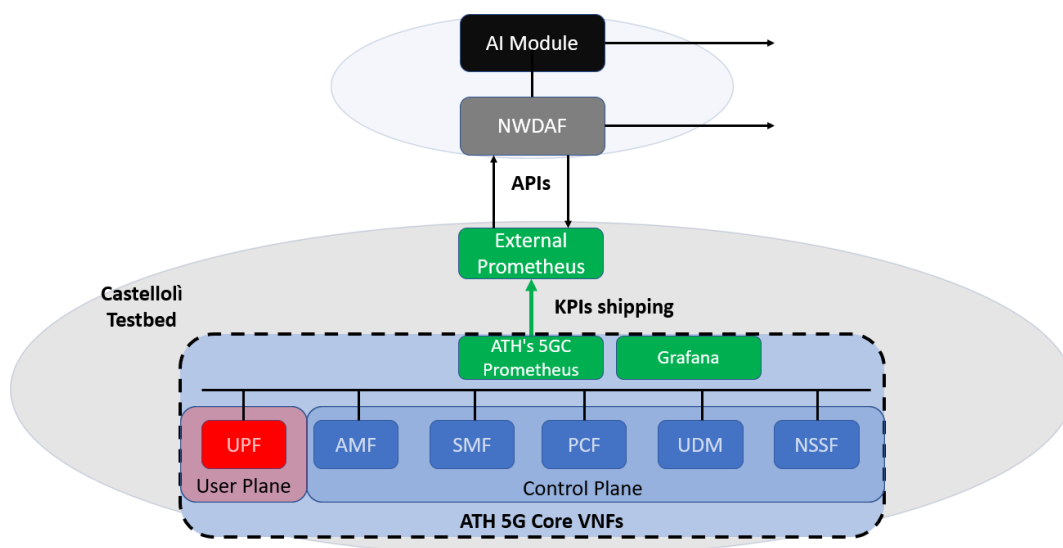


Figure 13 Architecture of the 5G core and NWDAF integration.

### 3.2.2 Orchestrator Integrations

NearbyOne as the E2E orchestrator in Castellolí, has integrated with different layers of components involved in the Affordable5G pilot in Castellolí, ranging from the baremetal/network provisioning to the integration with VNFs/Apps.

NearbyOne orchestrator is containerized, and can be deployed via a helm chart and has support to be installed at different managed and unmanaged k8s flavours e.g.: vanilla k8s, Redhat openshift, OKD, Rancher RKE, AWS EKS, Azure AKS, ARO, VMWare Tanzu, ...

In Castellolí, we have deployed NearbyOne on an RKE k8s cluster, running in 3 VMs deployed on one of the Castellolí paddock's servers.

#### 3.2.2.1 Provisioner

For the integrations between NearbyOne's baremetal provisioner and the specific Lenovo servers Cellnex has chosen for Castellolí, we have developed our interfaces supporting the Redfish standard (RDFSH) [23] ("DMTF's Redfish® is a suite of specifications that deliver an industry standard protocol providing a RESTful interface for the management of servers, storage, networking, and converged infrastructure.")

The standard is supported by most hardware vendors (WKRD) [24] (Advantech, Dell, Fujitsu, HPE, IBM, Lenovo, Supermicro and Cisco) but specific features required for the dynamic provision the nodes are left out of the standard and required some extensions e.g., mechanisms for redirecting operator interfaces such as serial console or virtual media are not included in the standard, as these can't be reasonably implemented as RESTful interfaces.

#### 3.2.2.2 i2CAT Slice Manager

NearbyOne orchestrator has integrated with the RESTful interfaces supported by i2CAT SliceManager to handle the RAN chunks of the slices. In the following lines we describe the current workflow related to slice provisioning among the Slice Manager, the non-rt-RIC and NearbyOne orchestrator.

1. The Slice Manager creates a user and register the RAN infrastructure in the slice manager.
2. The Slice Manager returns all available RAN infrastructure for the user.
3. The Slice Manager returns the configured RAN infrastructure.
4. The Slice Manager receives the request for deploying a slice and its 5G Core configuration.
5. The Slice Manager returns information regarding the activation of the RAN slice and of the slice deployed.

#### 3.2.2.3 Accelleran dRax

In Castellolí's pilot, the integration between NearbyOne orchestrator and Accelleran dRax includes its lifecycle management. Accelleran provides the helm chart for deploying dRax as a containerized application. NBC has exported Accelleran's helm chart as a block in NearbyOne. This block can be easily deployed from our dashboard to our provisioned k8s edge clusters, where also its placement/migration and lifecycle management policies can be defined.

Notice that all the interactions between the orchestrator and dRax go through i2CAT Slice Manager and its NonRT-RIC. NearbyOne orchestrator delegates the RAN chunks of the slices to the SliceManager, and these are the components interfacing with dRAX.

#### 3.2.2.4 Athonet 5G Core

In Castellolí's pilot, Athonet provided its 5G Core already pre-installed in a server and managed by themselves. In this scenario, the integration between NearbyOne orchestrator and Athonet's 5G Core skips the lifecycle management of the Core and focuses on the integration with its monitoring interfaces.

The purpose is to expose Athonet's 5G Core KPIs to the NearbyOne orchestrator, to let the orchestrator act as a single point of monitoring, in order to demonstrate the "monitoring management" capabilities provided by our solution.

This integration was built using the Prometheus server Athonet 5G Core exposes and configuring in NearbyOne the Prometheus federation to give the orchestrator and external consumers (e.g., the ML/AI engines from ATOS/i2cat/NKUA) a single point of monitoring.

#### 3.2.2.5 Nemergent MCS App

Similarly, to the above Accelleran dRax integration, the integration between NearbyOne orchestrator and Nemergent MCS App includes its lifecycle management. Nemergent provides the helm chart deploying the MCS as a containerized application. NBC has exported Nemergent helm chart as a Block in NearbyOne. This block can be easily deployed from our dashboard to our provisioned k8s edge clusters, where also its placement/migration and lifecycle management policies can be defined.

Unlike with the previous dRAX CNF, in this case the block will make use of the different KPIs provided by the orchestrated components and NWDAF to migrate/scale/heal the MCS application on different scenarios.

### 3.2.3 SMO and ORAN integration

This section focuses on the integration between the RIC manager, which corresponds to the implementation of the non-RT RIC in Affordable 5G and is part of the SMO layer, and the dRAX at the RAN, which includes the near-RT RIC. The description of this integration is presented in the first sub-section. Then, as a specific capability of this integration, the second sub-section describes an xApp that can be deployed in the dRAX via the RIC manager. The test cases for this integration will be presented in section 4.

#### 3.2.3.1 Integration of RIC manager and dRAX

As discussed in Deliverable D3.2 [3], the implementation of the non-RT RIC in Affordable5G is realized through a software module, referred to as RIC manager, which offers an ORAN compliant A1 interface and can control different types of near-RT RIC implementations regardless of whether they support or not the A1 interface. Moreover, it expands the definition of the ORAN non-RT RIC to include additional functions such as xApp discovery or onboarding. Then, following the architecture of the RIC manager presented in deliverable D3.2 (Figure 14), this deliverable covers the integration of this module with the new version 4.0 of Accelleran's dRAX, also known as dRAX 5G. The features of dRAX 4.0 are very similar to the ones present

in dRAX 2.1 that were considered in D3.2, but there are some additions that allow us to extend the integration with RIC Manager.

RIC Manager's API (Figure 15) still offers the same endpoints, which follow the O-RAN standard, allowing to seamlessly work with both dRAX's 4.0 and 2.1 versions as well as with O-RAN standard implementations of Near-RT and Non-RT RICs.

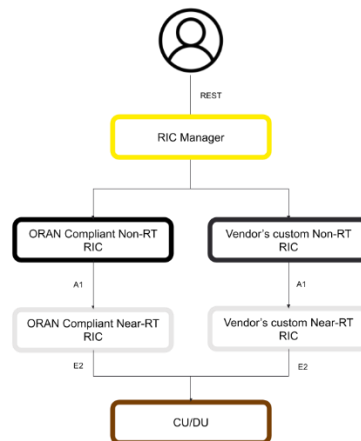


Figure 14 RIC manager proposed architecture

Registration		^
POST	/registration/nonrt Register Non Rt	✓
POST	/registration/nearrt/{nonrt_ric_name} Register Near Rt	✓
Non-RT		^
GET	/nrt_rics Get Neart Rics	✓
GET	/configuration/{nonrt_ric_name} Get Nonrt Config	✓
PUT	/configuration/{nonrt_ric_name} Put Nonrt Config	✓
xApp Deployment		^
GET	/xapps List All Xapps	✓
GET	/xapps/{nrt_ric_name} List Xapps	✓
POST	/deploy/{nrt_ric_name} Deploy Xapp	✓
DELETE	/deploy/{nrt_ric_name}/{xapp_name} Remove Xapp	✓
GET	/xapp_catalogue List Catalogue	✓
Policy		^
POST	/policy_type/{nrt_ric_name} Create Policy Type	✓
DELETE	/policy_type/{nrt_ric_name}/{policy_type_id} Delete Policy Type	✓
POST	/policy_instance Create Policy Instance	✓

Figure 15 Endpoints of RIC Manager's API



### 3.2.3.2 Telemetry xApp for dRAX 5G

As described in Deliverable D3.2 [3], the RIC Manager allows to deploy xApps in different Near-RT RICs, providing the specific implementation for each vendor but offering the same functionalities.

One of the xApps that we have developed extracts data and telemetry from the Near-RT RIC and exports it to a selected data lake. For dRAX 2.1 the only supported data lake is AWS S3, but for version 4.0 we are able to expose the xApp's ports outside of the K8s cluster, allowing us to integrate the Telemetry xApp with Prometheus. We have developed a Prometheus exporter that runs alongside the xApp, which is constantly exposing 4G and 5G telemetry to a Prometheus server.

The exporter can be configured through a new A1 policy (Figure 16) exclusive for dRAX 5G, which allows to toggle the exporter as well as select which specific metrics are to be exposed. It also supports all the configuration present for dRAX 2.1 Telemetry xApp.

```
{
  "name": "telemetry-xapp_policy",
  "description": "Policy type for dRAX's 5G telemetry-xapp",
  "policy_type_id": 20,
  "create_schema": {
    "$schema": "http://json-schema.org/draft-07/schema#",
    "title": "telemetry-xapp_policy",
    "description": "Policy type for dRAX's 5G telemetry-xapp",
    "type": "object",
    "properties": {
      "telemetry_topics": {
        "type": "string"
      },
      "update_interval": {
        "type": "integer"
      },
      "aws_access_key_id": {
        "type": "string"
      },
      "aws_secret_access_key": {
        "type": "string"
      },
      "aws_rest_api_key": {
        "type": "string"
      },
      "aws_rest_api_url": {
        "type": "string"
      },
      "prometheus_exporter": {
        "type": "boolean"
      }
    },
    "additionalProperties": false
  }
}
```

Figure 16 dRAX 5G A1 Policy Type



dRAX 4.0 also supports new 5G telemetry that can be exported via xAPP:

- UE Measurements
  - RSRP
  - RSRQ
  - SINR
- RRC Stats
  - RRC Attempted Connections
  - RRC Successful Connections
- CU-UP Throughput
  - Downlink Throughput [bps]
  - Uplink Throughput [bps]

A set of tests to validate the use of dRAX5G and the deployment of the telemetry xApp using RIC manager have been defined. These tests will be presented in section 4.2.7.

### 3.2.4 O-RAN Fronthaul & Synchronization Integrations

For O-RAN fronthaul infrastructure at Castellolí site, ADVA provides Transport Network Equipment (TNE) components. TNE ensures fibre connectivity and synchronization between O-RU and O-DU. The following dedicated hardware components have been shipped to the site: OSA-5401 and FSP 150 XG-118. The proposed fibre connectivity and timing distribution path is depicted below in Figure 17. In O-RAN fronthaul terminology, this configuration is known as LLS-C3 (Low Level Split, Configuration 3).

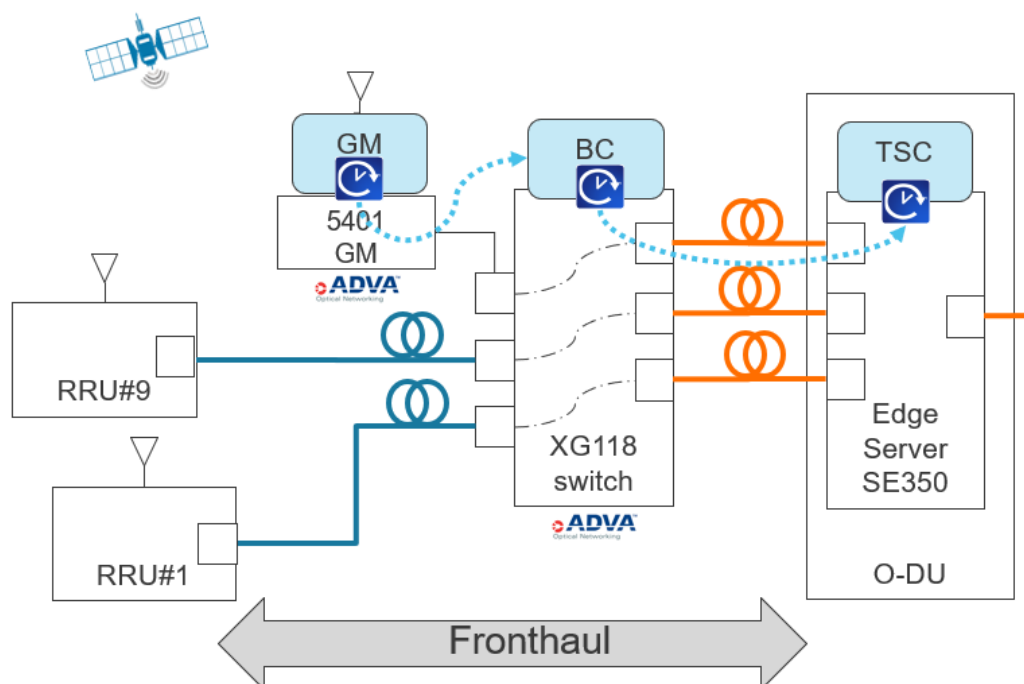


Figure 17 Castellolí Fronthaul infrastructure (LLS-C3)

The solid blue line is representing single-mode fibre, suitable for a distance up to 5~30 km, and orange line is multi-mode fibre (up to 300 m).

Prior to the shipment, we have assessed the quality of the synchronization at COTS Edge Server and reported it in Deliverable D2.2 [4] Chapter 5.2. To make on-site integration easier,

the equipment had been pre-configured. The configuration has been reviewed at face-to-face meeting.

### 3.3 Integration and Testing Timeline definition

The integration of all Infrastructure layer and Management layer elements in the two project sites involves many subtasks and dependencies. To better track the integration activity and identify possible bottlenecks, the consortium has developed and is maintaining a time plan in the form of a Gantt chart.

The time plan keeps track of all tasks, starting dates, durations, dependencies between tasks and involved partners. The network integration activity is tracked separately for the two project test sites (Castellolí and Málaga).

Figure 18 shows the updated time plan for Castellolí : Taking as a reference the diagram already presented in D4.1 [2], the timeplan changed significantly, presenting different deviations.

One of the issues that had to be treated was that the RU-DU integration presented significant delays that made impossible to assume that the equipment would be available for Affordable 5G's pilot demonstration in Castellolí's platform. Cellnex, the partner owning the platform, decided to purchase O-RAN equipment that can fulfill the needs of the project. This purchase presented a huge delay of the supply chain because of Covid19 and is expected to arrive the second week of August. The integrations related to RU and DU in Castellolí will start once the equipment arrives.

With that exception, all partners have been working hard in fulfilling the initial timeplan. There were some short delays in the 5G Core installation due to access problems in the platform and limited hardware resources, that was solved once the requirements were updated.

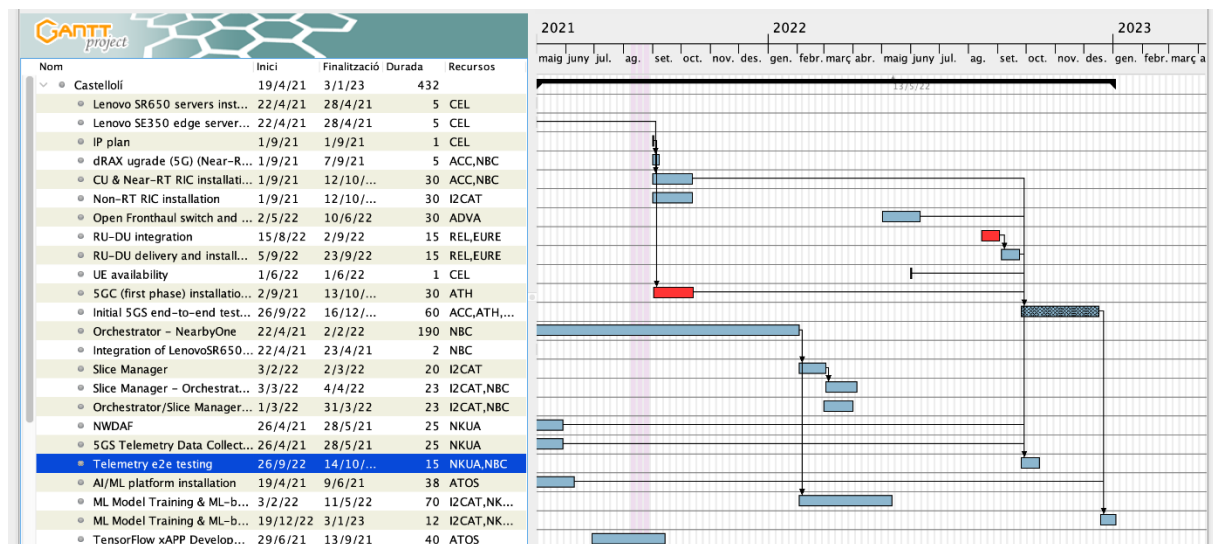


Figure 18 Castellolí Timeplan

## 4 TEST CASE DEFINITION AND KPI MEASUREMENTS

This section contains the detailed definition of the different Test cases and KPI Measurements by building block in both platforms used in Affordable 5G. The Test cases are structured in individual test cases and pre-integrations performed between the main components. The system level test cases will be detailed in the next deliverable D4.3, as well as the pilots validation.

The testing of individual components and pre-integrations testing phases are an important part for the future tasks that are approaching. The purpose of the individual test phase is precisely to minimize the margin of failure of each component, just as the test phase between different components, that not only proves that the individual tests are successful, but also demonstrates that the interaction between modules results in a successful outcome.

The nomenclature used to define each Test Case identifier is composed by a first set of letters that differentiates between Individual Test Case (Ind) and Integration Test Case (Int). These letters are followed by -test- and an enumeration of the component and the number of test cases defined by that specific component or Test Cases defined for the integration of several components. In that form, each component and set of component has a first enumeration, and the amount of Test Cases defined for it or them, has another enumeration, divided by "- ". An Example would be: Ind-test-01-01 (The first individual test case of the first component).

### 4.1 Individual Test Cases

This subsection contains the Individual Test Cases, defined to test and monitor the correct performance of the element or building block before its integration with the other elements conforming the architecture of the platform. There are some components that are used (instantiated) in both Malaga and Castellolí platforms such as the 5GCore.

The numerical KPIs are detailed in the section 4.1.2. This table presents in a joined-up way the different values that determine that the Test Cases have been fulfilled. Afterwards, results are provided for the cases that such an evaluation was made possible.

#### 4.1.1 OSM KNF Placement Test

The following test case aims to test out the KNF placement functionality enhancement for OSM, by fully deploying a service in the form of a NS/KNF both with and without using the new placement options at onboarding time. This to compare how quickly one can place the services in the nodes they need to be in versus the case where this functionality is not directly provided by OSM.

Test case name	OSM KNF Placement Test	Test Case id	Ind-test-01-01
Test purpose	Test that the placement functionality added to OSM allows for KNFs to be deployed on a specific node in a cluster at the same speed or faster than they would be without the placement functionality.		
Configuration	3 VMs: 1 running OSM, 2 VMs for a two nodes K8s cluster		
Test tool	OSM UI/Client		
KPI	Time for service to be deployed and functional		
Components Involvement	All 5GS components.		

Pre-test conditions	OSM up and running K8s Cluster up and running, registered to OSM Charts and KNFs/NSs for service registered to OSM	
Test sequence	Step 1	Manually onboard a KNF without the usage of the placement configuration through the OSM UI. Manually adjust the Kubernetes deployments on the cluster to place in a specific node (using the Kubernetes client
	Step 2	Record the time it took from beginning the onboarding to final placement.
	Step 3	Repeat the onboarding but providing placement configuration to OSM at the start.
	Step 4	Record the time it took from beginning the onboarding to final placement and compare with the time without the OSM placement functionality.
Test Verdict	Has the placement functionality provided an overall quicker deployment of the service on a specific node on average?	
Additional Resources	None	

#### 4.1.2 Slice Manager

The Slice Manager will oversee creating RAN slice subnets and sending an acknowledge to the Orchestrator informing that the radio chunks have been created successfully. Additionally, the Slice Manager will communicate to the Orchestrator the topology and characteristics of the current infrastructure. With that information, the Orchestrator will be able to choose which parameters it needs to deploy and then send the slicing creation requests so that the Slice Manager can create such RAN slice subnets. To confirm the creation of the slice subnet, an acknowledge code is sent to the Orchestrator using HTTP Methods. Finally, the time needed for the creation of the slices was computed under laboratory conditions.

<b>Test case name</b>	Unitary Test for the Slice Manager regarding the slice provisioning and radio service activation in 5G.	<b>Test Case id</b>	<b>Ind-test-02-01</b>
Test purpose	Assess the correct functioning of the atomic operations of the slice manager required for the slice provisioning and radio service activation in 5G.		
Configuration	Slice's User created, as well as the radio chunk and radio service activation,		
Test tool	Mongo DB, HTTP request, JSON.		
KPI	HTTP response codes, average initialization time.  The average radio service initialization time is: 3.184184  This time is for reference only as it was tested under laboratory conditions, and it must not be taken as final.  As an example of the timed test, the following is one of the logs used:		

	<pre> 2022-06-27 10:25:46 INFO clients.ran_controller {'id': '0be9f851-b995-45eb-90aa-d5a963c8d93a'} 2022-06-27 10:25:46 INFO business.ran_slice_subnet RESPONSE: 2022-06-27 10:25:46 INFO business.ran_slice_subnet {'id': '0be9f851-b995-45eb-90aa-d5a963c8d93a'} 2022-06-27 10:25:46 INFO business.ran_slice_subnet [] 2022-06-27 10:25:46 INFO business.ran_slice_subnet [] Code execution time in seconds is 1.252903938293457 code elapsed time in milliseconds is 1252.903938293457 127.0.0.1 - - [27/Jun/2022 10:25:46] "POST /api/v1.0/ran_infrastructure/62b975fe98f4f90ea32f7d0f/ran_slice_subnet HTTP/1.1" 200 - </pre>	
Components Involvement	5G components involved in the slice provisioning, mainly slice-manager, orchestrator, Non-RT-RIC.	
Pre-test conditions	All required developments, including communication interfaces are ready for the slice manager, orchestrator and Non-RT RIC.	
Test sequence	Step 1	If it is not already done, create a new user in the slice manager and register a ran infrastructure. Verify that indeed, a user and ran infrastructure is created and available in the slice manager DB.
	Step 2	Send http request for the reservation of radio resources. Verify that a radio chunk is created, and its status can be obtained through the proper http endpoints. Status verification should consider Non-RT-RIC and 5G antenna health status.
	Step 3	Initiate radio service.
Test Verdict	If the workflow for slice provisioning is correctly executed by the slice manager in terms of expected inputs and outputs, and radio service is active and the HTTP Response code is correct.	
Additional Resources	Example of keywords indicating the correct execution of the workflow for slice provisioning. <ul style="list-style-type: none"> <li>• HTTP response code: 200,202</li> <li>• Following fields are available through the proper http endpoints: user_id, physicalCellId, physicalInterfaceList, ran_controller_id, radio_service_id, radio_chunk_id.</li> </ul>	

#### 4.1.3 AI-ML

The ever-increasing demand on control loops in mobile network architectures and the current trends in MLOps clearly requires dedicated architectural blocks to facilitate and orchestrate all the different Machine Learning (ML) operations, such as training, evaluation and execution of the algorithms at every network level. Therefore, in the context of the Affordable5G project we have developed an AI/ML Framework based on open libraries and standards, including a set of interfaces for its integration with different network components of the 5G architecture. The AI/ML Framework developed in the context of the project is based on TensorFlow, a widely adopted open source set of libraries for numerical computation and machine learning. The architecture of the AI/ML framework is depicted in Figure 19.

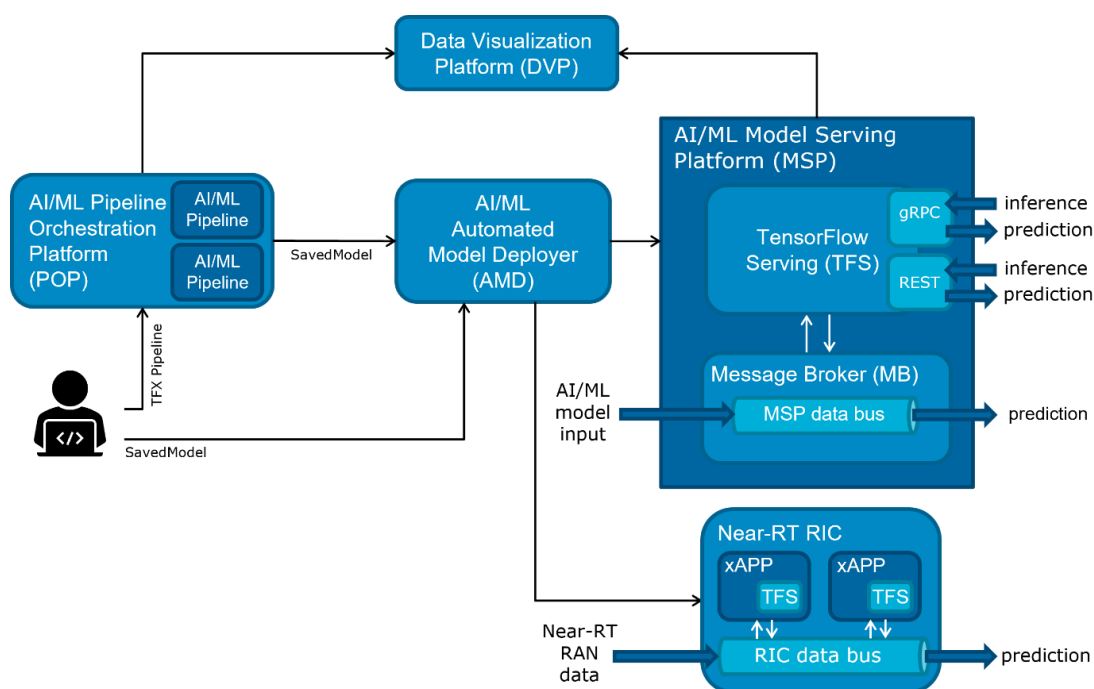


Figure 19 Affordable5G AI/ML Framework architecture

This subsection aims to assess the functional verification of the individual components that conforms the Affordable5G AI/ML Framework. This functional verification has been assessed in 4 test cases (Ind-test-03-01, Ind-test-03-02, Ind-test-03-03, Ind-test-03-04) covering the AI/ML Pipeline Orchestration Platform (POP), the AI/ML Automated Model Deployer (AMD) and the AI/ML Model Serving Platform (MSP), the essential building blocks of the architecture. In addition, this subsection also includes a performance evaluation of the MSP in the last test case (Ind-test-03-05) that will help to verify if the prediction latency of the MSP fulfills the requirements for the different ORAN control loops (Real-Time [ $<10\text{ms}$ ], Near-Real-Time [ $>10\text{ms}$  y  $<1\text{s}$ ], Non-Real-Time [ $>1\text{s}$ ]).

#### 4.1.3.1 Upload a ML pipeline to POP

The purpose of this individual test case is to demonstrate the proper deployment of the POP component of the AI/ML Framework including the onboarding of a toy AI/ML pipeline (chicago\_taxi pipeline provided in official TensorFlow documentation [TFX]) into the system.

Test case name	Upload a ML pipeline	Test Case id	Ind-test-03-01
Test purpose	Test the ML pipeline onboarding process of the POP		
Configuration	POP installed and running		
Test tool	POP (Airflow) GUI/CLI		
KPI	-		
Components Involvement	AI/ML Framework		
Pre-test conditions	The AI/ML Framework has to be deployed An AI/ML pipeline for testing is required		
Test sequence	Step 1	Manually onboard a ML pipeline into the POP (Airflow)	The pipeline should be loaded successfully



		without raising exceptions
	Step 2	List models loaded in POP (Airflow) CLI or check the GUI
Test Verdict	The test must show that the AI/ML pipeline is onboarded successfully to the POP either through the GUI or the CLI	
Additional Resources	CLI command: airflow dags list	

The results are demonstrated in the following figures. In first place Figure 20 shows the Airflow GUI with the toy model load and then Figure 21 shows the airflow CLI output, listing all the AI/ML pipelines (DAGs) onboarded into the system.

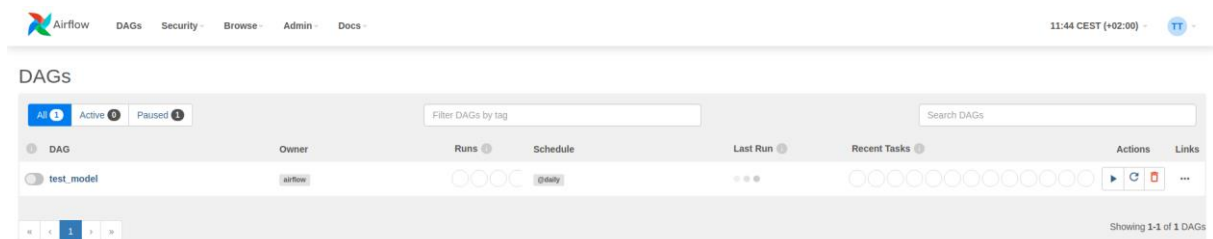


Figure 20 GUI showing the AI/ML pipeline successfully loaded

```
> airflow dags list
running bdist_wheel
running build
running build_py
creating build
creating build/lib
copying test.py -> build/lib
copying test_utils.py -> build/lib
installing to /tmp/tmpysxws5wo
running install
running install_lib
copying build/lib/test.py -> /tmp/tmpysxws5wo
copying build/lib/test_utils.py -> /tmp/tmpysxws5wo
running install_egg_info
running egg_info
creating tfx_user_code_Transform.egg-info
writing tfx_user_code_Transform.egg-info/PKG-INFO
writing dependency_links to tfx_user_code_Transform.egg-info/dependency_links.txt
writing top-level names to tfx_user_code_Transform.egg-info/top_level.txt
writing manifest file 'tfx_user_code_Transform.egg-info/SOURCES.txt'
reading manifest file 'tfx_user_code_Transform.egg-info/SOURCES.txt'
writing manifest file 'tfx_user_code_Transform.egg-info/SOURCES.txt'
Copying tfx_user_code_Transform.egg-info to /tmp/tmpysxws5wo/tfx_user_code_Transform-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9-py3.7.egg-info
running install_scripts
creating /tmp/tmpysxws5wo/tfx_user_code_Transform-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/WHEEL
creating '/tmp/tmp9ba0bk/tfx_user_code_Transform-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9-py3-none-any.whl' and adding '/tmp/tmpysxws5wo' to it
adding 'test.py'
adding 'test_utils.py'
adding 'tfx_user_code_Transform-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/METADATA'
adding 'tfx_user_code_Transform-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/WHEEL'
adding 'tfx_user_code_Transform-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/top_level.txt'
adding 'tfx_user_code_Transform-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/RECORD'
removing /tmp/tmpysxws5wo
running bdist_wheel
running build
running build_py
creating build
creating build/lib
copying test.py -> build/lib
copying test_utils.py -> build/lib
installing to /tmp/tmp3ym094m9
running install
running install_lib
copying build/lib/test.py -> /tmp/tmp3ym094m9
copying build/lib/test_utils.py -> /tmp/tmp3ym094m9
running install_egg_info
running egg_info
creating tfx_user_code_Trainer.egg-info
writing tfx_user_code_Trainer.egg-info/PKG-INFO
writing dependency_links to tfx_user_code_Trainer.egg-info/dependency_links.txt
writing top-level names to tfx_user_code_Trainer.egg-info/top_level.txt
writing manifest file 'tfx_user_code_Trainer.egg-info/SOURCES.txt'
reading manifest file 'tfx_user_code_Trainer.egg-info/SOURCES.txt'
writing manifest file 'tfx_user_code_Trainer.egg-info/SOURCES.txt'
Copying tfx_user_code_Trainer.egg-info to /tmp/tmp3ym094m9/tfx_user_code_Trainer-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9-py3.7.egg-info
running install_scripts
creating /tmp/tmp3ym094m9/tfx_user_code_Trainer-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/WHEEL
creating '/tmp/tmp6c615yan/tfx_user_code_Trainer-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9-py3-none-any.whl' and adding '/tmp/tmp3ym094m9' to it
adding 'test.py'
adding 'test_utils.py'
adding 'tfx_user_code_Trainer-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/METADATA'
adding 'tfx_user_code_Trainer-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/WHEEL'
adding 'tfx_user_code_Trainer-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/top_level.txt'
adding 'tfx_user_code_Trainer-0.0-bfb3a3ae36b5ed310f2bf59bfff08d5ff04b595d5ca48185c79630215faf854c9.dist-info/RECORD'
removing /tmp/tmp3ym094m9
dag_id | filepath | owner | paused
=====+=====
test_model | test_model/test.py | airflow | True |
```

Figure 21 CLI showing the AI/ML pipeline successfully loaded

#### 4.1.3.2 Execute a ML pipeline in the POP

The purpose of this individual test case is to demonstrate the proper operation of the POP showing an example of a toy pipeline (TFX) [21] properly executed by the Airflow instance of the AI/ML Framework.

Test case name	Execute a ML pipeline	Test Case id	Ind-test-03-02
Test purpose	Test the correct execution of a ML pipeline by the POP		
Configuration	POP installed and running ML pipeline uploaded to the POP		
Test tool	POP (Airflow) GUI/CLI		
KPI	-		
Components Involvement	AI/ML Framework		
Pre-test conditions	The AI/ML Framework has to be deployed Ind-test-03-01 successfully completed		
Test sequence	Step	Trigger the pipeline execution from the POP (Airflow) GUI/CLI	All the stages of the pipeline should run successfully
	Step	Check if the model is properly exported to the output directory	
Test Verdict	The test must show that the AI/ML pipeline is automatically deployed by the AMD in the MSP		
Additional Resources	None		

The results of this individual test case are presented in the following lines. Firstly, Figure 22 shows the Airflow GUI with the model enabled with different runs, demonstrating the proper operation of the POP. Then, Figure 23 shows all the different components that conforms the model pipeline (DAG) properly executed highlighted in green (success tag).

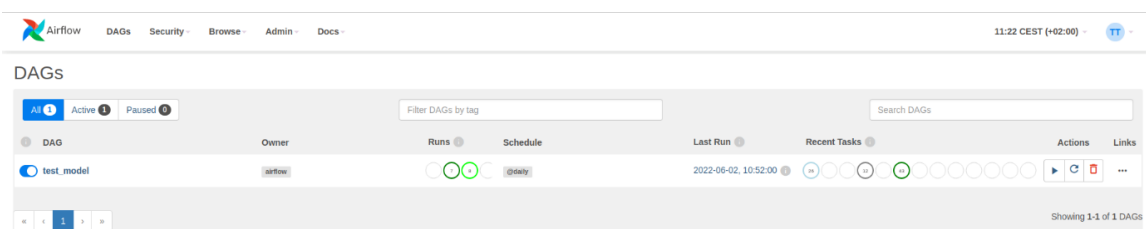


Figure 22 GUI showing the AI/ML pipeline enabled

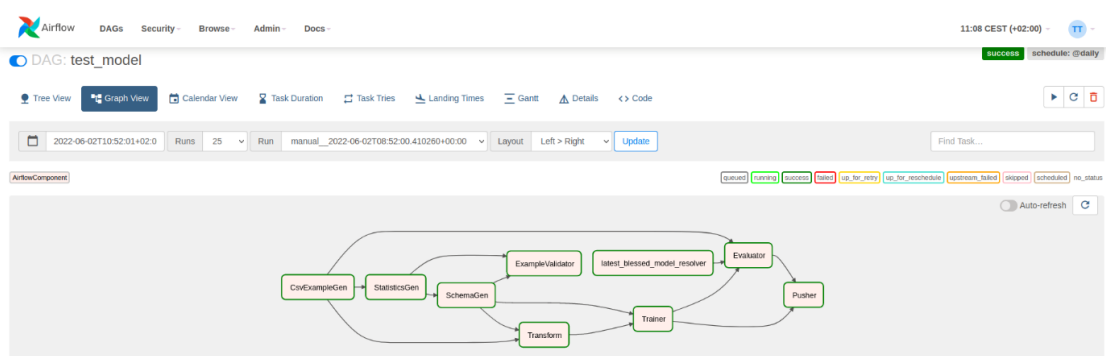


Figure 23 GUI showing the AI/ML pipeline graph view with all the pipeline components successfully executed



#### 4.1.3.3 Automatic deployment of models in the MSP by the AMD

The purpose of this individual test case is to demonstrate the proper operation of the AMD component of the AI/ML Framework, showing how the exported models are successfully detected by the component and the MSP configuration file is updated.

Test case name	Model deployment	Test Case id	Ind-test-03-03
Test purpose	Test the correct operation of the AMD and check if exported models are properly deployed in MSP		
Configuration	POP installed and running AMD installed and running ML pipeline uploaded to the POP ML model exported		
Test tool	CLI		
KPI	-		
Components Involvement	AI/ML Framework		
Pre-test conditions	The AI/ML Framework has to be deployed Ind-test-03-01 successfully completed Ind-test-03-02 successfully completed		
Test sequence	Step	Trigger the pipeline execution from the POP (Airflow) GUI/CLI	Model should be properly exported in the output folder
	Step	Check the logs of AMD to verify if the exported model has been detected	Logs should show the detection of the exported model
	Step	Check the MSP configuration file to check that the model has been deployed	MSP configuration file should be updated
Test Verdict	The ML pipeline is automatically deployed successfully by the AMD in the serving platform		
Additional Resources	Command: docker logs AMD		

The result of the aforementioned test case is presented in the following figures. Figure 24 shows the logs of the AMD instance demonstrating a successful automatic deployment of a toy model in the TFS instance of the AI/ML Model Serving Platform. In addition, Figure 25 shows the TFS configuration file that has been automatically updated by the AMD.

```
> docker logs AMD
Initializing /models/models.config ...
Looking for models in /models ...
saved_model.pb found in: /models/test_model/1653320215/saved_model.pb
Adding model ...
Model test_model succesfully added to: /models/models.config
```

Figure 24 AMD logs showing the successful deployment of the model in the MSP

```
> cat models/models.config
model_config_list{
  config{
    name: 'test_model'
    base_path: '/models/test_model'
    model_platform: 'tensorflow'
  }
}
```

Figure 25 MSP configuration file automatically updated by the AMD

#### 4.1.3.4 Serve a model in the MSP

The purpose of this individual test case is to demonstrate the proper operation of the TFS instance of the AI/ML Framework MSP component. This test aims to show that the model is correctly served through the REST interface of the TFS instance, which is the interface used in the context of the project. For this test we have used a toy model (half\_plus\_two) obtained from the official TensorFlow documentation TFSD [22].

Test case name	Model serving	Test Case id	Ind-test-03-04
Test purpose	Test the correct operation of the model serving platform		
Configuration	POP installed and running AMD installed and running ML pipeline uploaded to the POP ML model exported ML model deployed in the MSP		
Test tool	CURL		
KPI	-		
Components Involvement	AI/ML Framework		
Pre-test conditions	The AI/ML Framework has to be deployed Ind-test-03-01 successfully completed Ind-test-03-02 successfully completed Ind-test-03-03 successfully completed		
Test sequence	Step	Send POST with the model input to the MSP REST API	
	Step	Check response	
Test Verdict	Response status code is 200 OK, the model is being correctly served		
Additional Resources	CURL request example: curl -d '{"instances": [<model_input>']}' -X POST http://<ip_address>:8501/v1/models/<model_name>:predict		

The result of this test case is presented in the following lines. Firstly, Figure 26 shows that the model is available through the REST interface and secondly, Figure 27 shows an response to an exemplary prediction using the CURL command.

```
> curl http://localhost:8501/v1/models/test_model
{
  "model_version_status": [
    {
      "version": "1653320215",
      "state": "AVAILABLE",
      "status": {
        "error_code": "OK",
        "error_message": ""
      }
    }
  ]
}
```

Figure 26 MSP REST response showing the availability of a test model that is being served correctly

```
> curl -d '{"instances": [10]}' -X POST http://localhost:8501/v1/models/test_model:predict
{
  "predictions": [7.0
]
```

Figure 27 MSP REST response to a model prediction (half\_plus\_two)

#### 4.1.3.5 Measure the model inference latency of the MSP

The purpose of this individual test case is to validate that the MSP is able to meet the requirements of the different O-RAN control loops (Real-Time [ $<10\text{ms}$ ], Near-Real-Time [ $>10\text{ms}$  y  $<1\text{s}$ ], Non-Real-Time [ $>1\text{s}$ ]). Hence, this test case aims to evaluate the inference latency of two ML models with different complexity. First a baseline toy model (half\_plus\_two) provided by TensorFlow in the official documentation TFSD [22] and secondly, the CPU Prediction model developed by I2CAT in the context of Affordable5G Demo 3.

Test case name	Model inference latency	Test Case id	Ind-test-03-05
Test purpose	Measure the inference latency of the ML model serving platform		
Configuration	POP installed and running AMD installed and running ML model uploaded to the POP ML model exported ML model deployed in the MSP		
Test tool	Custom script		
KPI	Inference latency of the ML model serving platform (seconds)		
Components Involvement	AI/ML Framework		
Pre-test conditions	The AI/ML Framework has to be deployed Ind-test-03-01 successfully completed Ind-test-03-02 successfully completed Ind-test-03-03 successfully completed Ind-test-03-04 successfully completed		
Test sequence	Step	Send POST with the model input to the MSP REST API	
	Step	Check response	

	Step	Measure the time difference between the request and the response	
Test Verdict	Response status code is 200 OK, the model is being correctly served with a reasonable latency for each of the ORAN Control loops requirements (Real-Time [ $<10\text{ms}$ ], Near-Real-Time [ $>10\text{ms}$ y $<1\text{s}$ ], Non-Real-Time [ $>1\text{s}$ ])		
Additional Resources	Two models with different complexity will be tested in order to validate if it affects to the prediction latency		

The result of this test case is presented in Figure 28. The figure shows that the MSP of the AI/ML Framework is suitable for Real Time control loops obtaining an inference latency lower than 10ms.

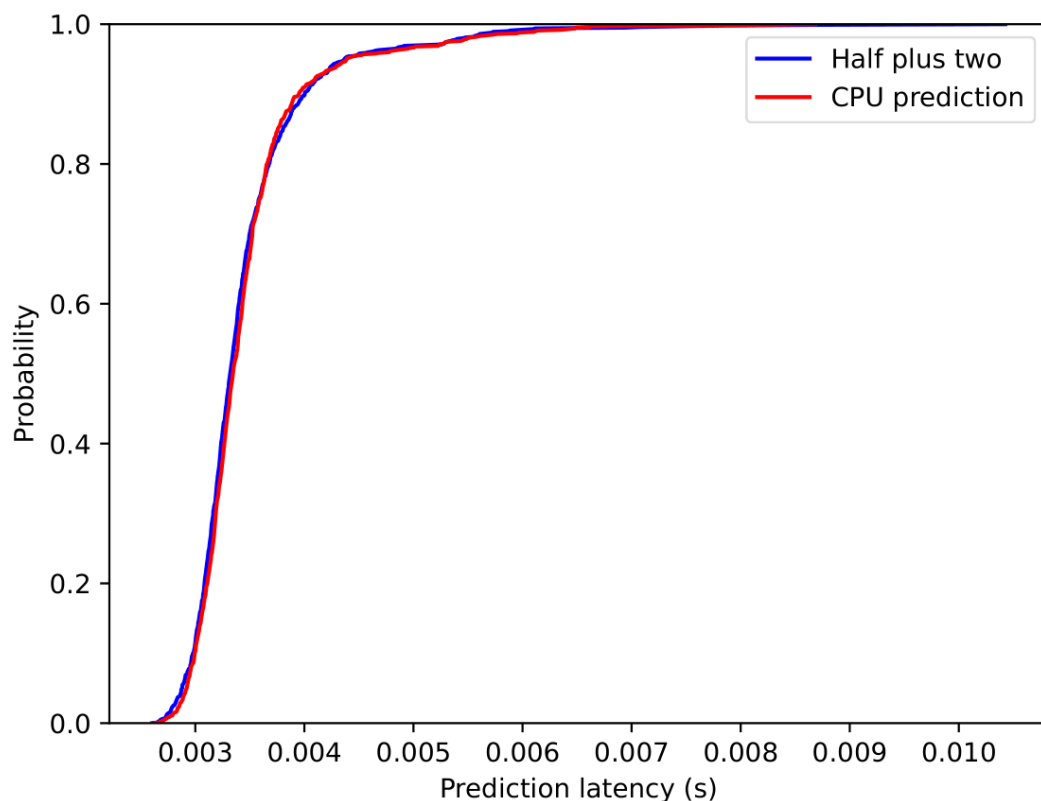
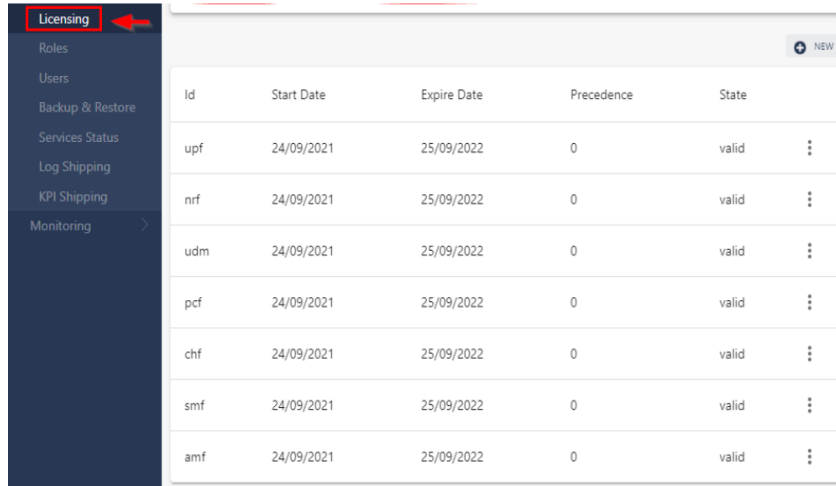


Figure 28 CDF of the prediction latency of a toy model (*half\_plus\_two*) and the CPU prediction model provided by I2CAT

## 4.1.4 5G Core

### 4.1.4.1 5GC License Validation test

This individual test case is needed to check the validation of all the 5GC's NFs, in order to make all the 5GC services running and available. Without a valid license, the NFs are not operative and configurable by the user.

<b>Test case name</b>	5GC License Validation	<b>Test Case id</b>	Ind-test-05-01
Test purpose	Validate the correct installation of all the 5GC Network Functions' license		
Configuration	5G core VM installation and network default configuration		
Test tool	5GC web GUI or REST API		
KPI	Functional test		
Components Involvement	5GC, license manager		
Pre-test conditions	The 5G core VM is installed and running. The network configuration is aligned with the network plan.		
Test sequence	Step 1	From the GUI or REST APIs, get the current Site Id of the 5G core instance	
	Step 2	Generate the license file for every Network Function using the Site Id in the license generator	
	Step 3	Install, one by one, all the obtained licenses	
Test Verdict	The licenses appear on the list of licenses in the Licensing page (GUI) or by retrieving the license list (REST APIs).		
Additional Resources	License page screenshot showing all the installed and active licenses: 		

The result of this test shows that all the licenses have been successfully registered, making the NFs fully operational.

#### 4.1.4.2 5GC Network configuration

The purpose of this test case is to check the network configuration of the 5GC in terms of available interfaces, IP pool settings, and modules reachability. Every NF that exposes its interface externally must be reachable from the other network elements. This test is extremely important as the connectivity with other network elements (gNB, dRAX, Orchestrators, etc.) depends on these configurations.

Test case name	5GC Network Configuration	Test Case id	Ind-test-05-02
Test purpose	Validate the correct network configuration and interfaces availability, according with the provided testbed network plan		
Configuration	5G core VM default installation		
Test tool	5GC web GUI, monitoring tool		
KPI	Functional test		
Components Involvement	5GC		
Pre-test conditions	The 5G core VM is installed and running on the Hypervisor		
Test sequence	Step 1	Proceed with the 5GC networking configuration based on the subnet availability.	
	Step 2	Setup the interfaces N1, N2, N3, N6	
	Step 3	Go to the monitoring tool to check that everything is up	
Test Verdict	The networking logs do not result any error, and interfaces IPs are reachable.		
Additional Resources			

Results of this test case is the confirmation that the interfaces are up, no errors are obtained after the network configuration and the core is reachable.

#### 4.1.4.3 UPF internal delay estimation

The purpose of this test is to evaluate the average packet latency through the user plane part (i.e., UPF). In this case, the session traffic packets are analyzed via some software tools (open-source and commercial ones) to estimate the average values on a total of processed traffic packets. The measured time is the time that elapses from the instant in which a packet enters the UPF to the instant it leaves it.

This test is relevant when it is required to analyze the total delay in an end-to-end 5G system and to confirm negligible delay introduced by the 5G core.

Test case name	UPF internal delay	Test Case id	Int-test-05-03
Test purpose	Estimation test of data packet delay introduced by the 5GC UP (UPF)		
Configuration	5GC installed on a Dell640 server with default installation and configuration. Network configured according to the network plan. gNB and UE simulators attached to the core. The server hardware in Malaga testbed has been taken as reference for the test-case.		
Test tool	Tcpdump, Proprietary test tool		
KPI	order of tens of $\mu$ s		
Components Involvement	Athonet 5GC		
Pre-test conditions	The 5G core VM is installed and running on the Hypervisor, network is configured and reachable. gNB and UE simulators are configured and reachable from the 5GC. A PDU session is established.		

Test sequence	Step 1	UE set up a connection to a PDN, in order to produce traffic.	
	Step 2	The testing tool starts to collect information about UP packets on the opposite UPF connection points, to measure the internal packet delay.	
	Step 2	After a certain amount of time, the testing tool stops to analyze the packets.	
Test Verdict	The average time is shown on the tool statistic		
Additional Resources			

The obtained result reflects what was expected from the various estimates, with an average delay of **tens of  $\mu$ s**, a negligible value compared to the total delay due to the whole end-to-end chain.

#### 4.1.4.4 Number of simultaneous UPFs supported

This test aims to evaluate the number of UPFs that the 5GC is able to support at the same time given the HW in use. Due to the importance of slicing in the context of the Affordable5G project, analyzing the ability of a 5GC to support concurrent UPFs in different edge nodes acquires some relevance.

The test case takes into consideration the hardware configuration installed at Malaga testbed (Dell R640). From the information extracted from specific metrics available from the 5GC user interface (see table below), it is possible to only estimate the number of UPFs concurrently connected.

Test case name	Simultaneous UPFs	Test Case id	Int-test-03-01
Test purpose	Estimation of the max number of UPFs a 5GC can support simultaneously in the current HW		
Configuration	5GC installed in a Dell R640 server default installation and configuration. Network configured according to the network plan. Several UPFs (Athonet UPFs or UPF simulators) are registered to the core. The server hardware in Malaga testbed has been taken as reference for the test-case.		
Test tool	GUI, REST API		
KPI	> 5 UPFs		
Components Involvement	Athonet 5GC		
Pre-test conditions	5GC installed on a Dell R.640 server is installed and running on the Hypervisor, network is configured and reachable.		
Test sequence	Step 1	A significant number of UPFs are registered and attached to the SMF.	
	Step 2	The stability and performance of the 5GC system is checked while the UPFs registration process is going	



	Step 3	The process is stopped once the system is not stable anymore or when the reference requirements are satisfied.	
Test Verdict	Number of attached UPF is visible in the GUI or can be retrieved via API		
Additional Resources			

With a typical network configuration and considering the performance of the hosting machine (HW+hypervisor), experiments indicate that the number of UPFs simultaneously installed can vary and go above 5, where this number depends on the required throughput and available computing-networking resources per UPF instance (constrained by the total HW and NIC card resources available).

#### 4.1.4.5 UPF Traffic throughput

This test evaluates the maximum throughput reachable from the user traffic through a single UPF. This specific metric is highly dependent on the adopted hardware and, in particular, on the network card, hence it is not a metric directly associated to core performance.

The test case takes into consideration the hardware configuration installed at Malaga testbed (Dell R640). The test has been conducted applying a mix of UL and DL traffic.

Test case name	UP throughput	Test Case id	Int-test-03-01
Test purpose	Evaluation of average maximum UL/DL traffic throughput supported by UP		
Configuration	5GC's VM default installation and configuration. Network configured according to the network plan. gNB and UE simulators attached to the core.		
Test tool	Tcpdump, Iperf, Proprietary tool		
KPI	Max 10 Gbps in UL/DL constrained by the HW in use and small deviations in the throughput observed due to the Hypervisor		
Components Involvement	Athonet 5GC		
Pre-test conditions	5GC installed on a Dell 640 server is installed and running, network is properly configured and reachable. A gNB and UE simulators are configured and reachable from the 5GC. A PDU session is established.		
Test sequence	Step 1	UE setup a connection to a PDN via Iperf, in order to produce traffic.	
	Step 2	After a certain amount of time, Iperf and/or other tool stops to execute	
Test Verdict	The average time is shown on the Grafana tool or a proprietary tool statistics		
Additional Resources			

#### 4.1.5 Power Measurement Testbed for NEOX Accelerators

NEOX [11] is a parallel multicore and multithreaded GPU-like Deep Neural Network (DNN) architecture based on the RISC-V RV64C ISA instruction set with adaptive Network-on-Chip (NoC) offered by THI. NEOX is fully customizable in terms of number of threads per core, width of the vector processing lanes and memory resources (private and shared caches as well as the cache prefetching options). The NEOX multithreading capabilities hides long latency delays from external memory controller maintaining high computation throughput for the entire array sources (both in terms of memory and computational capabilities) and also devices operating under tight power constraints).

The target of this testbed is to calculate the power consumption of NEOX accelerator at the ASIC level.

Test case name	Power consumption estimation of NEOX accelerator	Test Case id	Ind-test-05-03
Test purpose	Highly accurate estimation of power consumption of NEOX accelerators		
Configuration	The estimation of the power consumption will be performed at the netlist level (lowest level of the design, thus the one with the highest accuracy) using for 4 different configurations of NEOX accelerators (varying the number of cores and blender configurations).		
Test tool	Design compiler and power compiler of Synopsys. A single convolution layer of size equal to 24Kbytes.		
KPI	Average power consumption, peak power consumption, and maximum frequency		
Components Involvement	NEOX accelerator with four different configurations		
Pre-test conditions	HDL (verilog) code of NEOX can be synthesized into ASIC. Design compiler and power compiler of Synopsys are running. Two different process libraries are available.		
Test sequence	Step 1:	Synthesized code is executed at gate level to measure execution time	
	Step 2:	Synthesized code is executed at gate level to extract (bit) switching activities (SAIF) files	
	Step 3:	Power compiler is configured to take as input the SAIF files	
	Step 4:	Power compiler is configured to take as input two process libraries for two different process technologies	
Test Verdict	For the single convolution layer, the measured execution time and power consumption was from 2.58 to 4.11 mWatts (depending on the number of cores) for performing 30 inferences per second.		

The figure below shows the testbed for measuring the ASIC-level power consumption for NEOX accelerator. It is important to mention that the power consumption is measured at the netlist level, thus highly accurate measurements are performed. As we can see in the figure

below, the power estimation frameworks consist of three distinct levels: synthesis level, simulation at gate level, and netlist level.

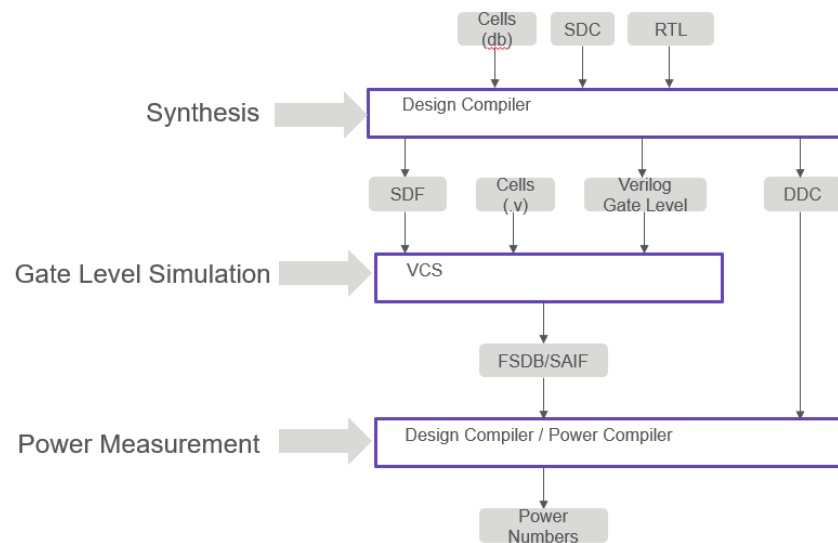


Figure 29 Simulation testbed for measuring power consumption of NEOX accelerator

The following picture shows the run-times in milliseconds as well as in frames or inferences per second when we vary the number of cores between 1 and 4.

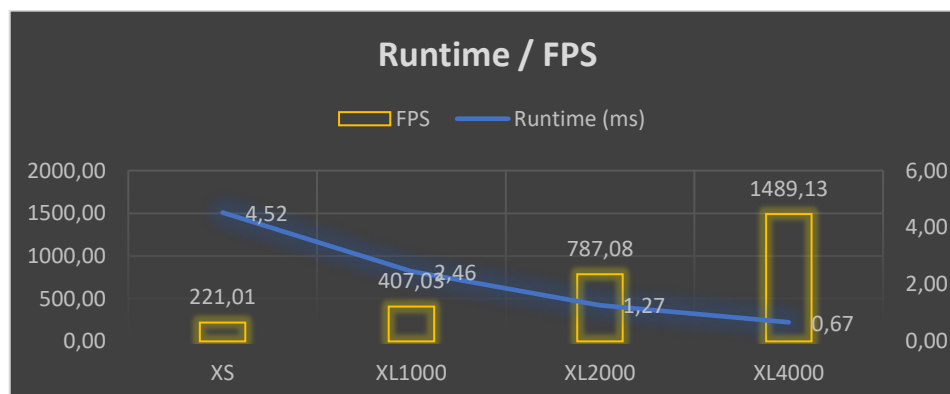


Figure 30 Latency and throughput for NEOX accelerator when the number of cores is modified

The next figure depicts the power (in mWatts) and energy consumption (in mJoules). As expected, the energy and power figures remain almost intact. Among others, this is evidence of the efficiency of the power and clock gating techniques that have been employed in NEOX hardware design.

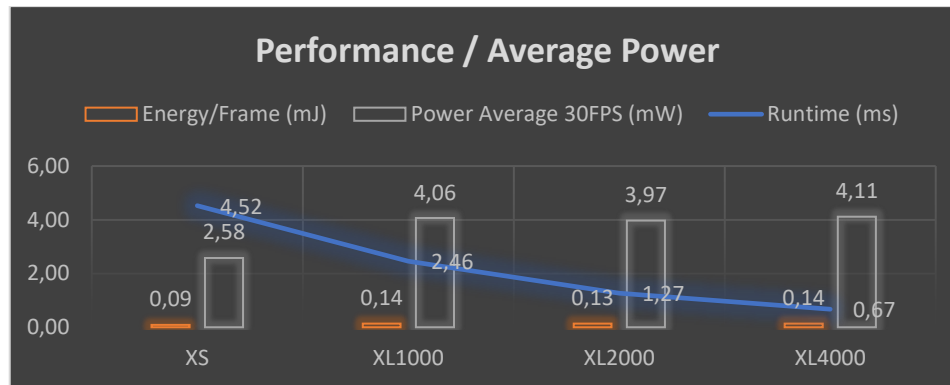


Figure 31 Power and energy of NEOX accelerator when the number of cores is varied

#### 4.1.6 TSN over 5G

This section includes individual test cases related to the TSN over 5G Proof of Concept developed by UMA, which is explained in detail in section 5. The test cases exposed below are related to e2e latency and jitter, which are the most representative KPIs in a time sensitive networking solution. As this solution is on an intermediate stage, the results obtained are expected to be improve with the final upgrades at the end of the project.

##### 4.1.6.1 TSN over 5G e2e latency

This first test case aims to validate one of the most important KPIs on a TSN over 5G solution, the e2e latency. Latency is calculated, using Network performance analyzer tool, as the difference between packets time stamp in sending and receiving, considering that both time clocks are synchronized. The test verdict has been defined according to service requirements and TSN features discussed on 5GACIA [5], using Cyclic-Asynchronous as traffic type.

Test case name	TSN e2e latency for critical traffic-		Test Case id	Ind-test-06-01
Test purpose	Test end-to-end latency between two synchronized TSN end points.			
Configuration	TSN end points configured to be able to be synchronized using ADVA equipment.			
Test tool	Network performance analyzer tool, Wireshark and packETH			
KPI	< 20 ms E2E latency.			
Components Involvement	All 5G components (Nokia RAN instead of O-RAN), NW-TT, DS-TT, ADVA FSP150 x2, TSN endpoints and Telit fn980m			
Pre-test conditions	TSN endpoints (PHC and system clocks) synchronized using ADVA equipment. Telit fn980m registered to 5G network.			
Test sequence	Step 1	Launch Stratum on both sides of the network (Network Side - TT and Device Side -TT)		The translators are operative, forwarding all packages, but not routing rules are configured yet.
	Step 2	P4Runtime is launched on both sides of the network (Network Side - TT and Device Side - TT)		Forwarding rules are included to prioritize critical traffic.

	Step 3	Configure packETH (packet generator) to send UDP packets, encapsulated on VLAN, from endpoint 1 (network side) to endpoint 2 (device side), through the 5G network.	The tool is ready to send traffic.
	Step 4	Open, configure and start Wireshark on sending and reception to capture traffic.	Wireshark is capturing traffic.
	Step 5	Send traffic with packETH with a period of 20 ms between packets.	Wireshark is now receiving critical traffic.
	Step 6	Save .pcap files with captured traffic and process them using Network performance analyzer tool to obtain latency.	Latency results are obtained.
Test Verdict	If peak latency is < 20 ms		
Additional Resources	None		

The results obtained after executing this test case can be observed in Figure 32. The figure shows that latency is below 20 ms during the whole test, so test verdict is PASS.

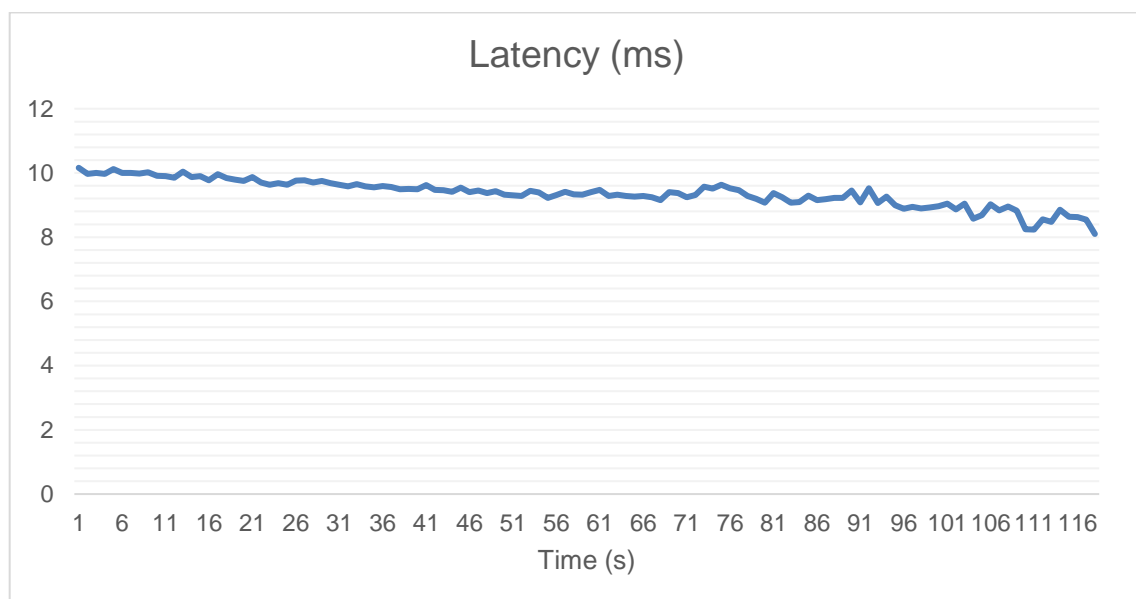


Figure 32 E2E latency results

#### 4.1.6.2 TSN over 5G e2e jitter

This test case is focused on e2e jitter measurement. As jitter measurement can be obtained while latency test case is performed, Ind-test-06-01 will be used as a pre-test condition. Again, test verdict is based on 5GACIA [5].

Test case name	TSN e2e jitter for critical traffic-	Test Case id	Ind-test-06-02
Test purpose	Test end-to-end jitter between two synchronized TSN end points.		

Configuration	TSN end points configured to be able to be synchronized using ADVA equipment.		
Test tool	Network performance analyzer tool, Wireshark and packETH		
KPI	5,12 ms E2E jitter.		
Components Involvement	All 5G components (Nokia RAN instead of O-RAN), NW-TT, DS-TT, ADVA FSP150 x2, TSN endpoints and Telit fn980m		
Pre-test conditions	Ind-test-06-01 executed		
Test sequence	Step 1	Use the same .pcap files obtained in Ind-test-06-01.	
	Step 2	Process the file with Network performance analyzer tool to obtain jitter.	Jitter results are obtained.
Test Verdict	If jitter is $< T$ , where $T$ is the time period.		
Additional Resources	None		

After processing the .pcap file, jitter is calculated. In Figure 33, we can see that the result is below time period.

```

11999 Packets send from udp.port==8888
11999 Packets received from udp.port==8888
Average delay(ns): 9360552
Average delay(ms): 9.360552
Average jitter(ns): 5129479
Average jitter(ms): 5.129479
Packet loss percentage: 0.0%

```

Figure 33 e2e jitter results

#### 4.1.6.3 TSN over 5G one-way latency and jitter test

In first stage of the project REL and ADVA have investigated 5G stripped-down system at REL lab. A TSN-optimized UPF-U prototype software (TSN-UPF) has been developed for this purpose. The goal was to assess the absolute minimal one-way latency and jitter values (Affordable5G, 2022).

The testbed for the stripped-down system with UP shortcut is illustrated in Figure 34.

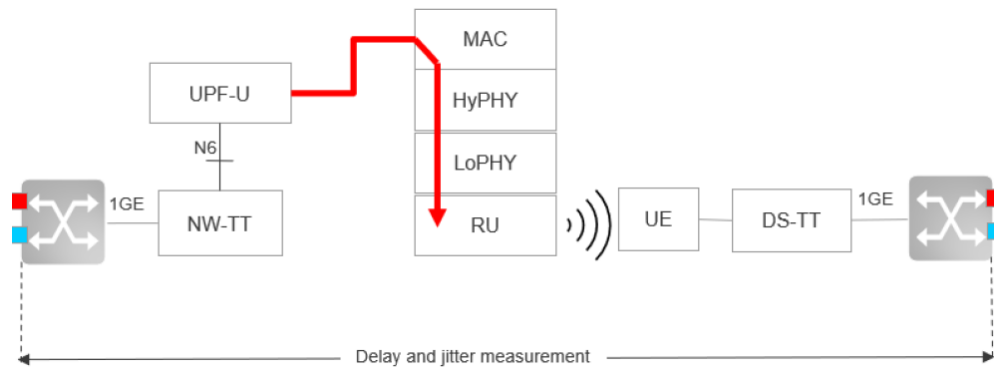


Figure 34 UP shortcut testbed – baseline

The latency and jitter results for stripped-down system are the baseline for the testing the whole Affordable5G system. To assess one-way latency and jitter for fully integrated 5G system at Malaga site we are developing UPF-U and UPF-C prototype software, which now are in the final integration phase. Like in the previous test setup, a synthetic UDP stream will be same parameters will be used.

Test case name	TSN one-way latency and jitter		Test Case id	Ind-test-06-03
Test purpose	Assess one-way latency and jitter for fully integrated Affordable5G system with TSN-optimized UPF			
Configuration	TSN-UPF, gNB,5GC, UE, PDU Session Type IPv4			
Test tool	Web browser to ADVA FSP-150 dashboard, ADVA embedded test generator and analyzer ECPA (EtherJack™ Connection Performance Analyzer), SSH client to TSN-UPF			
KPI	One-way latency and jitter			
Components Involvement	TSN-optimized UPF, a whole 5G system, UE			
Pre-test conditions	TSN-UPF is integrated with both 5GC and gNB. UPF and UE have a wired connection for the one-way latency measurement			
Test sequence	Step 1	Connect UE egress to UPF ingress		Ports status is Normal on both UPF and UE
	Step 2	Via web GUI configure ECPA towards UPF		Validate the UDP stream configuration
	Step 3	Start UDP test stream with N-frames and measure the latency and jitter for 3 minutes		All packets were received back by UPF
	Step 4	Start UDP test stream for 3 minutes		Monitor Min and Max delay
Test Verdict	Baseline for one-way latency and jitter - stripped-down system test (Affordable5G, 2022): Min delay: 1.192 ms, Max delay is 2.695 ms Jitter is 1.503 ms			
Additional Resources	None			



#### 4.1.7 Orchestrator

We define a series of tests to validate the use of Nearby One Orchestrator as discussed in section 3.2.2. These tests include:

- Near Zero Touch Provision of a COTS Server.
- Deployment of a sample xNF/App on a provisioned node.
- Undeployment of a xNF/App

For easy reproducibility of these tests, we use a combination of Concourse-CI (<https://concourse-ci.org/>) with Cypress (<https://www.cypress.io/>) end-to-end testing tool to run our whole set of individual tests.

These orchestrator individual tests verify that the orchestrator is able to provision the infrastructure edge nodes that will be used later to deploy all the apps, vnfs and slices configuration. It does so by running self-contained simple tests not requiring integrations with external components from other partners.

##### 4.1.7.1 Near Zero Touch Provision of a COTS Server

This test proves that the orchestrator can be used to provision COTS servers in Castellolí environment. For this test we use a server with no prior-configuration at all because the server is IPMI/redfish enabled, in other scenarios we also where this interface is not available, the only thing required si to configure the server to boot from a virtual-ISO or USB device that triggers the iPXE provision of the node:

Test case name	Nztp-provision Node		Test Case id	Ind-test-07-01
Test purpose	Nztp-Provision a COTS server.			
Configuration	NearbyOne e2e-orchestrator is up and running, NearbyOne e2e-orchestrator L3-reachable from the servers to be provisioned. The COTS-Server is configured to boot from the provided ipxe.iso			
Test tool	Cypress (it automatically follows the Test sequence steps and checks the expected responses)			
KPI	Functional test, Concurrent provisions ( $\geq 3$ nodes, no more COTS nodes were available for higher number of concurrent provision tests)			
Components Involvement	Orchestrator			
Pre-test conditions	The orchestrator is up and running			
Test sequence	Step 1	Open the NearbyOne GUI and visit <a href="#">/app/infrastructure/add-device</a> to register the device providing: <ul style="list-style-type: none"> <li>- location,</li> <li>- HW_identifier/TPM,</li> <li>- workflow (defines the OS, drivers and HW/SW</li> </ul>	The response code is 200. A deviceId is generated for the device. The device is registered in NearbyOne. If the HW_identifier already exists or some data is missing in the request an error is	

		configuration to be installed). - Site: "Castellolí"  Optionally: - IPMI address - IPMI port  IPMI credentials	reported and the test is considered a failure.
	Step 2	The server is physically installed in Castellolí and switched ON, and/or is started remotely.	The provisioning process is started, involving different Nearbyone components. It will be completed in <30min after the HW, OS and any other SW defined in the workflow has been configured. The provisioning goes through different stages, that are reported in the dashboard: 1 / 4: Initial boot 2 / 4: Installing base OS, drivers and configuring HW 3 / 4: Installing NearbyOne Agents 4 / 4: Ready
	Step 3	If after 60min the provision has not reached the Ready status, it will be flagged as a failure	
	Step 4	Visit <a href="#">app/infrastructure/view/device/\${devId}/specs</a> and check it's reporting the device inventory with all the resources it has available	The inventory includes the number of CPUs, memory, interfaces, ...
Test Verdict	The device status is reported as Ready. VNF/CNF/Apps can now be deployed to the server.		
Additional Resources	Cypress generates logs of the whole process and a video recording.		

#### 4.1.7.1.1 Near Zero Touch Provision of a COTS Server test results

The following screenshots Figure 35, Figure 36 and Figure 37 show the results obtained from running this test in Concourse CI using Cypress. It also includes other tests not documented here as these aren't as relevant (e.g., these checking that users without permissions can't provision devices, or that we can't provision devices with duplicated ids).

```

02:16:55 infrastructure page as viewer user
02:16:57 ✓ cannot create a device
02:16:57
02:16:57 infrastructure page as admin user
02:17:04 ✓ fails to create a device with existing mac (6522ms)
02:32:15 ✓ provisions an nZTP device through UI (911272ms)
02:32:16
02:32:16
02:32:16 3 passing (15m)
02:32:16 [mochawesome] Report JSON saved to /tmp/build/f541ec31/repo/cypress/results/mochawesome_002.json
02:32:16
02:32:17 (Results)
02:32:17
02:32:17
02:32:17 Tests: 3
02:32:17 Passing: 3
02:32:17 Failing: 0
02:32:17 Pending: 0
02:32:17 Skipped: 0
02:32:17 Screenshots: 0
02:32:17 Video: true
02:32:17 Duration: 15 minutes, 20 seconds
02:32:17 Spec Ran: device/provision_device.js
02:32:17
02:32:17 (Video)
02:32:17
02:32:17 - Started processing: Compressing to 32 CRF
02:33:30 - Finished processing: /tmp/build/f541ec31/repo/cypress/videos/device/provision_de (1 minute, 13
02:33:30 vice.js.mp4
02:33:30
  
```

Figure 35 Concourse-ci screenshot showing the provisioning device tests.

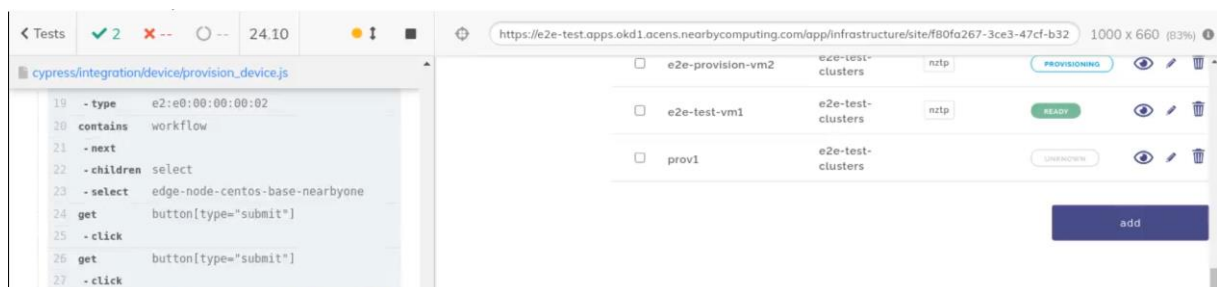


Figure 36 Cypress video's screenshot showing the e2e-provision-vm2 in « provisioning » status at time 24.20s of the test.

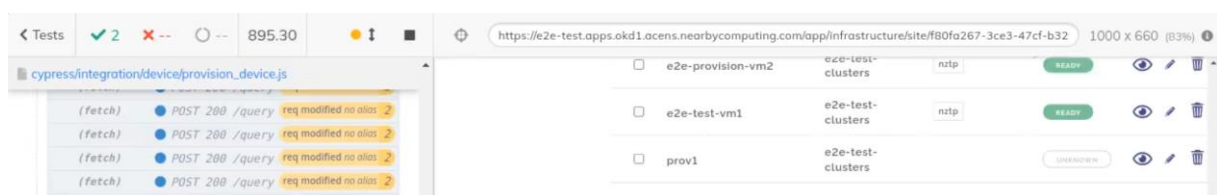


Figure 37 Cypress video's screenshot showing the e2e-provision-vm2 in « Ready » status at time 895.30s of the test.

#### 4.1.7.2 Deployment of a sample xNF/App on a provisioned node

This test shows how to deploy a sample Application using the same procedure that would be run in case of more complex slices involving other kind of resources:

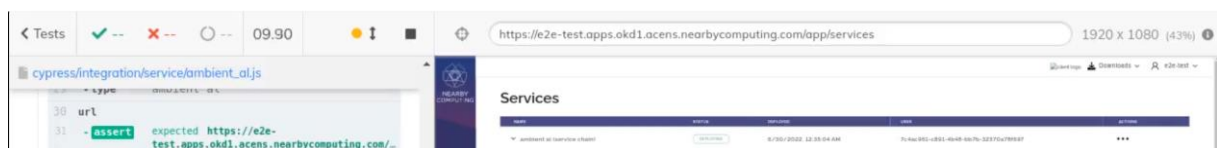
<b>Test case name</b>	Deploy sampleVNF/APP	<b>Test Case id</b>	<b>Ind-test-07-02</b>
Test purpose	Deploy a sampleVNF/APP to an edge Node. The sample app is a web server that echoes some strings provided as app configuration.		
Configuration	NearbyOne e2e-orchestrator is up and running, NearbyOne VNF/APP library includes the Ambient-AI sample app.		
Test tool	Cypress (it automatically follows the Test sequence steps and checks the expected responses)		
KPI	Functional test Time to trigger the deployment on the edge nodes: <5s (the completion time for a specific service to be ready depends on the size of the container/VM images and that can vary greatly for different Apps/VNFs).		
Components Involvement	Orchestrator		
Pre-test conditions	There's at least one nztp-provisioned edge server (as the one available after Ind-test-06-01) with resources matching what the APP requests. If no device is available all the steps are still valid but the final status of the sample app will remain as "pending" until such a valid device is available.		
Test sequence	Step 1	Open the NearbyOne GUI and visit <a href="/app/service-designer/marketplace">/app/service-designer/marketplace</a>	The browser shows a list of all the VNF/Apps available in Castelloli's NearbyOne env Marketplace.
	Step 2	Find the Ambient-AI sample app and click add to "Add to designer"	The view shows that 1 Instance of the app has been added to the designer.
	Step 3	Visit <a href="/app/service-designer/designer">/app/service-designer/designer</a> By default, the echoed "test" var in the http server is set to "foovar"	The view shows the different VNF/Apps added to the current service-chain. And its editable default configuration.
	Step 4	Edit the placement/site to "Castelloli" to have the sample app deployed to any node at Castelloli's site.	The inventory includes the number of CPUs, memory, interfaces, ...
	Step 5	Visit <a href="/app/service-designer/designer/deploy">/app/service-designer/designer/deploy</a> set a name for the deployment and click Deploy	
Test Verdict	The App is shown as Ready in <a href="/app/services">/app/services</a> with a link the sample-app service that can be reached and "foovar" is shown in its response.		
Additional Resources	Cypress generates logs of the whole process and a video recording.		

#### 4.1.7.2.1 Deployment of a sample xNF/App on a provisioned node test result

The following screenshots Figure 38, Figure 39 and Figure 40 show the results obtained from running this test in Concourse CI using Cypress. The same test includes in one go, both the deployment, update and deletion, that are documented as part of the next test in 4.1.7.3. It also includes other tests not documented here (e.g., updating the deployed version of an application/VNF).

```
02:34:56 Service designer page - Ambient al block
02:35:32 ✓ deploy, updates and removes Ambient-Al (35396ms)
02:35:41 ✓ updates a service chain to a newer version (9285ms)
02:35:41
02:35:41 2 passing (45s)
02:35:41
02:35:41 [mochawesome] Report JSON saved to /tmp/build/f541ec31/repo/cypress/results/mochawesome_009.json
02:35:41
02:35:42 (Results)
02:35:42
02:35:42 _Tests:      2
02:35:42 _Passing:    2
02:35:42 _Failing:    0
02:35:42 _Pending:    0
02:35:42 _Skipped:    0
02:35:42 _Screenshots: 0
02:35:42 _Video:      true
02:35:42 _Duration:   44 seconds
02:35:42 _Spec Ran:   service/ambient_al.js
02:35:42
02:35:42
02:35:42 (Video)
02:35:42
02:35:42 - Started processing: Compressing to 32 CRF
02:35:47 - Finished processing: /tmp/build/f541ec31/repo/cypress/videos/service/ambient_al_. (5 seconds)
02:35:47 js.mp4
```

Figure 38 Concourse-ci screenshot showing the provisioning device tests.



*Figure 39 Cypress video's screenshot showing the sample-deployment « ambient-al block » in « provisioning » status at time 9.90s of the test.*

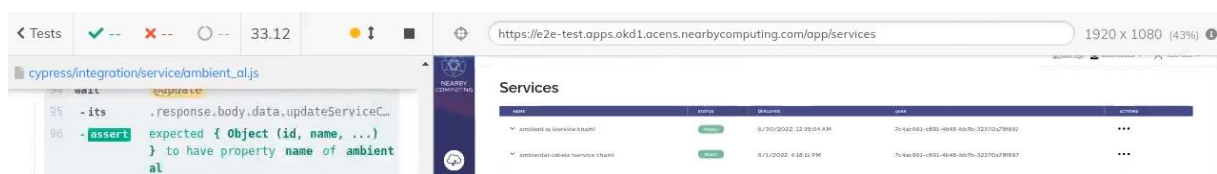


Figure 40 Cypress video's screenshot showing the sample-deployment « ambient-al block » in « Ready » status at time 33.12 s of the test.

#### 4.1.7.3 Undeployment of a xNF/App

This test shows how these resources previously allocated by deployments like the ones shown in the previous test can be removed from the orchestrator dashboard, triggering all the required actions to remove from the different platforms all the resources that were deployed in that slice/App/Xnf

Test case name	Delete sampleVNF/APP	Test Case id	Ind-test-07-03
Test purpose	Delete a sampleVNF/APP to an edge Node.		
Configuration	NearbyOne e2e-orchestrator is up and running.		

Test tool	Cypress (it automatically follows the Test sequence steps and checks the expected responses)		
KPI	Functional test Time to trigger the undeployment on the edge nodes: <5s		
Components Involvement	Orchestrator		
Pre-test conditions	There's a VNF/APP available to be deleted, e.g., Ind-test-06-01 has passed successfully: there's no need for the app to be Ready.		
Test sequence	Step 1	Open the NearbyOne GUI and visit <a href="/app/services">/app/services</a>	The browser shows a list of all the VNF/Apps deployed in Castelloli's NearbyOne env.
	Step 2	Find the Ambient-AI sample instance and click "delete" icon.	The view opens a confirmation text-box before the instance is deleted.
	Step 3	Write the name of the service and click "delete".	
Test Verdict	The sample app is undeployed and disappears from <a href="/app/services">/app/services</a> the http service is not available anymore.		
Additional Resources	Cypress generates logs of the whole process and a video recording.		

#### 4.1.7.3.1 Undeployment of a xNF/App test result

The previous screenshot from Figure 38, shown this test passing as part of the same one that was creating and updating the deployment. The Following screenshot Figure 41 shows its undeployment.



Figure 41 Cypress video's screenshot showing the sample-deployment « ambient-al block » in « Undeploying » status at time 34.90 s of the test.

### 4.1.8 Open 5G-RAN

The components of the Open 5G NG-RAN, consisting of the O-CU, O-DU, O-RU and Near-RT RIC fully interconnect and interact to create a running 5G gNB, so most of the test are documented in the integration test cases section 4.2. The individual test cases are related to the initial installation and configuration of the network functions, into a running state before attachment.

#### 4.1.8.1 dRAX Near-RT RIC Installation

This test installs a clean version of the cloud-native dRAX Near-RT RIC and the associated dRAX Dashboard.

Test case name	dRAX Near-RT RIC Installation	Test Case id	Ind-test-08-01
Test purpose	Validate the correct installation and configuration of the dRAX Near-RT RIC, following the dRAX installation guide <a href="https://accelleran.github.io/drax-docs/drax-install/">https://accelleran.github.io/drax-docs/drax-install/</a>		
Configuration	None specific pre-configuration.		
Test tool	Web browser		
KPI	Functional test		
Components Involvement	O-RAN Near-RT RIC		
Pre-test conditions	Software & hardware requirements fulfilled as per <a href="https://accelleran.github.io/drax-docs/drax-install/#software-and-hardware-prerequisites">https://accelleran.github.io/drax-docs/drax-install/#software-and-hardware-prerequisites</a> , correct Linux, Docker, K8S, Helm versions. dRAX license files installed <a href="https://accelleran.github.io/drax-docs/drax-install/#prepare-license-and-certificate">https://accelleran.github.io/drax-docs/drax-install/#prepare-license-and-certificate</a> DU license USB installed (not applicable when using OAI DU) K8S namespaces & advertise IP addresses <a href="https://accelleran.github.io/drax-docs/drax-install/#plan-parameters">https://accelleran.github.io/drax-docs/drax-install/#plan-parameters</a> and setup \$NODE_IP and \$NODE_INT according to the installation IP addressing plan.		
Test sequence	Step 1	Pre-requirements: adding Helm chart repo, namespaces & credentials, as per <a href="https://accelleran.github.io/drax-docs/drax-install/#pre-requirements">https://accelleran.github.io/drax-docs/drax-install/#pre-requirements</a>	
	Step 2	Install dRAX RIC and Dashboard as per <a href="https://accelleran.github.io/drax-docs/drax-install/#install-drax-ric-and-dashboard">https://accelleran.github.io/drax-docs/drax-install/#install-drax-ric-and-dashboard</a> . Ensure 5G components are enabled.	
	Step 3	Connect a web browser to the dRAX Dashboard: <a href="http://\$NODE_IP:31315">http://\$NODE_IP:31315</a> . Verify the installation as per <a href="https://accelleran.github.io/drax-docs/drax-install/#verifying-the-drax-installation">https://accelleran.github.io/drax-docs/drax-install/#verifying-the-drax-installation</a>	
Test Verdict	Via the dashboard, check that the dRAX RIC has been installed and is operational but at this point, there will be no 5G RAN components (CU, DU) or xApps installed, so a clean & empty RIC.		
Additional Resources	<a href="https://accelleran.github.io/drax-docs/drax-install/">https://accelleran.github.io/drax-docs/drax-install/</a>		

#### 4.1.8.2 dRAX CU Installation

This test installs the dRAX O-CU gNB network function. The CU-CP is installed.

For Malaga (where the O-RAN is configured manually), all the CU-UP instances will also be installed.

For Castellolí, only the CU-CP will be manually configured as the CU-UP instances will be created by the slice manager during addition of PLMNs & network slices.

Test case name	dRAX CU Installation	Test Case id	Ind-test-08-02
Test purpose	Validate the correct installation and configuration of the dRAX CU, following the dRAX CU installation guide		



	<a href="https://acceleran.github.io/drax-docs/drax-install/#install-drax-5g-components">https://acceleran.github.io/drax-docs/drax-install/#install-drax-5g-components</a>	
Configuration	<p>The configuration parameters entered depend on the network (both internal 5GC/RAN IP addressing and 5GS NG-RAN PLMN) configurations, so will differ for Malaga &amp; Castellolí.</p> <p>For Malaga (where the O-RAN is configured manually), all the steps will be followed.</p> <p>For Castellolí, only the CU-CP will be manually configured as the CU-UP instances will be created by the slice manager during addition of PLMNs &amp; network slices.</p>	
Test tool	Web browser to dRAX dashboard: <a href="http://\$NODE_IP:31315">http://\$NODE_IP:31315</a>	
KPI	Functional test	
Components Involvement	O-RAN O-CU	
Pre-test conditions	<p>dRAX RIC and Dashboard installed (previous test case executed).</p> <p>CU-CP configuration parameters defined and available: <a href="https://acceleran.github.io/drax-docs/drax-install/#required-parameters">https://acceleran.github.io/drax-docs/drax-install/#required-parameters</a></p> <p>If required: CU-UP configuration parameters defined and available: <a href="https://acceleran.github.io/drax-docs/drax-install/#required-parameters_1">https://acceleran.github.io/drax-docs/drax-install/#required-parameters_1</a></p>	
Test sequence	Step 1	From dRAX Dashboard, select New 5G CU deployment: <a href="https://acceleran.github.io/drax-docs/drax-install/#install-drax-5g-components">https://acceleran.github.io/drax-docs/drax-install/#install-drax-5g-components</a>
	Step 2	Select CU-CP and configure the parameters. Submit.
	Step 4	For each CU-UP instance to be manually configured, configure and submit as per <a href="https://acceleran.github.io/drax-docs/drax-install/#5g-cu-up-installation">https://acceleran.github.io/drax-docs/drax-install/#5g-cu-up-installation</a>
	Step 5	<p>Connect a web browser to the dRAX Dashboard: <a href="http://\$NODE_IP:31315">http://\$NODE_IP:31315</a>.</p> <p>Verify the installation as per <a href="https://acceleran.github.io/drax-docs/drax-install/#verifying-the-drax-installation">https://acceleran.github.io/drax-docs/drax-install/#verifying-the-drax-installation</a></p> <p>Also connect and check on the 5G system health dashboard on <a href="http://\$NODE_IP:30300">http://\$NODE_IP:30300</a>. pick the Accelleran dRAX 5G System Dashboard from the list of pre-built Grafana dashboards. (Note: not all services and interface may be running at this point)</p>
Test Verdict	Via the dashboards, check that the dRAX CU has been installed and is operational but at this point, there will be no connected DU	
Additional Resources	<a href="https://acceleran.github.io/drax-docs/drax-install/">https://acceleran.github.io/drax-docs/drax-install/</a>	

#### 4.1.8.3 gNB Transport Network Cross-haul and Time/Frequency Synchronization

This test case aims to validate the Open Fronthaul S-plane, implemented as PTP distribution between TNE (XG118) and O-DU (SE350). The O-RU receives the synchronization from GNSS independently, so it is not part of this test case. The configuration of the test is shown in Figure 17.

Test case name	Open Fronthaul S-plane	Test Case id	Ind-test-08-03
Test purpose	Validate the correct installation, configuration and operation of the Open Fronthaul S-plane between TNE and O-DU		
Configuration	The configuration parameters of PTP domain shall match the configuration of BC at TNE		
Test tool	Web browser to SoftSync dashboard: <a href="http://\$NODE_IP:8080">http://\$NODE_IP:8080</a> Web browser to TNE(XG118) dashboard: <a href="https://\$TNE_IP">https://\$TNE_IP</a>		
KPI	Functional test		
Components Involvement	O-RAN TNE, O-RAN O-DU		
Pre-test conditions	TNE is installed with GPS antenna connected. Fiber is connected between TNE Port#2 and O-DU port #N		
Test sequence	Step 1	Connect to TNE web GUI and validate PTP Boundary Clock Operational status is Normal	
	Step 2	Validate PTP stream is arrived at COTS (tcpdump -nne -i \$IF_NAME)	
	Step 3	Install SoftSync Software on O-DU COTS	
	Step 4	Configure IP address on COTS interface \$IF_NAME as 50.0.0.2/24	
	Step 5	Configure PTP clock and PTP port via web GUI	
	Step 6	Via web GUI check there is no alarms and the PTP Master IP is the correct one (50.0.0.1)	
	Step 7	Via web GUI monitor the state of PTP clock is Normal and no Alarms during 15 min	
Test Verdict	Via SoftSync dashboards, check that the SoftSync has been installed and is operational. The PTP clock should report proper values for {Clock Recovery State=Locked, Phase Recovery State=Locked, Time Traceability Status=True, Current Time Of Day=\$date}		
Additional Resources	<a href="https://www.oscilloquartz.com/en/products-and-services/embedded-timing-solutions/osa-softsync">https://www.oscilloquartz.com/en/products-and-services/embedded-timing-solutions/osa-softsync</a> <a href="https://www.adva.com/en/products/packet-edge-and-aggregation/edge-computing/fsp-150-xg-118pro">https://www.adva.com/en/products/packet-edge-and-aggregation/edge-computing/fsp-150-xg-118pro</a>		

The below test case is validating Open Fronthaul S-plane client telemetry.

<b>Test case name</b>	Open Fronthaul S-plane telemetry	<b>Test Case id</b>	<b>Ind-test-08-04</b>
Test purpose	Validate the S-plane telemetry streaming towards gNMI Collector platform		
Configuration	The configuration of SoftSync as in previous test		
Test tool	Web browser to SoftSync dashboard: <a href="http://\$NODE_IP:8080">http://\$NODE_IP:8080</a> , gNMIc client or Python3.7		
KPI	Functional test		
Components Involvement	O-RAN O-DU with SoftSync, gNMI collector application		
Pre-test conditions	SoftSync is up and running on O-DU COTS. gNMI collector software is installed (either gNMIc or Python3.7 pyGNMI)		
Test sequence	Step 1	Using gNMIc application, connect to SoftSync running on O-DU /\$NODE_IP:20830 and request capabilities	Got response : ModelData: name: ietf-ntp organization:

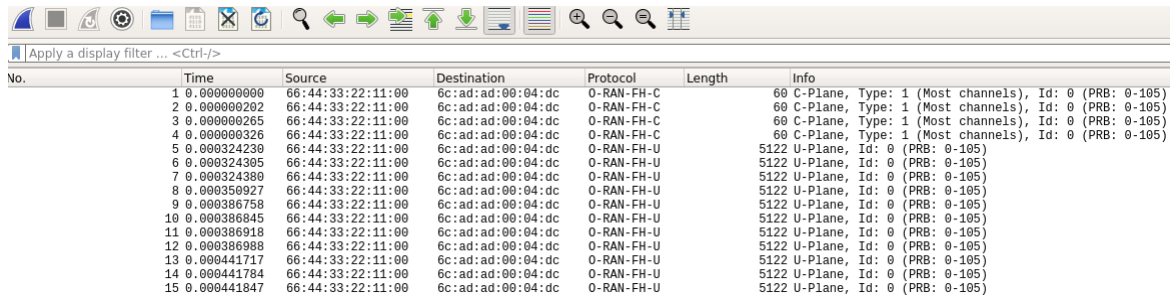
			IETF TICTOC Working Group
	Step 2	Subscribe to the telemetry stream gnmic -a \$NODE_IP:20830 --insecure -u admin -p admin sub --path "/ptp/instance-list/1/time-properties-ds/time-traceable" --sample-interval 2s	Successful completion
	Step 3	Validate the telemetry is updated streamed every 2 seconds	
Test Verdict	Via gNMIc application receive S-plane telemetry stream with the following paths [ /ptp/instance-list/1/current-ds/offset-from-master /ptp/instance-list/1/current-ds/mean-path-delay /ptp/instance-list/1/time-properties-ds/time-traceable /ptp/instance-list/1/default-ds/clock-quality/clock-class ]		
Additional Resources	<a href="https://www.oscilloquartz.com/en/products-and-services/embedded-timing-solutions/osa-softsync">https://www.oscilloquartz.com/en/products-and-services/embedded-timing-solutions/osa-softsync</a> <a href="https://netdevops.me/2020/gnmic-gnmi-cli-client-and-collector/">https://netdevops.me/2020/gnmic-gnmi-cli-client-and-collector/</a> <a href="https://gnmic.kmr.ddev/">https://gnmic.kmr.ddev/</a> <a href="https://datatracker.ietf.org/doc/html/rfc8575">https://datatracker.ietf.org/doc/html/rfc8575</a> <a href="https://github.com/Affordable5G/tsn-latency/tree/main/telemetry">https://github.com/Affordable5G/tsn-latency/tree/main/telemetry</a>		

#### 4.1.8.4 O-DU packet validation

The O-RAN compliant OAI O-DU will support the connection to 1 commercial O-RU at the time, a simultaneous multi-RU connection is forecasted after the validation of the single connection. The OAI O-DU will support a 2x2 MIMO, higher MIMO capabilities are currently under development. The OAI O-DU supports the PTP synchronization to the network grand master using linuxptp. The management plane can directly be accessed in the O-DU machine and a Netconf client-server configuration will be developed as a further step. The OAI O-DU supports both U-Plane and C-Plane but for now will not support beamforming.

Test case name	O-DU network and O-RAN packets validation	Test Case id	Int-test-02-03
Test purpose	Network setup and O-DU O-RAN packet transmission		
Configuration	OAI O-DU connected to a Eth port not binded with DPDK		
Test tool	Wireshark		
KPI	Filtered packets PTPV2 and O-RAN		
Components Involvement	Standalone OAI O-DU		
Pre-test conditions	O-RAN FHI library compiled OAI DU project built Network setup connection to the switch and Grand Master clock OAI-DU machine optimized for the O-RAN Front Haul		
Test sequence	Step 1	Run linux ptp in hardware mode and consequently phc2sys	

	Step 2	Run the OAI O-DU, and check the O-RAN library synchronization	
	Step 3	Run Wireshark capturing packets on the network interface not binded with DPDK	
Test Verdict	The Wireshark Capture shows the PTP messages and the O-RAN C-Plane and U-Plane packets		
Additional Resources			



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-C	60	C-Plane, Type: 1 (Most channels), Id: 0 (PRB: 0-105)
2	0.000000202	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-C	60	C-Plane, Type: 1 (Most channels), Id: 0 (PRB: 0-105)
3	0.000000265	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-C	60	C-Plane, Type: 1 (Most channels), Id: 0 (PRB: 0-105)
4	0.000000326	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-C	60	C-Plane, Type: 1 (Most channels), Id: 0 (PRB: 0-105)
5	0.000324230	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
6	0.000324305	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
7	0.000324380	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
8	0.000350927	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
9	0.000386758	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
10	0.000386845	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
11	0.000386918	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
12	0.000386988	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
13	0.000441717	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
14	0.000441784	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)
15	0.000441847	66:44:33:22:11:00	6c:ad:ad:00:04:dc	O-RAN-FH-U	5122	U-Plane, Id: 0 (PRB: 0-105)

Figure 42: PCAP capture of PTP, C-Plane and U-Plane traffic

#### 4.1.9 Individual Test Cases KPIs Summary table

The following table summarize the different KPIs defined per relevant individual test cases. In this table there are not detailed the functional test cases.

Table 1 Individual Test Cases KPIs Summary Table

Component	KPI	Test Case ID	Expected Value
OSM	Time for service to be deployed	Ind-test-01-01	3 s
Slice Manager	HTTP response codes Average initialization time	Ind-test-02-01	3.18 s
AI/ML Framework	Inference latency of the ML model serving platform	Ind-test-03-05	10 ms
5G Core	Estimation test of data packet delay introduced by the 5GC UP (UPF)	Ind-test-04-03	70 $\mu$ s
5G Core	Estimation of the max number of UPFs a 5GC can support simultaneously	Ind-test-04-04	20 UPFs
5G Core	Evaluation of average maximum UL/DL traffic throughput supported by UP	Ind-test-04-05	Max 8 Gbps in UL/DL
NEOX Accelerator	Average power consumption, peak power consumption, and maximum frequency	Ind-test-05-03	Between 2.58 and 4.11 mWatts
TSN over 5G	End-to-end latency between two synchronized TSN end points.	Ind-test-06-01	< 20 ms E2E latency
TSN over 5G	Test end-to-end jitter between two synchronized TSN end points	Ind-test-06-02	5,12 ms E2E jitter
TSN over 5G	One-way latency and jitter	Ind-test-06-03	Min delay: 1.192 ms,

			Max delay is 2.695 ms Jitter is 1.503 ms
NBYONE Orchestrator	Time to trigger the deployment on the edge nodes	Ind-test-07-02	<5s

## 4.2 Integration Test Cases

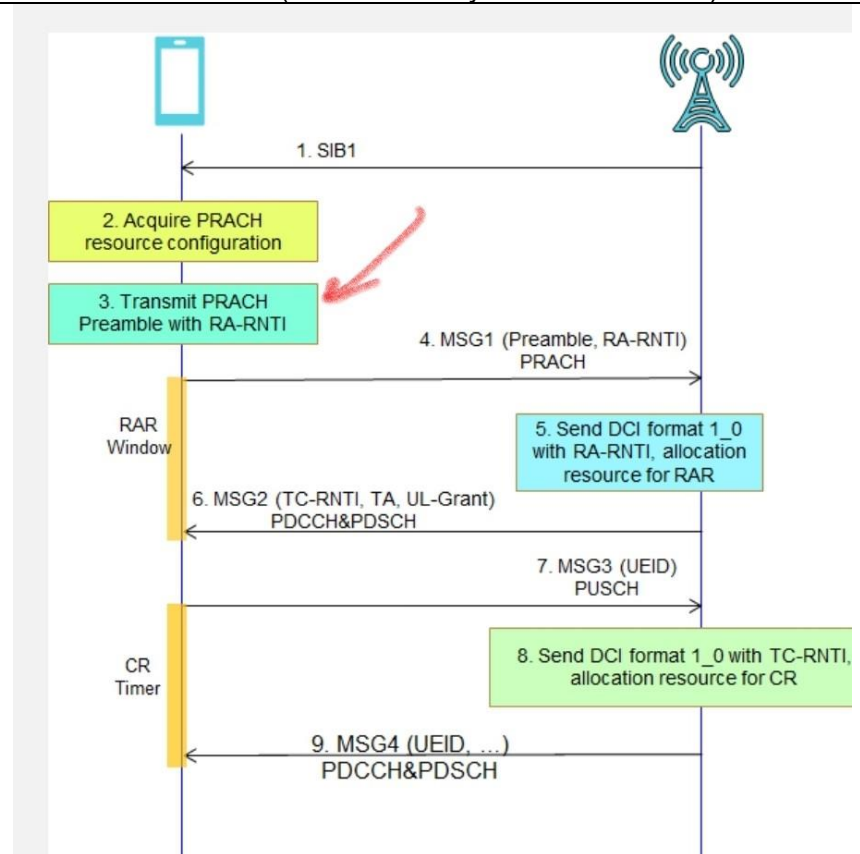
This subsection contains the Integration Test Cases, defined to test and monitor the correct performance of the elements and building blocks while interacting among each other conforming the architecture of the platform.

### 4.2.1 RU-DU

The RU and DU integration have encountered different problems, in terms of interfaces and interaction between components, but finally the following different Test Cases show the evolution of the integration status. The results of this integration test cases will be specified in the following deliverable D4.3.

Test case name	Validation of the 5G SA emitted spectrum		Test Case id	Int-test-01-01
Test purpose	Validation of the O-DU / O-RU transmitted signal			
Configuration	Indoor O-DU - O-RU RF			
Test tool	5G SA spectrum vector analyser			
KPI	Vector analyser can decode the principal radio channels			
Components Involvement	Integrated O-DU and O-RU			
Pre-test conditions	O-RAN FHI CP and UP between O-DU and O-RU is validated O-DU and O-RU synchronized O-DU, O-RU and Vector analyser properly configured			
Test sequence	Step 1	O-RU and Vector spectrum analyser located in the same room and RF I/O connected with RF cable		If it is not possible to connect them using proper antenna and LoS
	Step 2	Waveform through analyser and BCCH (DL): MIB, and SIBS decoding		
Test Verdict	The spectrum vector analyser decoded the radio channels			
Additional Resources	N/A			

Test case name	UE connection	Test Case id	Int-test-01-02
Test purpose	COTS UE properly connected		
Configuration	Indoor E2E testing O-DU - O-RU - UE		
Test tool	OAI UE or COTS UE + OAI logs + Wireshark + UE logs		
KPI	3GPP 5G SA message exchange for the UE connection		

Components Involvement	Integrated O-DU and O-RU + UE provisioned in the CN database		
Pre-test conditions	O-RAN FHI protocol between O-DU and O-RU is validated O-DU and O-RU synchronized O-DU, O-RU 5G SA spectrum validated UE provisioned with a test SIM card		
Test sequence	Step 1	Run the CN	
	Step 2	Run the gNB (O-DU - O-RU)	Use wireshark to verify the gNB-CN successful connection
	Step 3	Turn on the UE	Verify using UE logs and OAI O-DU logs the successful UE connection
Test Verdict	The UE connection ended properly, IP address assigned, and connection is stable (no connectivity loss with no data)		
Additional Resources	 <p>The diagram illustrates the 5G NR RACH (Random Access Channel) procedure between a UE (User Equipment) and a gNB (gNodeB). The procedure consists of the following steps:</p> <ol style="list-style-type: none"> <li>1. SIB1: The gNB transmits SIB1 to the UE.</li> <li>2. Acquire PRACH resource configuration: The UE acquires the PRACH resource configuration.</li> <li>3. Transmit PRACH Preamble with RA-RNTI: The UE transmits a PRACH preamble with RA-RNTI. A red arrow points to this step.</li> <li>4. MSG1 (Preamble, RA-RNTI) PRACH: The gNB receives the preamble and sends MSG1 back to the UE.</li> <li>5. Send DCI format 1_0 with RA-RNTI, allocation resource for RAR: The gNB sends DCI format 1_0 with RA-RNTI and allocation resource for RAR.</li> <li>6. MSG2 (TC-RNTI, TA, UL-Grant) PDCCH&amp;PDSCH: The gNB sends MSG2 (TC-RNTI, TA, UL-Grant) over PDCCH&amp;PDSCH.</li> <li>7. MSG3 (UEID) PUSCH: The UE sends MSG3 (UEID) over PUSCH.</li> <li>8. Send DCI format 1_0 with TC-RNTI, allocation resource for CR: The gNB sends DCI format 1_0 with TC-RNTI and allocation resource for CR.</li> <li>9. MSG4 (UEID, ...) PDCCH&amp;PDSCH: The gNB sends MSG4 (UEID, ...) over PDCCH&amp;PDSCH.</li> </ol> <p>The diagram also shows the RAR Window and CR Timer on the UE side.</p>		

Test case name	1 COTS UE data rate measurement	Test Case id	Int-test-01-03
Test purpose	Network performance both for UL and DL with 1 COTS UE		
Configuration	Indoor E2E testing O-DU - O-RU - COTS UE		
Test tool	ping and iperf3		
KPI	ping measured in ms and data rate UDP + TCP in Mbps		
Components Involvement	Integrated O-DU and O-RU + COTS UE connected		



Pre-test conditions	O-RAN F1I protocol between O-DU and O-RU is validated O-DU and O-RU synchronized O-DU, O-RU 5G SA spectrum validated COTS UE provisioned with a test SIM card COTS UE successfully connected to the network		
Test sequence	Step 1	Run the ping from the CN machine to the IP address of the UE	
	Step 2	Run the iperf3 server in the UE app	
	Step 3	Run the Iperf3 client in the CN server targeting the UE IP address	note: UL iperf3 applies using option -R on client side
Test Verdict	Ping RTT measurement <10ms, iperf3 UL > 10Mbps, iperf3 DL > 30Mbps		
Additional Resources	<a href="https://linux.die.net/man/8/ping">https://linux.die.net/man/8/ping</a> <a href="https://iperf.fr/iperf-doc.php">https://iperf.fr/iperf-doc.php</a>		

<b>Test case name</b>	ORAN Interface validation between O-RU and O-DU	<b>Test Case id</b>	<b>Int-test-01-04</b>
Test purpose	Validation of ORAN interface according to ORAN Option 7.2		
Configuration	Indoor E2E testing using Eurescom O-DU ORAN Emulator and O-RU		
Test tool	Eurecom O-DU ORAN Emulator SW		
KPI	Two Way communication ( UL and DL ) Between RunEL O-RU and Eurecom O-DU ORAN Emulator		
Components Involvement	RunEL O-RU Eurecom O-DU ORAN Emulator installed on RunEL Server		
Pre-test conditions	Successful Installation of the ORAN Emulator SW on RunEL Server		
Test sequence	Step 1	Connect the O-DU Oran Emulator to RunEL O-RU via Ethernet Port with 10GBPS capacity	
	Step 2	Send DL stream from the Emulator to the O-RU	Check reception of the stream at the O-RU
	Step 3	Send UL stream from the O-RU to the O-DU ORAN emulator	Check reception of the stream at the O-DU emulator
Test Verdict	The test is successful when both UL and DL streams are received without error at both sides of the ORAN Interface. Following the successful test the O-DU ORAN emulator will be replaced with a full O-DU protocol stack from Eurecom		
Additional Resources	None		

#### 4.2.1.1 Test Case Results Int-test-01-01 to Int-test-01-04

The final results of the Integration test between the Eurecom O-DU and the RunEL O-RU over ORAN Interface (Option 7.2) have not been completed at the publication date of this document due to the plan change at Eurecom that started the ORAN integration of the Eurecom O-DU



with other third party O-RUs (Foxconn and Mavenir) and delayed the Integration with the RunEL O-RU to a later date (planned to August 2022).

#### 4.2.2 DU-CU

These tests focus on the F1 mid-haul interface and interaction between O-CU and O-DU of the O-RAN gNB network functions.

Test case name	gNB DU to CU attachment		Test Case id	Int-test-02-01
Test purpose	Setup and validate the attachment between O-RAN DU and CU			
Configuration	Network configured according to the network plan. CU-CP started, creating CU-UP. CU discovered and monitored by dRAX SMO dashboard			
Test tool	dRAX SMO dashboard and wireshark			
KPI	-			
Components Involvement	ACC CU (disaggregated into CU-CP and CU-UP) and EUR OAI DU			
Pre-test conditions	dRAX RIC+CU installed. RU configured and powered on			
Test sequence	Step 1	Start dRAX via Helm chart (RIC, CU and SMO dashboard). Ensure CU is discovered and display on dRAX dashboard as unconnected		
	Step 2	Start the DU. Verify F1 attachment signalling. Verify gNB now active		
Test Verdict	Wireshark messages show F1 exchange. gNB now in active transmission mode. This is demonstrated in 2.2.1.1 for the EuCNC demonstration video, more specifically Figure 5			
Additional Resources	None			

Test case name	gNB DU to CU performance		Test Case id	Int-test-02-02
Test purpose	Validate O-RAN DU and CU at peak performance, ensuring no packet loss and acceptable CPU load			
Configuration	Network configured according to the network plan. O-RAN gNBs started and attached to 5GC. UE's attached to 5GS and generating maximum traffic			
Test tool	dRAX SMO dashboard and wireshark E2E iperf3 ensuring throughput without packet loss			
KPI	Max throughput without packet loss for configure 5G-NR spectrum			
Components Involvement	ACC CU, EUR OAI DU, RunEL RU, ATH 5GC, commercial UEs. Performance application running in UEs and DN behind UPF.			
Pre-test conditions	O-RAN CU/DU/RU running, attached to running 5GC Performance test server (iperf3 or librespeed) setup in DN behind UPF. Performance test clients installed in UE (e.g., smartphone) or behind 5G modem in CPE			
Test sequence	Step 1	Attach UEs to 5GS. Run ping to ensure IP connectivity		
	Step 2	Start performance tests.		

		Measure throughput & delay and note packet loss
	Step 3	In 5GC UPF, CU-UP and DU, analyze CPU and memory system load. Check system logs for errors/warning on packet drops or other performance related issues.
	Step 4	Identify performance bottlenecks to see if any performance tweaks or optimizations are possible to achieve expected maximum throughputs or eliminate incorrect packet loss.
Test Verdict	Still to be executed when OAI DU and REL RU integrated into the 5GS in Castellolí/Málaga	
Additional Resources	None	

#### 4.2.3 CU – 5G Core

These tests focus on the N2/N3 backhaul interface and interaction between the 5GC and the O-RAN gNB, specifically the O-CU network function.

Test case name	gNB-5GC attachment		Test Case id	Int-test-03-01
Test purpose	Setup and validate the attachment between gNB and 5GC (AMF)			
Configuration	5G core VM default installation and configuration. Network configured according to the network plan.			
Test tool	5GC web GUI, tcpdump for attachment confirmation messages			
KPI	Functional test			
Components Involvement	Athonet 5GC and gNB			
Pre-test conditions	The 5G core VM is installed and running on the Hypervisor, network is configured and reachable. gNB is configured and reachable			
Test sequence	Step 1	Configure the network interfaces and the CP, including all the related NFs with a default configuration		
	Step 2	Set the IP address of the gNB in the whitelist (by using web GUI)		
	Step 3	Configure the N2 interface for interconnection between AMF and gNB		
	Step 4	Attach the gNB to the 5GC AMF		
Test Verdict	Log messages show gNB successfully attached to the AMF. Pcap traces show the successful attachment of the gNB with the 5GC. The success of the attachment of gNB to 5GC and subsequent UE Internet connection over the 5G SNPN is described with screenshot evidence in 2.2.1.1 for the EuCNC demonstration video.			
Additional Resources	None			

#### 4.2.4 Slice Manager – Orchestrator

The following tests cases verify the integration between the orchestrator and the slice manager. It proves that the orchestrator is using the REST API interfaces provided by the Slice

manager, to allow the user of the orchestrator to create and delete slices, in particular the first test shows how the orchestrator creates the RAN chunks of the slice, and the second test how that RAN configuration is deleted.

Test case name	Slice Manager – Orchestrator Connectivity		Test Case id	Int-test-04-01
Test purpose	Validate that the Orchestrator connects successfully with the Slice Manager.			
Configuration	Slice Manager and Orchestrator up and running.			
Test tool	Orchestrator – Slice Manager (API).			
KPI	Functional Test			
Components Involvement	Orchestrator – Slice Manager.			
Pre-test conditions	A pre-created user on the Slice Manager.			
Test sequence	Step 1	Orchestrator: Access the Slice Manager through a GET Request from the Orchestrator.		
	Step 2	Slice Manager: Response with the pre-created user information through a json body.		
	Step 3	Orchestrator: Receive the request and check the response received.		
Test Verdict	Response status code is 200.			
Additional Resources	Request example:  curl -X GET "http://<IP>:8989/api/v1.0/user" -H "accept: application/json" Response example: <pre>[   {     "name": "Affor User",     "email": "affor.user@example.com",     "id": "5b62faa058f5682e3b0da3f1"   } ]</pre>			

For the second test, the initialization is tested between the Orchestrator and the Slice Manager using an API REQUEST. The Orchestrator sends with the GET request the information from the Slice Manager and then answers sending such information to the Orchestrator. With that information, the Orchestrator selects the proper parameters and then requests a POST command for the initialization of specific RAN parameters. With this request, the Slice Manager creates RAN slice subnets and sends an acknowledge to the Orchestrator.

<b>Test case name</b>	Slice Manager – Orchestrator - radio	<b>Test Case id</b>	<b>Int-test-04-02</b>
Test purpose	Radio Service Initialization between the Orchestrator and Slice Manager.		
Configuration	Slice Manager and Orchestrator up and running. Racoon needs to interact with the other infrastructure.		
Test tool	Orchestrator – Slice Manager (API).		

KPI	Functional Test	
Components Involvement	Orchestrator – Slice Manager.	
Pre-test conditions	A pre-created user on the Slice Manager. RAN infrastructure registered in Slice Manager through Raccoon.	
Test sequence	Step 1	Orchestrator: GET Request of the RAN topology to the Slice Manager.
	Step 2	Slice Manager: Response with the pre-created topology information through a json body.
	Step 3	Orchestrator: Receive the response and selects the parameters based on the information received.
	Step 4	Orchestrator: POST Request of the RAN Service Initialization to the Slice Manager.
	Step 5	Slice Manager: Receives the POST Request from the orchestrator and creates the RAN slice subnet based on the information from the Orchestrator.
Test Verdict	Response status GET code is 200.	
Additional Resources	<p>Request POST example:</p> <pre>curl -X POST</pre> <pre>"http://0.0.0.0:8989/api/v1.0/ran_infrastructure/629f6b4746ebfc9ec31e76c2/ran_slice_subnet" -H "accept: application/json" -H "Content-Type: application/json" -d '{"user_id":"60ba00e06533d50001a2df41","name":"aws-rc-2","chunk_topology":{"selectedPhys":[{"id":"76344247-5896-4359-9b96-227e6cd3322f","name":"primaryPLMN","type":"ACCELERAN_CELL","config":{"earfcnd1":41690,"phyCellId":512,"refSignalPower":-7}}],"coreAddress":"172.16.0.10","corePort":8088,"plmnId":"00103","vlanId":1500}'</pre> <p>Response example:</p> <pre>{   "name": "aws-rc-7",   "user_id": "60e5b579ed3e7c0001677920",   "id": "5b62faa058f5682e3b0da3f1"   "radio_chunk": {     "id": "5b63089158f568073093f70d",     "ran_infrastructure_id": "610bf39938283c0001954db2",     "chunk_topology": {       "physicalInterfaceList": [         {           "id": "76344247-5896-4359-9b96-227e6cd3322f",           "name": "primaryPLMN",           "type": "ACCELERAN_CELL",           "config": {             "earfcnd1": 41690,             "phyCellId": 512,             "refSignalPower": -7           }         }       ]     }   },   "radio_service": {     "id": "60ed58b51bf21f00013e7cc5",     "coreAddress": "172.16.0.10",     "corePort": "8088",     "plmnId": "00103",     "vlanId": 1500   } }</pre>	

This integration test case shows how the orchestrator connects with the Slice Manager API. Figure 43 shows the logs of the orchestrator, these lines show how the Slice Manager replies to a GET request from the orchestrator.

```
2021-11-12 14:25:27: INFO    running GET ran_infrastructure request to http://84.88.36.223:5000/api/v0.1/ran_infrastruct
ure?user_id=6144931231c04b0001db9619
2021-11-12 14:25:27: INFO    [{"name": 'Castelloli', 'id': '6176c18be252f00001734288', 'ran_infrastructure_data': {'cont
roller_url': 'http://192.168.124.177:8008/', 'username': 'user1', 'password': 'pass123', 'status': 'reachable', 'quota': {
'available_plmnids': ['00102', '00103', '00104', '00105', '00106', '00101']}}, 'user_id': '6144931231c04b0001db9619', 'loc
ation': {'latitude': 41.39, 'longitude': 2.15}}]
```

Figure 43 NearbyOne orchestrator logs showing a GET request to the Slice Manager

This integration test case shows how the orchestrator uses the Slice Manager API to run a POST request creating a RAN slice.

Figure 44 shows the final POST of the interaction between these 2 components, in earlier steps the components resolve all the required configuration and ids needed to define the slice resources.

The first line of the logs shows the POST request, the second one shows the body being sent, and the next two lines are showing the response sent by the Slice Manager and the radio\_chunk and radio\_service\_ids that need to be stored in order to be able to undeploy that ran chunk of the slice.

```
2021-11-12 14:25:28: INFO    running POST request to http://84.88.36.223:5000/api/v0.1/ran_infrastructure/6176c18be252f0
0001734288/aff5gslice
2021-11-12 14:25:28: INFO    body: {"user_id": "6144931231c04b0001db9619", "name": "c044ee00-444d-437b-8c8d-3b9be19ac7",
"chunk_topology": {"physicalInterfaceList": [{"id": "d6080766-4d6d-4790-8e3d-faca2fc04b84", "name": "primaryPLMN", "typ
e": "ACCELLERAN_CELL", "config": {"earfcn": 42590, "refSignalPower": -8, "physicalCellId": 512}}]}, "coreAddress": "10.1
7.7.31", "corePort": "36412", "plmnId": "00101", "vlanId": "1"}
2021-11-12 14:25:30: INFO    {"user_id": "6144931231c04b0001db9619", "name": "c044ee00-444d-437b-8c8d-3b9be19ac7", 'ra
n_infrastructure_id': '6176c18be252f00001734288', 'ran_controller_id': '14827b5c-3136-4a93-a13b-fe4c378aff1a', 'radio_chun
k_id': '618e7958e252f0000173430f', 'cellular_config': {}, 'radio_service_id': '618e795ae252f00001734311'}
2021-11-12 14:25:30: INFO    radio_chunk: 618e7958e252f0000173430f radio_service_id: 618e795ae252f00001734311
```

Figure 44 NearbyOne Orchestrator logs showing POST creating a ran slice in Slice Manager

#### 4.2.5 AIMA – Telemetry - Orchestrator

With the increasing heterogeneity of mobile networks resource, orchestration tasks are becoming more challenging. In such scenario, Machine Learning (ML)-based network reconfiguration techniques play a key role to obtain an optimal trade-off between network performance and the utilization of computing resources.

Therefore, this subsection aims to showcase the integration tests performed in order to verify the proper integration between the Telemetry component (Prometheus), the AI/ML Framework and the NearbyOne Orchestrator that enables the execution of a CPU prediction algorithm to optimize the CPU resource allocation of network slices in the Affordable5G ecosystem.

##### 4.2.5.1 AI/ML Framework integration with Prometheus

This integration test case demonstrates the proper integration between the AI/ML Framework Message Broker and the Telemetry component (Prometheus). The CPU prediction AI/ML model requires two metrics for the optimization of the CPU utilization of a given container: i) the CPU utilization of the container (container\_cpu\_usage\_seconds\_total) and ii) the transmitted bytes by the container (container\_network\_transmit\_bytes\_total).

Test case name	Prometheus integration	Test Case id	Int-test-05-01
----------------	------------------------	--------------	----------------

Test purpose	Verify the correct integration of the Prometheus instance with the AI/ML Framework through the Message Broker		
Configuration	Prometheus deployed AI/ML Framework deployed Message Broker deployed CPU prediction model deployed in the AI/ML Framework		
Test tool	Message broker logs, CURL		
KPI	-		
Components Involvement	Prometheus, AI/ML Framework		
Pre-test conditions	Prometheus instance running Prometheus collecting the required metrics AI/ML Framework installed and running Message Broker installed and running Message Broker configured with configuration file CPU prediction model deployed in the AI/ML Framework		
Test sequence	Step 1	Check logs of the message broker	
	Step 2	Check if data is correctly gathered from Prometheus	
Test Verdict	The Message Broker is able to get the required metrics for the AI/ML model from Prometheus		
Additional Resources	Command: docker logs message broker		

The result of this integration test case is presented in Figure 45 and Figure 46, showing how these two metrics are successfully gathered by the Message Broker from the Telemetry component using the REST API of Prometheus.

```

{
  "status": "success",
  "data": {
    "resultType": "matrix",
    "result": [
      {
        "metric": {
          "name": "container_cpu_usage_seconds_total",
          "container": "iperf",
          "cpu": "total",
          "description": "kubelet cAdvisor exporter",
          "exported_name": "k8s_iperf_iperf-55d466c5f9-9h2p5_slice2_331aea9c-9b1d-4696-99fa-e3d833c2b7a8_146",
          "id": "/kubepods.slice/kubepods-pod331aea9c-9b1d-4696-99fa-e3d833c2b7a8.slice/docker-fc77d8f5b492f57d9812a916cb1c2128bca036eb005ab4e59583040dea4dc5d2.scope",
          "image": "registry.nearbycomputing.com/nbycomp/apps/iperf3@sha256:e730a361eda144ac640f37ceca8873df2834b052997aa05cbdf70acb11ff90d4",
          "instance": "b24a5b3a-ef62-4a3a-a7cc-28bfaf08de95:8080",
          "job": "discovery",
          "name": "kubelet-cadvisor",
          "namespace": "slice2",
          "pod": "iperf-55d466c5f9-9h2p5"
        },
        "values": [
          [
            1654611195.597,
            "0.051410689"
          ],
          [
            1654611200.597,
            "0.051410689"
          ],
          [
            1654611205.597,
            "0.051410689"
          ],
          [
            1654611210.597,
            "0.051410689"
          ],
          [
            1654611215.597,
            "0.051410689"
          ],
          [
            1654611220.597,
            "0.051410689"
          ],
          [
            1654611225.597,
            "0.051410689"
          ],
          [
            1654611230.597,
            "0.051410689"
          ],
          [
            1654611235.597,
            "0.051410689"
          ],
          [
            1654611240.597,
            "0.051410689"
          ],
          [
            1654611245.597,
            "0.051410689"
          ],
          [
            1654611250.597,
            "0.051410689"
          ],
          [
            1654611255.597,
            "0.051410689"
          ],
          [
            1654611260.597,
            "0.051410689"
          ],
          [
            1654611265.597,
            "0.051410689"
          ],
          [
            1654611270.597,
            "0.051410689"
          ]
        ]
      }
    ]
  }
}

```

Figure 45 Metric "container\_cpu\_usage\_seconds\_total" obtained from the Prometheus instance by the Message Broker



```

{
  "status": "success",
  "data": {
    "resultType": "matrix",
    "result": [
      {
        "metric": {
          "name": "container_network_transmit_bytes_total",
          "container": "POD",
          "description": "kubelet cAdvisor exporter",
          "exported_name": "k8s_POD_iperf-55d466c5f9-9h2p5_slice2_331aea9c-9b1d-4696-99fa-e3d833c2b7a8_665",
          "id": "/kubepods.slice/kubepods-pod331aea9c-9b1d-4696-99fa-e3d833c2b7a8.slice/docker-c73551be84fe80074189b90a5e37cd3f45eee486c7d0604a539b5fbd867b8f6.scope",
          "image": "k8s_gcr.io/pause:3.2",
          "instance": "b24a5b3a-ef62-4a3a-a7cc-28bfaf08de95:8080",
          "interface": "eth0",
          "job": "discovery",
          "name": "kubelet-cadvisor",
          "namespace": "slice2",
          "pod": "iperf-55d466c5f9-9h2p5"
        },
        "values": [
          [
            1654611195.597,
            "0"
          ],
          [
            1654611200.597,
            "0"
          ],
          [
            1654611205.597,
            "0"
          ],
          [
            1654611210.597,
            "0"
          ],
          [
            1654611215.597,
            "0"
          ],
          [
            1654611220.597,
            "0"
          ],
          [
            1654611225.597,
            "0"
          ],
          [
            1654611230.597,
            "0"
          ],
          [
            1654611235.597,
            "0"
          ],
          [
            1654611240.597,
            "0"
          ],
          [
            1654611245.597,
            "0"
          ],
          [
            1654611250.597,
            "0"
          ],
          [
            1654611255.597,
            "0"
          ],
          [
            1654611260.597,
            "0"
          ],
          [
            1654611265.597,
            "0"
          ],
          [
            1654611270.597,
            "0"
          ]
        ]
      }
    ]
  }
}

```

*Figure 46 Metric "container\_network\_transmit\_bytes\_total" obtained from the Prometheus instance by the Message Broker*

#### 4.2.5.2 AI/ML Framework interface to the Orchestrator

This integration test case shows how the output of the CPU prediction model is published in the Message Broker data bus and available to the rest of components (including the Orchestrator) through a RabbitMQ queue.

Test case name	AI/ML Framework interface to the Orchestrator	Test Case id	Int-test-05-02
Test purpose	Test the correct integration of the AI/ML Framework and the NearbyOne Orchestrator through the Message Broker		
Configuration	Prometheus deployed AI/ML Framework deployed Message Broker deployed CPU prediction model deployed in the AI/ML Framework		
Test tool	Message broker logs, testing script		
KPI	-		
Components Involvement	Prometheus, AI/ML Framework, Message Broker		
Pre-test conditions	Prometheus instance running Prometheus collecting the required metrics AI/ML Framework installed and running Message Broker installed and running Message Broker configured with configuration file CPU prediction model deployed in the AI/ML Framework		
Test sequence	Step 1	Check logs of the message broker	
	Step 2	Check if prediction is published in the queue	
Test Verdict	The AI/ML model prediction is successfully published by the Message Broker in the RabbitMQ and the prediction can be accessed from the Orchestrator		
Additional Resources	Command: docker logs message_broker		

The results are presented in the following figures. Firstly, Figure 47 shows the logs of the Message Broker container demonstrating that the predictions are properly published in the queue, then, Figure 48 demonstrate that the predictions can be accessed from a RabbitMQ client using a testing python script (receiver\_test.py).

```
> docker logs message_broker
[*] Model broker started. To exit press CTRL+C
[+] Model prediction: 3.06904913e-05
[+] Hysteresis: 0
[x] Model prediction (hysteresis) published: 0

[+] Model prediction: 3.06904913e-05
[+] Hysteresis: 0
[x] Model prediction (hysteresis) published: 0

[+] Model prediction: 3.06904913e-05
[+] Hysteresis: 0
[x] Model prediction (hysteresis) published: 0
```

Figure 47 Message Broker logs showing the publication of the prediction in the queue

```
> python3 receiver_test.py
[*] Waiting for messages. To exit press CTRL+C
[x] Received prediction: 0
[x] Received prediction: 0
[x] Received prediction: 0
```

Figure 48 Test script showing that the prediction is properly published in the queue

#### 4.2.5.3 Orchestrator integration with AI/ML Framework

This integration test case shows how the predictions published in the Message Broker data bus are collected by the orchestrator and re-introduced to Prometheus to be able to use them by the alarms/rules that will trigger slice resizing, app/CNF migrations, etc. It reuses the data that was already published to the message broker in Int-test-05-02.

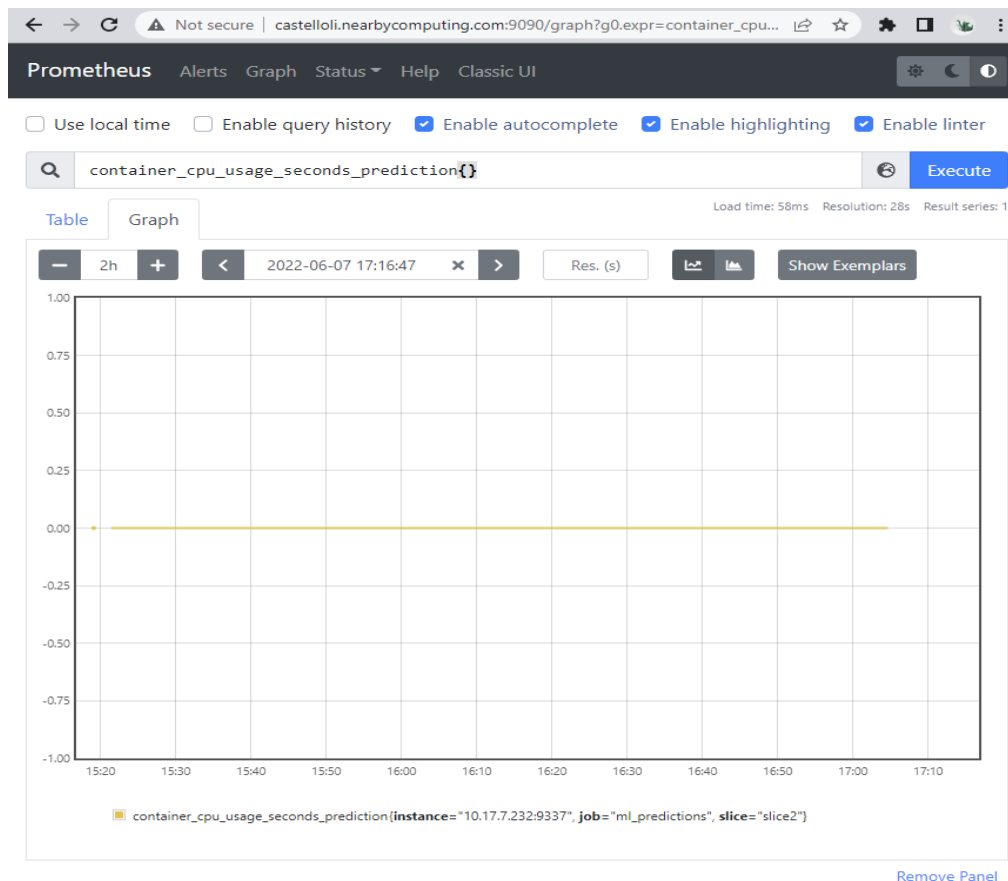
Test case name	Orchestrator reacting to KPI	Test Case id	Int-test-05-03
Test purpose	Verify the predictions introduced in the Message Broker are pushed to Prometheus and are available to define orchestration rules/alarms		
Configuration	Message Broker deployed Orchestrator Deployed Prometheus deployed		
Test tool	Testing script, Prometheus dashboard		
KPI	-		
Components Involvement	Message Broker, Prometheus, NearbyOne Orchestrator		
Pre-test conditions	Message Broker installed and running Message Broker configured with configuration file Prometheus instance running Prometheus collecting the required metrics Orchestrator instance running		
Test sequence	Step 1	Check data collected in Prometheus	
	Step 2	Check alarms status in Prometheus	

Test Verdict	The prediction publication is collected in the aggregated Prometheus database and can be used to define alarms/rules of the Orchestrator
Additional Resources	None

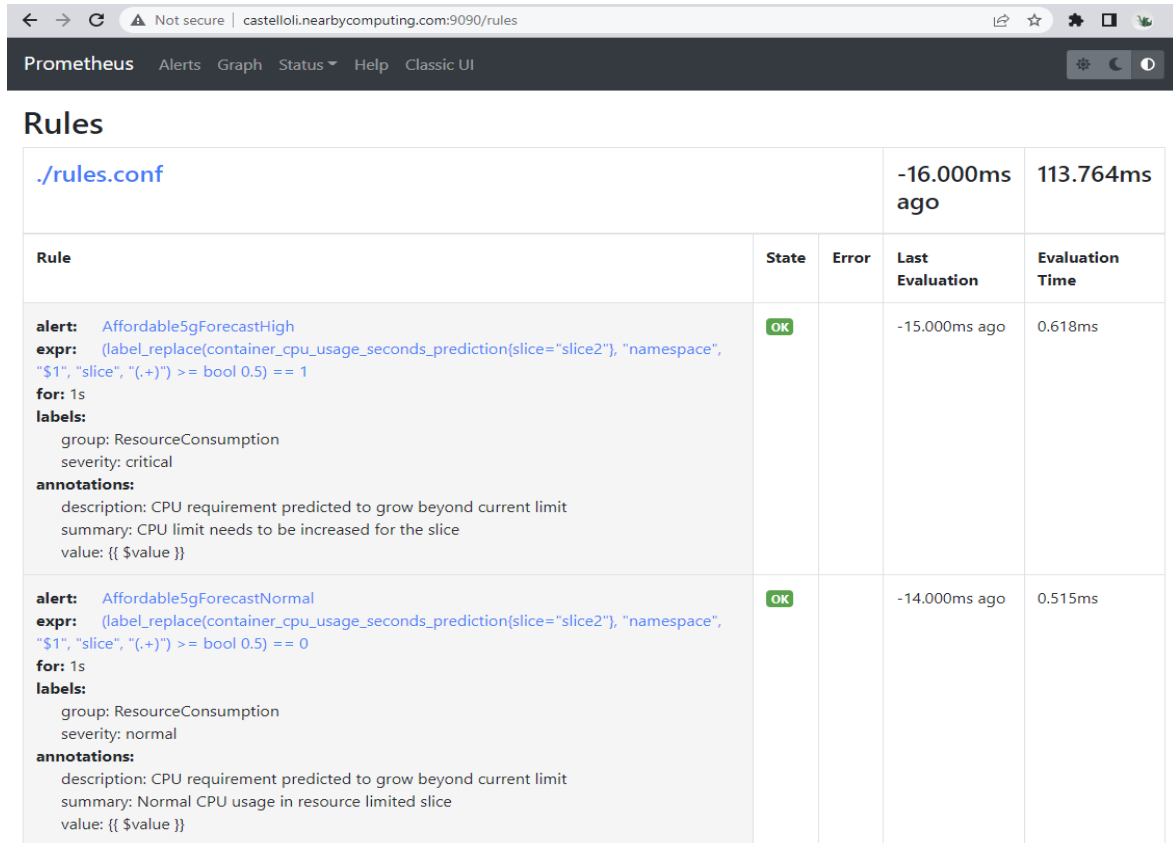
The results are presented in the following lines. In first place, Figure 49 shows in the NearbyOne Prometheus dashboard how the data has been exported from the Message broker to the existing Prometheus database where all KPIs are aggregated, and the rules/alerts used to trigger slice migration/resizing are defined.

Secondly, Figure 50 shows 2 rules that are used to monitor the predictions. E.g., when a slice has been defined with a Service Level Agreement (SLA) associated to the Affordable5GForecastHigh rule, it will increase the resources assigned to that slice as soon as that alarm triggers.

Lastly, Figure 51 shows the current status of these alarms.



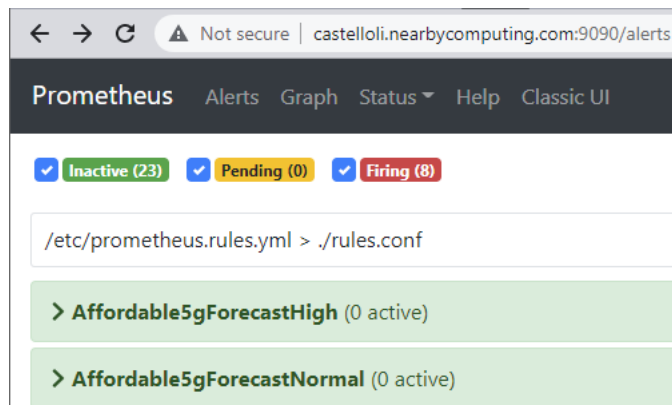
*Figure 49 Prometheus dashboard showing the prediction data that was published in the Message Broker*



The screenshot shows the Prometheus Rules dashboard. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', 'Help', and 'Classic UI'. Below this, the 'Rules' section is active, showing a list of rules. The first rule is 'Affordable5gForecastHigh' and the second is 'Affordable5gForecastNormal'. Both rules are in a 'State' of 'OK' and have no errors. The 'Last Evaluation' for the first rule was 15.000ms ago, and for the second, it was 14.000ms ago. The 'Evaluation Time' for the first rule was 0.618ms, and for the second, it was 0.515ms. The rules are defined in './rules.conf'.

Rule	State	Error	Last Evaluation	Evaluation Time
<b>alert:</b> Affordable5gForecastHigh <b>expr:</b> (label_replace(container_cpu_usage_seconds_prediction{slice="slice2"}, "namespace", "\$1", "slice", "(.+)") >= bool 0.5) == 1 <b>for:</b> 1s <b>labels:</b> group: ResourceConsumption severity: critical <b>annotations:</b> description: CPU requirement predicted to grow beyond current limit summary: CPU limit needs to be increased for the slice value: {{ \$value }}	OK		-15.000ms ago	0.618ms
<b>alert:</b> Affordable5gForecastNormal <b>expr:</b> (label_replace(container_cpu_usage_seconds_prediction{slice="slice2"}, "namespace", "\$1", "slice", "(.+)") >= bool 0.5) == 0 <b>for:</b> 1s <b>labels:</b> group: ResourceConsumption severity: normal <b>annotations:</b> description: CPU requirement predicted to grow beyond current limit summary: Normal CPU usage in resource limited slice value: {{ \$value }}	OK		-14.000ms ago	0.515ms

Figure 50 Prometheus dashboard showing the rules defined monitoring the predictions



The screenshot shows the Prometheus Alerts dashboard. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', 'Help', and 'Classic UI'. Below this, the 'Alerts' section is active, showing a summary of alert states: 'Inactive (23)', 'Pending (0)', and 'Firing (8)'. Below the summary, there's a list of alerts. The first alert is 'Affordable5gForecastHigh' and the second is 'Affordable5gForecastNormal'. Both alerts are in a 'State' of 'Inactive' and have no errors. The 'Last Evaluation' for the first alert was 15.000ms ago, and for the second, it was 14.000ms ago. The 'Evaluation Time' for the first alert was 0.618ms, and for the second, it was 0.515ms. The alerts are defined in './rules.conf'.

Alert	State	Error	Last Evaluation	Evaluation Time
<b>Alert:</b> Affordable5gForecastHigh <b>Expr:</b> (label_replace(container_cpu_usage_seconds_prediction{slice="slice2"}, "namespace", "\$1", "slice", "(.+)") >= bool 0.5) == 1 <b>For:</b> 1s <b>Labels:</b> group: ResourceConsumption severity: critical <b>Annotations:</b> description: CPU requirement predicted to grow beyond current limit summary: CPU limit needs to be increased for the slice value: {{ \$value }}	Inactive		-15.000ms ago	0.618ms
<b>Alert:</b> Affordable5gForecastNormal <b>Expr:</b> (label_replace(container_cpu_usage_seconds_prediction{slice="slice2"}, "namespace", "\$1", "slice", "(.+)") >= bool 0.5) == 0 <b>For:</b> 1s <b>Labels:</b> group: ResourceConsumption severity: normal <b>Annotations:</b> description: CPU requirement predicted to grow beyond current limit summary: Normal CPU usage in resource limited slice value: {{ \$value }}	Inactive		-14.000ms ago	0.515ms

Figure 51 Prometheus dashboard showing the triggering alarms

#### 4.2.6 Telemetry Execution in Far Edge Devices

The ML reconfiguration techniques mentioned in the previous section will be also deployed in THI platform. THI has developed a FPGA prototype equipped with NEOX accelerator [11]. Apart from the hardware IP, THI will also utilize the NEOX SDK for optimizing the Convolutional Neural Networks (CNN) models in terms of memory footprint and execution time. Prime targets for customization are the number of cores, the number of threads per core, the wide of the vector processing lanes and the memory resources (private and shared caches as well as the cache prefetching options). NEOX will be provided in a fully functional FPGA prototype based on ZYNQ platforms (NEMA) [12]. ZYNQ FPGAs contain (apart of the FPGA programmable

logic) a dual core A9 ARM processor in which a regular Linux operating system have been ported. In this way, the communication of the remaining computational and network components can be performed with standard Linux processes.

The goal is to explore how far at the edge such ML based telemetry functionality can be deployed. Far edge is characterized by devices with scarce resources (both in terms of memory and computational capabilities) and also devices operating under tight power constraints).

Therefore, the target is to showcase that the required performance can be achieved (in terms of ms) but under a very tight memory and power constraints. To achieve these goals, the NEOX accelerator will be configured to execute the ML based telemetry modules and also the NEOX AI-SDK will be used to compress the ML models using two different compression techniques (quantization to int8 arithmetic and low-rank factorization).

Test case name	Telemetry integration in THI platform	Test Case id	Int-test-06-01
Test purpose	Verify the operation of the ML-based telemetry module in far edge platform		
Configuration	The ML model of the telemetry module has been verified that is compatible with the NEOX AI-SDK deployment framework. The ML model of the telemetry module has been verified that is compatible with the NEOX AI-SDK compression framework. NEOX Bits FPGA platform with NEOX accelerator is released and its correct operation has been verified.		
Test tool	NEOX Bits FPGA platform		
KPI	-		
Components Involvement	Telemetry module NEOX AI-SDK deployment framework NEOX AI-SDK compression framework NEOX Bits FPGA platform		
Pre-test conditions	AI/ML Framework installed and running in x86 platform. AI/ML Framework installed and running in ARM platform. NEOX AI-SDK deployment framework is installed and running. NEOX AI-SDK compression framework is installed and running. NEOX Bits FPGA platform is installed and running.		
Test sequence	Step 1:	ML model of telemetry module is executed in x86 machines	
	Step 2:	ML model of telemetry module is executed in ARM machines	
	Step 3:	ML model of telemetry module is compressed using quantization to int8 numbers	
	Step 4:	ML model of telemetry module is further compressed using Low-Rank Factorization (LRF) technique	
	Step 5:	ML model of telemetry module is analyzed by the NEOX AI-SDK deployment framework	
	Step 6:	ML model of telemetry module is deployed to NEOX accelerator	
	Step 7:	Correct operation is validated by comparing the gathered logs to the one generated in a x86 machine	
Test Verdict	Telemetry module is properly integrated in THI FPGA platform and effectively parallelized in at least 32 threads, while the ML		

	based model is compressed by a factor of 5x compared to the initial size of the model. The execution of each inference step is executed in less than 500ms at a power consumption less than 8mWatts.
--	--

#### 4.2.7 Non-RT RIC (RIC manager) – dRAX

We define a series of tests to validate the use of dRAX 5G and the deployment of the Telemetry xApp using RIC Manager, as discussed in section 3.2.3. These tests include:

- Registration of Mocked Non-RT dRAX 5G.
- Registration of dRAX 5G.
- Deployment of the Telemetry xApp.
- Creation of dRAX's Policy Type.
- Creation of dRAX's Policy Instance.
- List all the information stored about dRAX (Policy Instances, xApps)
- Deletion of dRAX's Policy Type and all associated Policy Instances.
- Undeployment of the Telemetry xApp.

For easy reproducibility of these tests, we provide a Swagger GUI, which can be found at: [https://<ric\\_manager\\_ip>:8080/docs](https://<ric_manager_ip>:8080/docs)

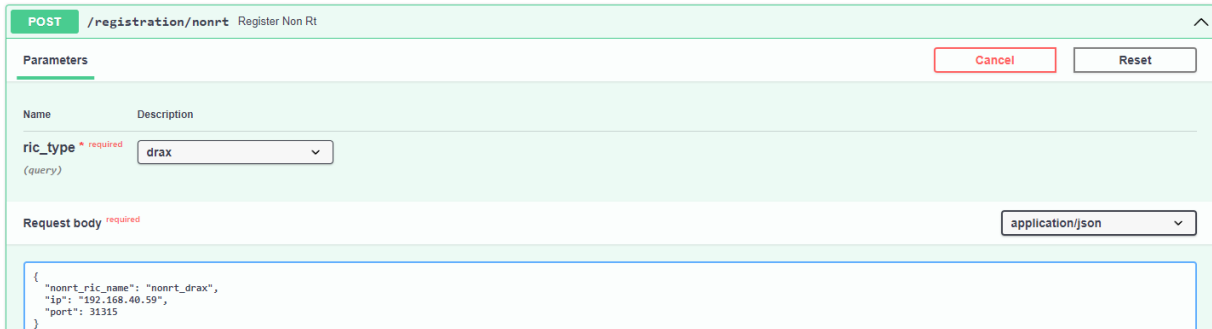
##### 4.2.7.1 Registration of Mocked Non-RT dRAX 5G

To follow the O-RAN standard, we have developed a Non-RT dRAX (as described in Deliverable 3.2) that implements all the logic that would be part of an O-RAN Non-RT RIC. To register it, it is only required to provide a friendly name and the same IP address and dRAX's API port as the regular dRAX. RIC-Manager's API also provided a "ric\_type" field to indicate which kind of implementation is going to be registered. The only two currently supported options are ORAN and dRAX. The following table and present the steps of the test and a screenshot with the execution.

Test case name	Registration of Mocked Non-RT dRAX 5G		Test Case id	Int-test-07-01
Test purpose	Register a mocked Non-RT dRAX in RIC-Manager			
Configuration	RIC-Manager and dRAX up and running.			
Test tool	Swagger			
KPI	-			
Components Involvement	RIC Manager and dRAX			
Pre-test conditions	RIC-Manager has connectivity to the dRAX.			
Test sequence	Step 1	Fetch the dRAX's REST API IP address and port.		
	Step 2	Use the information fetched to perform a POST request to the RIC-Manager endpoint "/registration/nonrt"		
	Step 3	In the request's body, use "drax" for the "ric_type" field and select a name to identify the mocked Non-RT dRAX with in the "nonrt_ric_name" field.		
	Step 4	Send the request and check the response received.		
Test Verdict	Response status code is 200.			



Additional Resources	Request example: <pre>{   "ric_type": "drax",   "nonrt_ric_name": "nonrt_drax",   "ip": "192.168.40.74",   "port": 31315 }</pre>
----------------------	--



POST /registration/nonrt Register Non Rt

Parameters

ric\_type \* required (query) drax

Request body required application/json

```
{
  "nonrt_ric_name": "nonrt_drax",
  "ip": "192.168.40.59",
  "port": 31315
}
```

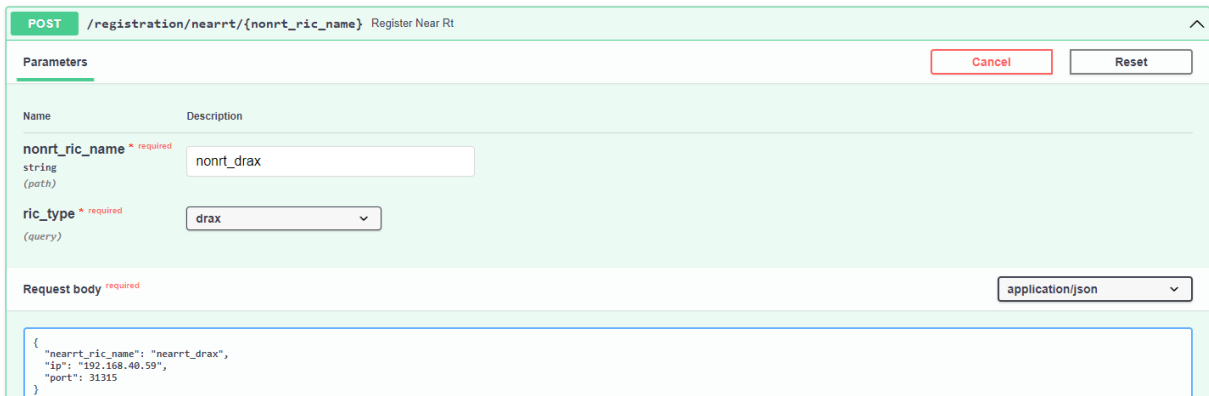
Figure 52 Screenshot with the execution of test "Registration of Mocked Non-RT dRAX 5G"

#### 4.2.7.2 Registration of dRAX 5G

Once the Non-RT dRAX is registered, we can register the dRAX. To do so, we specify the Non-RT friendly name provided in the previous step, to which the dRAX will be associated, as well as a friendly name for the dRAX and the IP and port. The following table and Figure 53 present the steps of the test and a screenshot with the execution.

Test case name	Registration of dRAX5G	Test Case id	Int-test-07-02
Test purpose	Register dRAX 5G in RIC-Manager		
Configuration	RIC-Manager and dRAX up and running.		
Test tool	Swagger		
KPI	-		
Components Involvement	RIC manager and dRAX		
Pre-test conditions	RIC-Manager has connectivity to the dRAX.		
Test sequence	Step 1	Fetch the dRAX's REST API IP address and port.	
	Step 2	Use the information fetched to perform a POST request to the RIC-Manager endpoint "/registration/neart/{nonrt_ric_name}", where "nonrt_ric_name" is the registration name given to the "fake" Non-RT RIC registered during Int-Test-07-01.	
	Step 3	In the request's body, use "drax" for the "ric_type" field and select a name to identify the dRAX with in the "neart_ric_name" field.	
	Step 4	Send the request and check the response received.	
Test Veredict	Response status code is 200.		
Additional Resources	Request example:		

	<pre>{   "ric_type": "drax",   "nearrt_ric_name": "nearrt_drax",   "ip": "192.168.40.74",   "port": 31315 }</pre>
--	---



POST /registration/nearrt/{nonrt\_ric\_name} Register Near Rt

Parameters

Name Description

nonrt\_ric\_name \* required  
string  
(path) nonrt\_drax

ric\_type \* required  
(query) drax

Request body \* required  
application/json

```
{
  "nearrt_ric_name": "nearrt_drax",
  "ip": "192.168.40.59",
  "port": 31315
}
```

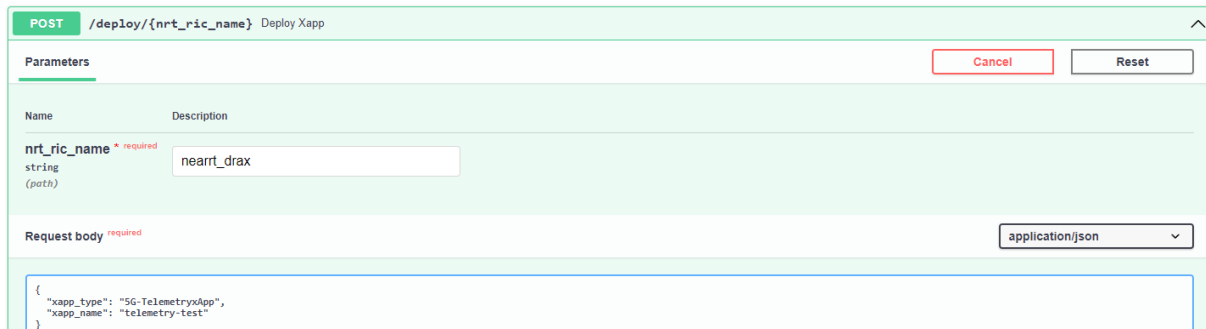
Figure 53 Screenshot with the execution of test "Registration of dRAX 5G"

#### 4.2.7.3 Deployment of the Telemetry xApp

To deploy the Telemetry xApp we need to specify the xApp in which the xApp is going to be deployed. The request's body expects two parameters: "xapp\_type" and "xapp\_name". To deploy the Telemetry xApp in a 5G dRAX, we set the value of "xapp\_type" to be "5G-TelemetryxApp". The "xapp\_name" is an identifier to distinguish the exact instance of the xApp that is going to be deployed. The following table and figure present the steps of the test and a screenshot with the execution.

Test case name	xApp Deployment Test	Test Case id	Int-test-07-03
Test purpose	Deploy an xApp on dRAX using RIC-Manager.		
Configuration	RIC-Manager and dRAX up and running.		
Test tool	Swagger		
KPI	-		
Components Involvement	RIC manager and dRAX		
Pre-test conditions	RIC-Manager has connectivity to the dRAX.		
Test sequence	Step 1	Select an xApp type from the list of available ones	
	Step 2	Perform a POST request to the RIC-Manager endpoint "/registration/nonrt".	
	Step 3	Within the request's body include the xApp type and a name to identify this particular instance.	

	Step 4	Check the response to confirm whether the xApp has been properly deployed.
Test Verdict	Response status code is 200.	
Additional Resources	Request example: <pre>{   "xapp_type": "TelemetryxApp",   "xapp_name": "Affordable-Telemetry" }</pre>	



POST /deploy/{nrt\_ri\_name} Deploy Xapp

Parameters

Name	Description
nrt_ri_name * required string (path)	nearrt_drax

Request body required

application/json

```
{
  "xapp_type": "5G-TelemetryxApp",
  "xapp_name": "telemetry-test"
}
```

Figure 54 Screenshot with the execution of test "Deployment of the Telemetry xApp"

#### 4.2.7.4 Creation of dRAX's Policy Type

To create the xApp's Policy Type, we need to specify the destination Near-RT RIC and pass as the request's body the policy type of the xApp (see Figure 55).

Test case name	Creation of dRAX's Policy Type	Test Case id	Int-test-07-04
Test purpose	Create a policy type for a Near-RT RIC using RIC-Manager.		
Configuration	RIC-Manager and dRAX up and running.		
Test tool	Swagger		
KPI	-		
Components Involvement	RIC Manager and dRAX		
Pre-test conditions	RIC-Manager has connectivity to the dRAX.		
Test sequence	Step 1	Create a policy type following the ORAN standard.	
	Step 2	Perform a POST request to the RIC-Manager endpoint "/policy_type/{nearrt_ric_name}", where "nearrt_ric_name" is the name given to the dRAX during registrartion.	
	Step 3	Include the policy type in the body of the request.	
	Step 4	Send the request and check the response received.	
Test Verdict	Response status code is 201.		
Additional Resources	Request example:		

	<pre>{   "name": "telemetry-xapp_policy",   "description": "Policy type for dRAX's telemetry-xapp",   "policy_type_id": 20,   "create_schema": {     "\$schema": "http://json-schema.org/draft-07/schema#",     "title": "telemetry-xapp_policy",     "description": "Policy type for dRAX's telemetry-xapp",     "type": "object",     "properties": {       "telemetry_topics": {         "type": "string"       },       "update_interval": {         "type": "integer"       },       "aws_access_key_id": {         "type": "string"       },       "aws_secret_access_key": {         "type": "string"       },       "additionalProperties": false     }   } }</pre>
--	---

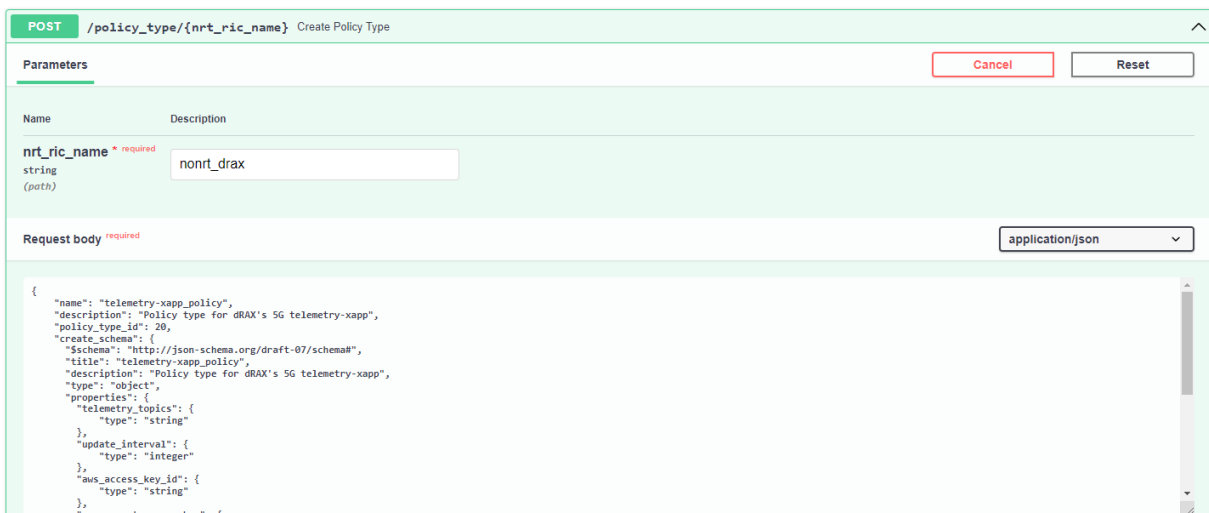


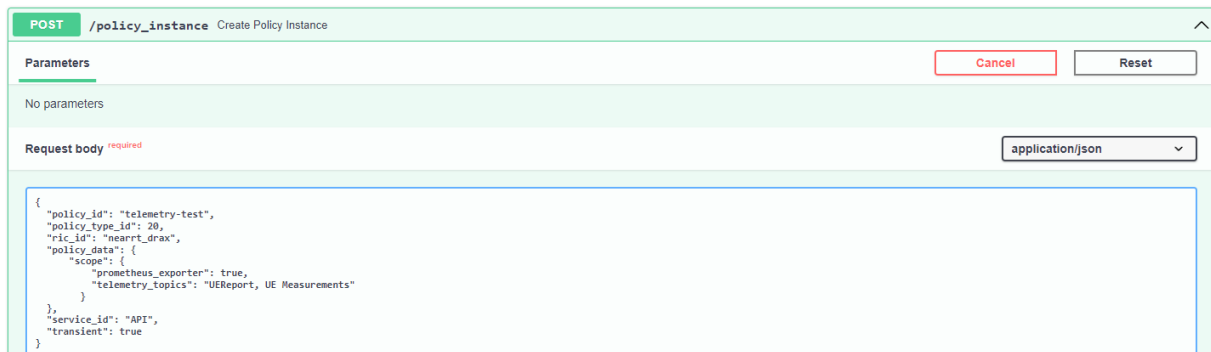
Figure 55 Screenshot with the execution of test "Creation of dRAX's Policy Type"

#### 4.2.7.5 Creation of dRAX's Policy Instance

Once the Policy Type has been created, we can proceed to create the Policy Instance to set the values of the desired fields. Note that this request does not have a RIC as destination since all the required information has been defined in the policy. The following table and figure present the steps of the test and a screenshot with the execution.

Test case name	Creation of dRAX's Policy Instance	Test Case id	Int-test-07-05
Test purpose	Create a policy instance for a Near-RT RIC using RIC-Manager.		
Configuration	RIC-Manager and dRAX up and running.		
Test tool	Swagger		
KPI	-		
Components Involvement	RIC Manager and dRAX		
Pre-test conditions	There is a policy type created in the Near-RT RIC associated with the policy instance that is about to be created.		
Test sequence	Step 1	Create a policy instance following the ORAN standard.	

	Step 2	Perform a POST request to the RIC-Manager endpoint "/policy_instance".
	Step 3	Include the policy instance in the body of the request. In the field "policy_id" include the name of the xApp to which the configuration of the policy will be applied.
	Step 4	Send the request and check the response received.
	Test Verdict	Response status code is 201.
Additional Resources	Request example: <pre>{   "policy_id": "Affordable-Telemetry",   "policytype_id": "20",   "ric_id": "nearrt_drax",   "policy_data": {     "scope": {       "telemetry_topics": "serviceFound, beaconInfo",       "update_interval": 15,       "aws_access_key_id": "AKIASQJG6UFUL76WQH",       "aws_secret_access_key": "k452+n5ZL1m2QnvwacpX+PzPgb/mnYdGx/n13+Pf"     }   },   "service_id": "API",   "transient": true }</pre>	



POST /policy\_instance Create Policy Instance

Parameters: No parameters

Request body required: application/json

```
{
  "policy_id": "telemetry-test",
  "policy_type_id": 20,
  "ric_id": "nearrt_drax",
  "policy_data": {
    "scope": {
      "prometheus_exporter": true,
      "telemetry_topics": "UEReport, UE Measurements"
    }
  },
  "service_id": "API",
  "transient": true
}
```

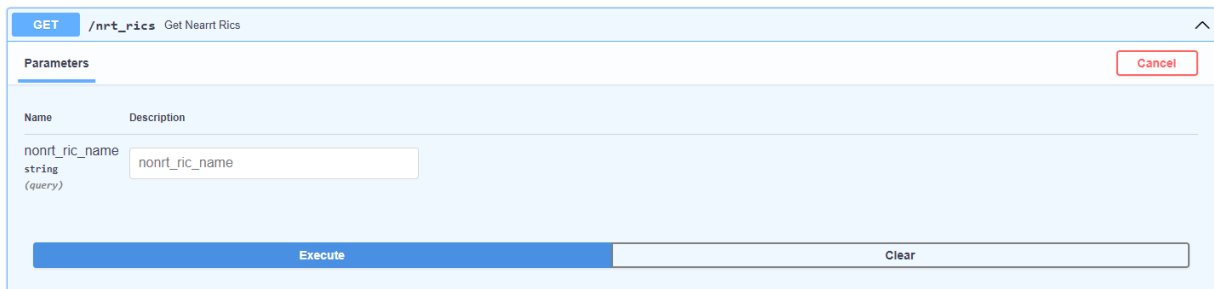
Figure 56 Screenshot with the execution of test "Creation of dRAX's Policy Type"

#### 4.2.7.6 List all the information stored about dRAX

We can recover all the information stored about all the registered Near-RT RICs (or specify a name to only recover one). That information includes the registration parameters (IP and port) as well as all the policy instances and xApps associated with the devices.

Test case name	List Stored Information Test	Test Case id	Int-test-07-06
Test purpose	List all the stored information about a Near-RT RIC.		
Configuration	RIC-Manager and dRAX up and running.		
Test tool	Postman/Browser		
KPI	-		
Components Involvement	RIC Manager, dRAX.		
Pre-test conditions	A Telemetry xApp has been deployed on a dRAX. There is at least one Policy Type created for said xApp and at least on instance of the policy has been created.		

Test sequence	Step 1	Perform a GET request to the RIC-Manager endpoint “/nrt_rics”.
	Step 2	Optionally select one instance of a Near-RT RIC to obtain only the information about said RIC.
	Step 3	Validate that the response contains all the information about the Near-RT RIC configuration as well as the deployed xApps and the Policy Instances associated to the Near-RT RIC.
Test Verdict	Response status code is 200.	
Additional Resources	-	



GET /nrt\_rics Get NearRT Rics

Parameters

Cancel

Name	Description
nonrt_ric_name string (query)	nonrt_ric_name

Execute Clear

Figure 57 Screenshot with the execution of test "List all the information stored about dRAX"

#### 4.2.7.7 Deletion of dRAX's Policy Type and all associated Policy Instances

To delete a Policy Type, and subsequently all the policy instances associated with it, we only need to specify the dRAX's friendly name and the Policy Type ID present in said dRAX.

Test case name	Delete Policy Test	Test Case id	Int-test-07-07
Test purpose	Delete a Policy Type associated to a Near-RT RIC and subsequently delete all the instances of said policy.		
Configuration	RIC-Manager and dRAX up and running.		
Test tool	Postman/Browser		
KPI	-		
Components Involvement	RIC Manager and dRAX.		
Pre-test conditions	There is at least one Policy Type associated to the Near-RT RIC and at least on instance of the policy has been created.		
Test sequence	Step 1	Perform a DELETE request to the RIC-Manager endpoint “/policy_type”.	
	Step 2	In the path of the request, specify which Near-RT RIC is associated to the policy, as well as the Policy Type ID.	
	Step 3	Validate that the response contains a confirmation message.	
Test Verdict	Response status code is 200.		
Additional Resources	-		



Figure 58 Screenshot with the execution of the "Delete Policy test"

#### 4.2.7.8 Undeployment of the Telemetry xApp

Finally, we can undeploy an instance of the Telemetry xApp by passing to RIC-Manager the dRAX's friendly name and the name of the exact instance, which is defined in the "xapp\_name" field of the deployment request.

Test case name	Undeploy xApp test		Test Case id	Int-test-07-08
Test purpose	Undeploy a xApp previously deployed in a Near-RT RIC.			
Configuration	RIC-Manager and dRAX up and running.			
Test tool	Postman/Browser			
KPI	-			
Components Involvement	RIC Manager and dRAX.			
Pre-test conditions	There is an instance of one of the xApps available in RIC-Manager's catalogue deployed in a Near-RT RIC.			
Test sequence	Step 1	Perform a DELETE request to the RIC-Manager endpoint “/deploy”.		
	Step 2	In the path of the request, specify in which Near-RT RIC the xApp is deployed, as well as the name given to the instance of the xApp.		
	Step 3	Validate that the response contains a confirmation message.		
Test Veredict	Response status code is 200.			
Additional Resources	-			

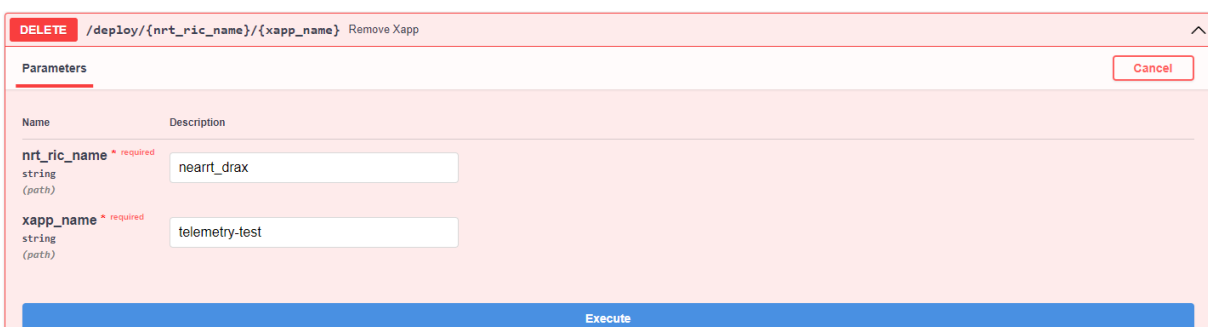


Figure 59 Screenshot with the execution of test "Undeployment of the Telemetry xApp"



#### 4.2.8 Orchestrator– 5G Core

As discussed earlier the integration between the orchestrator and the 5G Core doesn't include its lifecycle management or configuration. But the project has several requirements to use the data exposed by the 5G core. In such scenario, the integration of the orchestrator was focused on the federation techniques used to expose the Prometheus server included in the 5G Core to the other components of the architecture.

Therefore, this subsection aims to showcase the integration tests performed in order to verify the proper integration between the Orchestrator (Prometheus aggregator), and the federated Prometheus instance of the 5G Core in the Affordable5G ecosystem.

Test case name	Orchestrator federated monitoring	Test Case id	Int-test-08-01
Test purpose	Verify the aggregated Prometheus instance in the orchestrator receives a predefined list of KPIs from the 5G Core Prometheus server.		
Configuration	5G Core deployed Orchestrator Deployed Aggregated Prometheus configured to target the federated Prometheus		
Test tool	curl, Prometheus dashboard		
KPI	-		
Components Involvement	NearbyOne Orchestrator, Athonet 5G Core		
Pre-test conditions	Orchestrator instance running Orchestrator's Aggregated Prometheus configured with configuration file 5G Core instance running 5G Core Prometheus instance running		
Test sequence	Step 1	Check data collected in 5G Core Prometheus instance	
	Step 2	Check same data is collected in NearbyOne Aggregated Prometheus instance	
Test Verdict	The data in NearbyOne Aggregated Prometheus instance is the same as in the 5G Core Prometheus instance		
Additional Resources	-		

#### 4.2.9 Synchronization of TSN endpoint with ADVA master clock

The test case below shows the process to synchronize a TSN endpoint with a master clock. This master clock is provided, in our case, by ADVA FSP 150 equipment. Synchronization is critical in TSN, as time awareness in the full network is needed. Thus, this is the first step to achieve that final goal.

Test case name	TSN synchronization	Test Case id	Int-test-09-01
Test purpose	Synchronize TSN endpoint using ADVA FSP 150.		
Configuration	Relyum NIC integrated with TSN endpoint. TSN endpoint connected to ADVA FSP 150.		
Test tool	ptp4l, phc2sys, Relyum web manager		
KPI	< 100 ns Clocks' offset		

Components Involvement	ADVA FSP 150 TSN endpoint with Relyum NIC		
Pre-test conditions	ADVA FSP 150 configured as master clock for ethernet multicast.		
Test sequence	Step 1	Modify Relyum ptp4l config file with master clock parameters (domain number, transport, clock class, priorities and client-only mode).	Relyum card is ready to be synchronized with ADVA master clock.
	Step 2	Activate synchronization in Relyum web manager (ptp4l and phc2sys)	Relyum card is synchronized with ADVA master clock.
	Step 3	Run ptp4l and phc2sys commands on TSN endpoint hosting Relyum NIC.	TSN endpoint is synchronized with ADVA master clock
Test Verdict	If clock offset is < 100 ns, between both Master/PHC and PHC/System clock		
Additional Resources	None		

Results are exposed in Figure 60 and Figure 61. In Figure 60, we can see the output of executing the ptp4l command, which synchronizes the PHC with ADVA master clock. As it can be observed, the offset is consistently below 100 ns, which indicates that the synchronization has been achieved.

```

ptp4l[1561367.156]: master offset      -2 s2 freq +12378 path delay -190
ptp4l[1561368.156]: master offset     -13 s2 freq +12366 path delay -190
ptp4l[1561369.156]: master offset     -17 s2 freq +12359 path delay -190
ptp4l[1561370.156]: master offset      12 s2 freq +12382 path delay -190
ptp4l[1561371.156]: master offset      -6 s2 freq +12368 path delay -190
ptp4l[1561372.156]: master offset       0 s2 freq +12372 path delay -190
ptp4l[1561373.157]: master offset       4 s2 freq +12376 path delay -190
ptp4l[1561374.156]: master offset       3 s2 freq +12376 path delay -190
ptp4l[1561375.157]: master offset     -12 s2 freq +12362 path delay -189
ptp4l[1561376.157]: master offset       3 s2 freq +12374 path delay -189
ptp4l[1561377.157]: master offset       8 s2 freq +12380 path delay -189
ptp4l[1561378.157]: master offset       6 s2 freq +12380 path delay -189
ptp4l[1561379.157]: master offset      -3 s2 freq +12373 path delay -188
ptp4l[1561380.157]: master offset       0 s2 freq +12375 path delay -189
ptp4l[1561381.157]: master offset       1 s2 freq +12376 path delay -188
ptp4l[1561382.157]: master offset     -14 s2 freq +12361 path delay -187
ptp4l[1561383.157]: master offset       1 s2 freq +12372 path delay -189
ptp4l[1561384.157]: master offset      11 s2 freq +12382 path delay -189
ptp4l[1561385.157]: master offset       1 s2 freq +12376 path delay -188
ptp4l[1561386.157]: master offset       6 s2 freq +12381 path delay -188
ptp4l[1561387.157]: master offset       0 s2 freq +12377 path delay -188
ptp4l[1561388.157]: master offset       6 s2 freq +12383 path delay -188
ptp4l[1561389.157]: master offset       1 s2 freq +12380 path delay -188
ptp4l[1561390.157]: master offset       3 s2 freq +12382 path delay -188
ptp4l[1561391.157]: master offset       5 s2 freq +12385 path delay -188
ptp4l[1561392.157]: master offset       5 s2 freq +12386 path delay -188
ptp4l[1561393.157]: master offset     -12 s2 freq +12371 path delay -188
ptp4l[1561394.157]: master offset       0 s2 freq +12379 path delay -188
ptp4l[1561395.157]: master offset       7 s2 freq +12386 path delay -188
ptp4l[1561396.157]: master offset      -2 s2 freq +12379 path delay -187
ptp4l[1561397.157]: master offset      -6 s2 freq +12375 path delay -187
ptp4l[1561398.157]: master offset       1 s2 freq +12380 path delay -188
ptp4l[1561399.157]: master offset     -10 s2 freq +12369 path delay -188
ptp4l[1561400.157]: master offset       9 s2 freq +12385 path delay -188
ptp4l[1561401.157]: master offset       5 s2 freq +12384 path delay -188
ptp4l[1561402.157]: master offset       2 s2 freq +12382 path delay -188
ptp4l[1561403.157]: master offset       8 s2 freq +12389 path delay -189

```

Figure 60 Output of ptp4l command

In addition, in Figure 61 we can see the output of executing phc2sys command, which synchronizes TSN endpoint's system clock with PHC's clock of Relyum card. Again, the offset is always below 100 ns, so we can confirm that the TSN endpoint is fully synchronized with ADVA FSP 150.

```

phc2sys[2513988.750]: CLOCK_REALTIME phc offset 40 s2 freq -14248 delay 5146
phc2sys[2513989.750]: CLOCK_REALTIME phc offset -14 s2 freq -14290 delay 5116
phc2sys[2513990.750]: CLOCK_REALTIME phc offset 17 s2 freq -14264 delay 5150
phc2sys[2513991.750]: CLOCK_REALTIME phc offset -39 s2 freq -14314 delay 5101
phc2sys[2513992.751]: CLOCK_REALTIME phc offset 18 s2 freq -14269 delay 5145
phc2sys[2513993.751]: CLOCK_REALTIME phc offset -13 s2 freq -14295 delay 5099
phc2sys[2513994.751]: CLOCK_REALTIME phc offset 11 s2 freq -14275 delay 5147
phc2sys[2513995.751]: CLOCK_REALTIME phc offset 3 s2 freq -14279 delay 5147
phc2sys[2513996.751]: CLOCK_REALTIME phc offset 31 s2 freq -14250 delay 5103
phc2sys[2513997.751]: CLOCK_REALTIME phc offset -41 s2 freq -14313 delay 5126
phc2sys[2513998.751]: CLOCK_REALTIME phc offset 47 s2 freq -14237 delay 5098
phc2sys[2513999.751]: CLOCK_REALTIME phc offset -61 s2 freq -14331 delay 5102
phc2sys[2514000.751]: CLOCK_REALTIME phc offset 1 s2 freq -14288 delay 5099
phc2sys[2514001.752]: CLOCK_REALTIME phc offset -14 s2 freq -14302 delay 5110
phc2sys[2514002.752]: CLOCK_REALTIME phc offset 36 s2 freq -14257 delay 5114
phc2sys[2514003.752]: CLOCK_REALTIME phc offset 7 s2 freq -14275 delay 5102
phc2sys[2514004.752]: CLOCK_REALTIME phc offset -23 s2 freq -14303 delay 5101
phc2sys[2514005.752]: CLOCK_REALTIME phc offset 61 s2 freq -14226 delay 5099
phc2sys[2514006.752]: CLOCK_REALTIME phc offset -21 s2 freq -14289 delay 5138
phc2sys[2514007.752]: CLOCK_REALTIME phc offset 5 s2 freq -14270 delay 5110
phc2sys[2514008.752]: CLOCK_REALTIME phc offset -46 s2 freq -14319 delay 5133
phc2sys[2514009.752]: CLOCK_REALTIME phc offset 33 s2 freq -14254 delay 5107
phc2sys[2514010.753]: CLOCK_REALTIME phc offset -16 s2 freq -14293 delay 5146
phc2sys[2514011.753]: CLOCK_REALTIME phc offset 8 s2 freq -14274 delay 5099
phc2sys[2514012.753]: CLOCK_REALTIME phc offset -48 s2 freq -14327 delay 5101
phc2sys[2514013.753]: CLOCK_REALTIME phc offset 34 s2 freq -14260 delay 5134
phc2sys[2514014.753]: CLOCK_REALTIME phc offset -35 s2 freq -14319 delay 5100
phc2sys[2514015.753]: CLOCK_REALTIME phc offset 56 s2 freq -14238 delay 5131
phc2sys[2514016.753]: CLOCK_REALTIME phc offset -44 s2 freq -14321 delay 5099
phc2sys[2514017.753]: CLOCK_REALTIME phc offset 15 s2 freq -14275 delay 5146
phc2sys[2514018.754]: CLOCK_REALTIME phc offset 26 s2 freq -14260 delay 5117
phc2sys[2514019.754]: CLOCK_REALTIME phc offset -28 s2 freq -14306 delay 5147
phc2sys[2514020.754]: CLOCK_REALTIME phc offset 23 s2 freq -14264 delay 5131
phc2sys[2514021.754]: CLOCK_REALTIME phc offset -31 s2 freq -14311 delay 5148
phc2sys[2514022.754]: CLOCK_REALTIME phc offset 30 s2 freq -14259 delay 5116
phc2sys[2514023.754]: CLOCK_REALTIME phc offset -19 s2 freq -14299 delay 5133
phc2sys[2514024.754]: CLOCK_REALTIME phc offset 37 s2 freq -14249 delay 5137
phc2sys[2514025.754]: CLOCK_REALTIME phc offset -25 s2 freq -14300 delay 5137
phc2sys[2514026.754]: CLOCK_REALTIME phc offset -47 s2 freq -14329 delay 5101
phc2sys[2514027.755]: CLOCK_REALTIME phc offset -9 s2 freq -14305 delay 5148
phc2sys[2514028.755]: CLOCK_REALTIME phc offset 57 s2 freq -14242 delay 5122
phc2sys[2514029.755]: CLOCK_REALTIME phc offset -43 s2 freq -14325 delay 5099
phc2sys[2514030.755]: CLOCK_REALTIME phc offset 33 s2 freq -14262 delay 5133
phc2sys[2514031.755]: CLOCK_REALTIME phc offset -22 s2 freq -14307 delay 5083
  
```

Figure 61 Output of phc2sys command

#### 4.2.10 Integration Test Cases KPIs Summary table

The following table summarizes the different KPIs defined per integration test cases. In this table there are not detailed the functional test cases.

Table 2 Integration Test Cases Summary Table

Component	KPI	Test Case ID	Expected Value
RU-DU	Validation of the 5G SA emitted spectrum	Int-test-01-01	Spectrum Validated with Spectrum Analyser (100MHz at 3.5GHz)
RU-DU	5G UE Connectivity in SA mode	Int-test-01-02	3GPP 5G SA message exchange for the UE connection
RU-DU	COTS UE ping rates	Int-test-01-03	ping with COTS UE measured in ms and data rate UDP + TCP in Mbps

RU-DU	ORAN Interface Validation	Int-test-01-04	Oran Interface Validated according to ORAN 7.2 definition in Uplink and downlink
TSN – Adva master clock	Synchronize TSN endpoint using ADVA FSP 150.	Int-test-08-01	< 100 ns Clocks' offset

## 5 TSN OVER 5G PROOF OF CONCEPT

### 5.1 Introduction

The TSN over 5G Proof of Concept, what is being developed to be integrated in Malaga platform, was already explained in D3.2 [3]. However, after several discussions and developments, the architecture has been improved with new functionalities and components. The final architecture can be found in Figure 62. To ease its comprehension, we will divide the explanation in the three main challenges coming from the development of a TSN over 5G solution and how to solve them, these are: Translation from TSN domain to 5G, prioritization over 5G and time synchronization. These parts will be explained in detail in the next subsection.

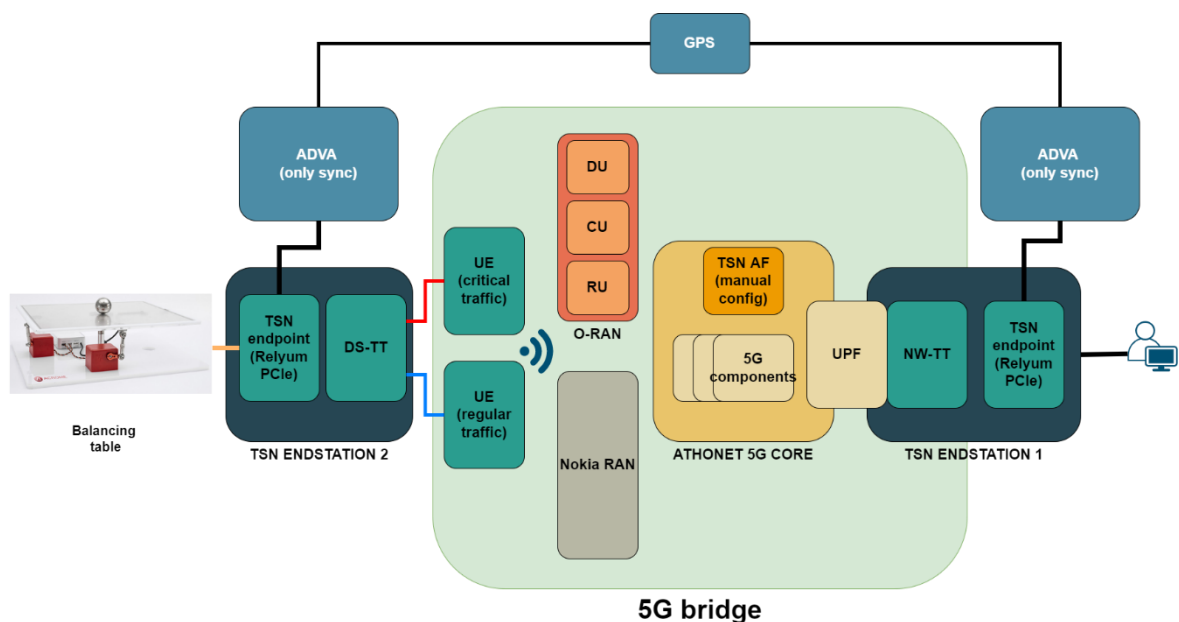


Figure 62 TSN over 5G PoC final architecture

### 5.2 Pilot Building Blocks

#### 5.2.1 Translation from TSN domain to 5G

One of the most critical challenges in TSN over 5G is how to adapt the traffic from TSN domain to 5G domain. To achieve that, the following components are involved: the wired TSN network, composed by TSN endpoints; the TSN translators, Device Side Translator (DS-TT) in the device side and Network Side Translator (NW-TT) in the network side; and 5G UEs and UPF, in the 5G network.

On the wired TSN network, critical traffic is generated and received together with regular traffic. This critical traffic will be simulated in the final demo with a balancing table, which is on the wired device side, controlled by another endpoint on the wired network side. In order to send the traffic from one side of the network to the other one through the 5G network, the first step is to translate these critical and best-effort traffic from TSN domain to 5G domain. For this purpose, UMA has developed two translators.

These translators are software-based switches using Stratum CLORAN [7] as OS, which is an open-source silicon-independent switch operating system for software defined networks.

Moreover, translators' functionality is implemented using match-action tables based on P4 language (SLI) [6], which is an open source, domain-specific programming language for network devices, specifying how data plane devices (switches, routers, NICs, filters, etc.) process packets. In addition, table entries can be added, removed or modified using P4Runtime interface, which allows to do these changes dynamically.

### 5.2.2 Prioritization over 5G

In wired TSN networks, scheduling and traffic shaping allows for the coexistence of different traffic classes with different priorities on the same network. These traffic classes are, in practice, identified with the PCP field in 802.1Q header. Thus, one of the challenges in TSN over 5G is how to guarantee these priorities in the 5G network with regard to assure bandwidth and end-to-end latency.

To achieve that in 5G, a mapping between TSN traffic classes and 5QIs is required. This mapping involves a 5G PDU session establishment with the 5QI demanded, which can be carried out either statically or dynamically. The dynamic request of the bearers in 5G can be managed by the AF. However, the development of a full TSN AF is out of the scope of this project.

In Affordable5G project, the bearers are established in a static way. Two bearers are configured: one for critical traffic, associated with a specific 5QI with higher priority, and the default bearer for best-effort traffic. Unfortunately, there aren't UEs in the market supporting simultaneous DNN for data traffic. For this reason, in the final architecture two different UEs will be used, one for critical traffic and another for the regular one. The TSN translators' entities (NW-TT and DS-TT) are in charge of mapping the TSN class into a 5QI, and vice versa, modifying the PCP field.

### 5.2.3 Synchronization

Synchronization is another key feature of the TSN over 5G network. The optimal approach would be to use a unique master clock for the full network. Thus, transmitting the synchronization packets (PTP) through 5G network. However, this is not available in 3GPP Release 15, which is the version supported in Affordable5G network. For this reason, in this PoC, synchronization is carried out with the help of two FSP 150-GE100Pro, provided by ADVA, that act as grand master clocks. It is important to note that we are using two different master clocks, one for the device side and another for the network side. However, these master clocks use the same GPS signal as a reference. Thus, an important assumption here is that, as they are both using the same reference signal, both sides are essentially synchronized.

Signal clock is propagated through both sides of the wired network using PTP flows, which are generated by the masters and distributed to the clients' PHC clocks. Moreover, at the end stations, not only the TSN cards are synchronized, but also the OS system clocks, to be able to generate and receive synchronized traffic at application level. Finally, for this proof of concept, this synchronization signal will not be distributed through the 5G network for the aforementioned reasons. Instead, own 5G mechanisms and configurations will be modified to be optimized for TSN over 5G, both in RAN and 5G core.



### 5.3 Integration and Deployment Status

The current status of the solution is depicted in Figure 63. The main differences between the final architecture and this one is detailed as follows.

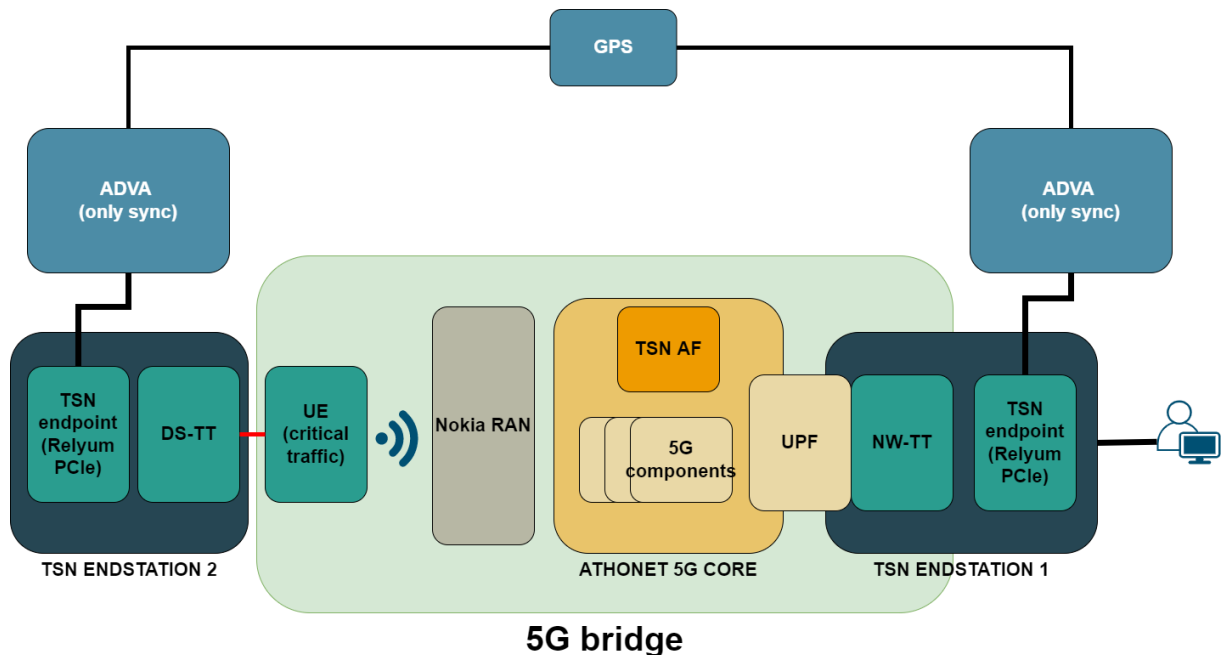


Figure 63 TSN over 5G PoC current architecture

Focusing on architecture level, at the moment only Nokia RAN solution is deployed, which is part of the Malaga platform in which this proof of concept is carried out. It is expected to have, at the end of the project, two different O-RAN architectures available, together with the Nokia RAN. One, coming directly from Affordable5G project and an additional one, acquired by UMA, in order to show multivendor interoperability. The second difference in the architecture is that only one 5G UE (Telit fn980m) is used in the current status. Thus, it is needed to add an additional UE in order to allow the management of QoS. Basically, each UE will be associated to a different bearer, which has been configured for specific 5QIs.

These are the only differences in the architecture. However, there will also be modifications in the functionality of the components. Specifically, the inclusion of an additional UE will include a new port in the DS-TT translator that needs to be managed. Therefore, it is needed to improve the features of the translator. The updated entity, depending on the access port, will modify the PCP field, in order to assign the corresponding priority. That is, if data is coming from the 5G UE for critical traffic, the PCP field will be modified with a higher value than if it is coming from the 5G UE for regular traffic. In addition, another important job to be done is the research on the optimal configuration of the 5G network for TSN traffic. Several configurations will be tested, and the optimal parameters will be set up in the final solution.

Finally, it is also important to mention that the synchronization part will not be updated, as it is assumed that the current solution is good enough for this proof of concept. Final results will show the suitability of this approach.



## 6 SMARTCITY

### 6.1 Introduction

The SmartCity pilot intends to demonstrate the usage of a 5G private network, and the advances provided by Affordable5G in an emergency scenario occurring at a shopping mall. The demo scenario will host in-door, assuming the loss of a child in a shopping mall.

The pilot will provide two types of services: a dynamic and a static service. The dynamic service will be hosted indoors, and its main goal is to find a missing person given the provided image captured by mobile device. The static service consists of a security CCTV system for person detection. The following figure illustrates the links between the person detection system and the Affordable5G components and provides the essential information and data flows required for addressing the pilot's requirements and specifications.

The distinction between these two services besides the ML algorithm, equipment and their core functionalities are in the fact that it is expected that the 5G Core network will change the bandwidth priority for the Dynamic service when needed.

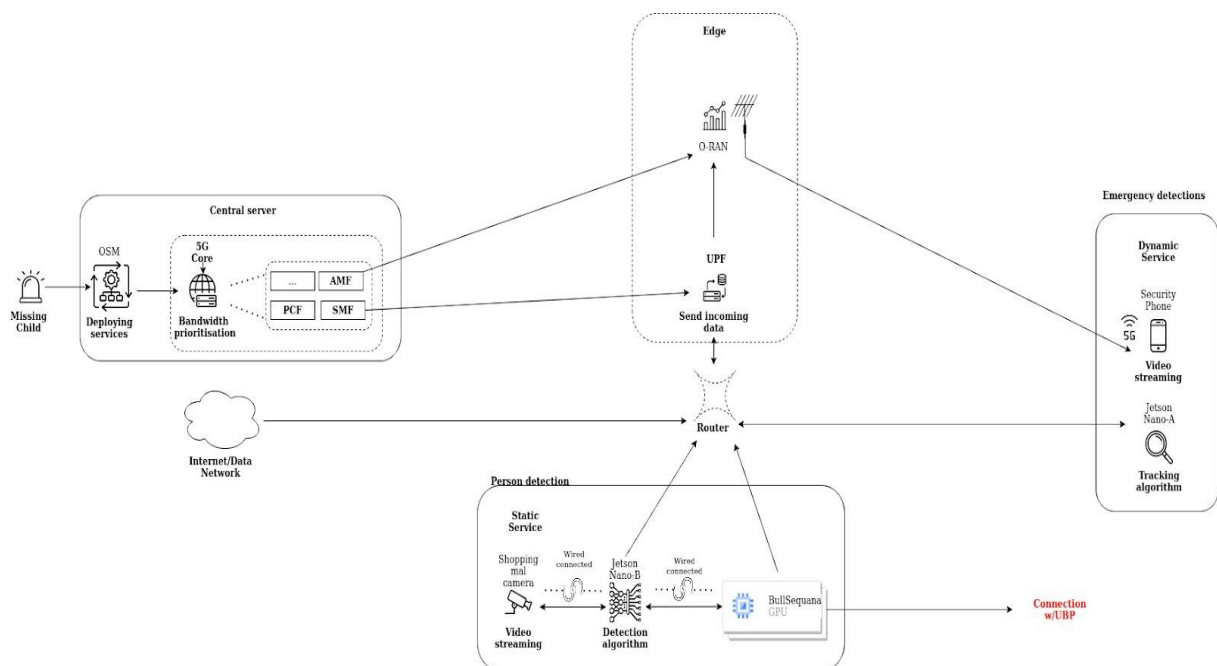


Figure 64 Smartcity Pilot Architecture

This architecture (Figure 64) assumes the existence of a telecommunication network. Such a network should be formed by a 5G-NPN deployment, consisting of the Core and the Radio Access parts and moreover extended by a wired connection. It should be noted that such a network must provide the necessary integration mechanisms for the use case to succeed, in other words a wired, but most importantly, a 5G-NPN enabled communication channel is of the utmost importance. According to discussions with the different partners involved in the infrastructure (for this specific use case) a router is used to enable point to point communication between the network actors. With this in mind, the use case that we are proposing should be connected to this router and, by that, have access to the 5G capabilities of the infrastructure (5G Core, UPF, OSM, etc.).

## 6.2 Pilot Building Blocks

The pilot is focused on two types of services: a dynamic and a static service. The dynamic service will be hosted indoors and consists in finding a missing person given the provided picture. The static service consists of a security CCTV system for person detection. Figure 64 illustrates the links between the person detection system and the Affordable5G components and provides the essential information and data flows required for addressing the pilot's requirements and specifications.

### 6.2.1 Dynamic service

The demo scenario starts with a parent reporting his missing child in a shopping mall. The Affordable5G's OSM (orchestrator) module will then deploy the detection system, located at the edge layer of the 5G private network, to analyze the video streams from all the cameras in the mall. This analysis will require the usage of a UPF, which may be privately owned or belong to a national operator. It is responsible for the routing of the actual data coming from the RAN to the Internet/DN. The UPF quickly and accurately routes the packets to the correct destination over the Internet/DN, addressing the need for a security guard to perform video streaming using a mobile phone.

The 5G Core and the O-RAN modules will be used to supply the highest possible bandwidth, through their prioritization feature, to the video equipment in order to allow video streaming over the 5G network.

Having received the information, consisting of at least one image of the missing person, a Machine Learning (ML) algorithm focused on person re-identification is used to search for the target person in the mall camera's video streams. In re-identification approaches (REID) [17], a system tries to identify a target person in a gallery/query set. The following picture illustrates the flow of designing a practical person Re-ID system, which consists of the following five steps:

1. Raw data collection
2. Bounding Box Generation
3. Training Data Annotation
4. Model Training
5. Pedestrian Retrieval

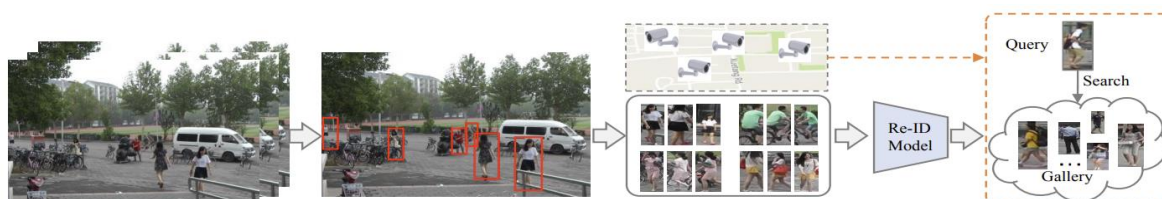


Figure 65 Smartcity steps for re-identification

From these five steps, the re-identification methods are generally divided into two classes: Open-World Person Re-Identification and Closed-World Person Re-Identification. The difference between both classes is summarized in the following table (REID) [17]:



The next image represents an alternative diagram which illustrates how the association between the different components of the dynamic service at a protocol level is envisioned to be.

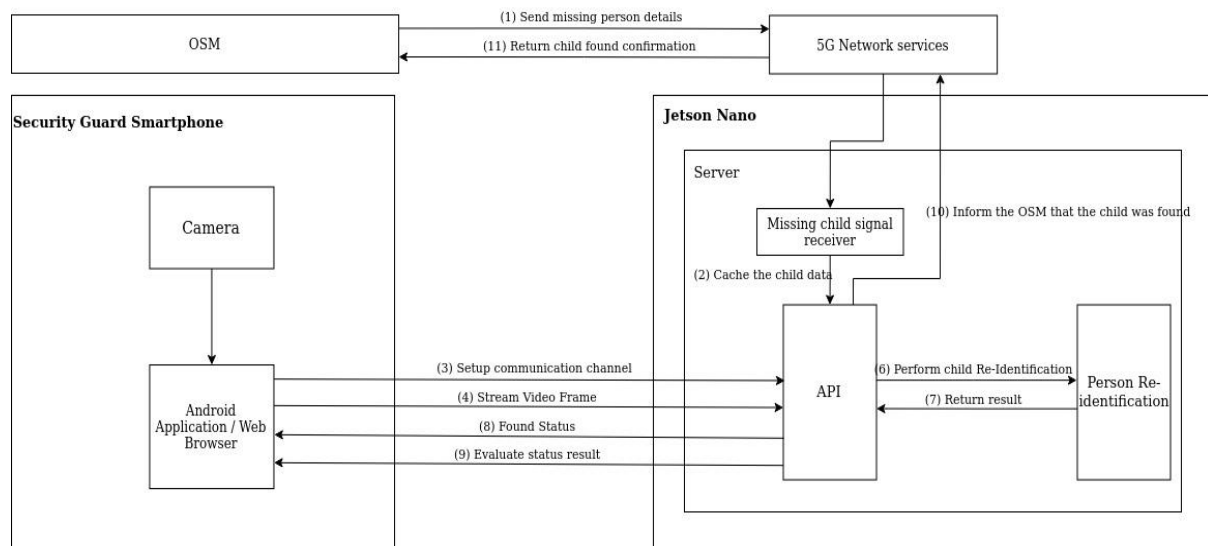


Figure 67 Smartcity association between different components

The different components have the following functionalities:

- Missing child signal receiver receives and makes the obtained information from the missing child alarm available to the server, in order for it to be used by the API to inform the guard and by the re-ID algorithm. The information received from the external services contains the information of the lost child.
- API: processes the communication/data between the Application/Web browser and the server instantiated in the Jetson Nano. While transmits/receives data with Person Re-identification when an alarm of a lost child is enabled.
- Person Re-identification: tries to find a match between the provided missing child's picture/picture set and the gallery set. This task starts once a missing child alarm is received and will continuously make a positive result available to the server, in order to be sent to the security guard for confirmation, until the security guard confirms the result.
- Camera: The smartphone's camera captures the scene pointed by the security guard.
- Android Application/Web Browser: Platform used by the security guard to receive the notification that an alarm of a lost child is ongoing and that he must start a new search. Additionally, this platform allows the user to confirm/dismiss a detection made by the re-ID algorithm. Furthermore, depending on the manner that the access to the security guard smartphone camera is done, it may transmit the frames to the Jetson Nano.
- OSM: triggers the lost children alert across the network until it reaches the Missing child signal receiver.
- 5G Network services: When necessary, provides higher prioritization, transmits/receives the status of the missing child alert between the OSM and the server.

Considering the scenario, a successful and rapid resolution of the emergency depends on how the 5G network supports the intervention of the security team member in the search for the missing kid. This is only possible due to two specific network features: the capability of locally

maintaining the user data traffic, in order to minimize the latencies and allow smooth data exchange between the mobile phone and the Jetson Nano; and the capability of prioritizing the traffic of the security guard's smartphone compared to other User Equipment (UES), so as to avoid service quality degradation in case of network congestion due to a crowded area.

Additional traffic prioritization and QoS control are performed via specific configurations of the UPF at the PDU session management level. These configurations are implemented within the UPF as packet filters and are identified by a series of specific parameters that differentiate the QoS flows within PDU sessions. For each device, these parameters are provided to the UPF by the Session Management Function (SMF), which, in turn, receives updates on traffic policies from the Policy Control Function (PCF).

### 6.2.2 Static service

The static service is available through two types of computer systems: a server capable of performing person detection in multiple video streams provided by connected edge nodes, and a Jetson Nano (Jetson Nano-B) which performs the role of an edge node, that also consists in person detection given the video stream obtained from a connected camera. It's referred to as a static service since it doesn't take advantage of the 5G capabilities due to some equipment not being capable of connecting to the 5G network.

The Jetson Nano performs person detection using a lightweight person detection algorithm based on You Only Look Once (YOLO), which makes predictions with a single network evaluation.

When a person is detected by the Jetson Nano, the video stream is transmitted to the BullSequana Server (BLLSQ) [18], a more computationally powerful device compared to the Jetson Nano, where a more capable and robust machine learning algorithm processes the stream. On this server, there is also hosted a web server that displays to a user of the UP the video stream with the ongoing detections. These detections are expected to be stored in a Wasabi [19] bucket, which is a cloud storage service that provides a high-performance, reliable, and secure data storage infrastructure. These stored files will also be available to access in the UP.

Additionally, the connection between the Jetson Nano and the network provider will be supported by an ethernet connection since this equipment natively does not support 5G connections.

The ML algorithm running in the server is based on DeepSORT and YOLOv4, [20] which enables the detection and tracking of individuals within the camera's field of view. The overall workflow of this system is illustrated in the following image:

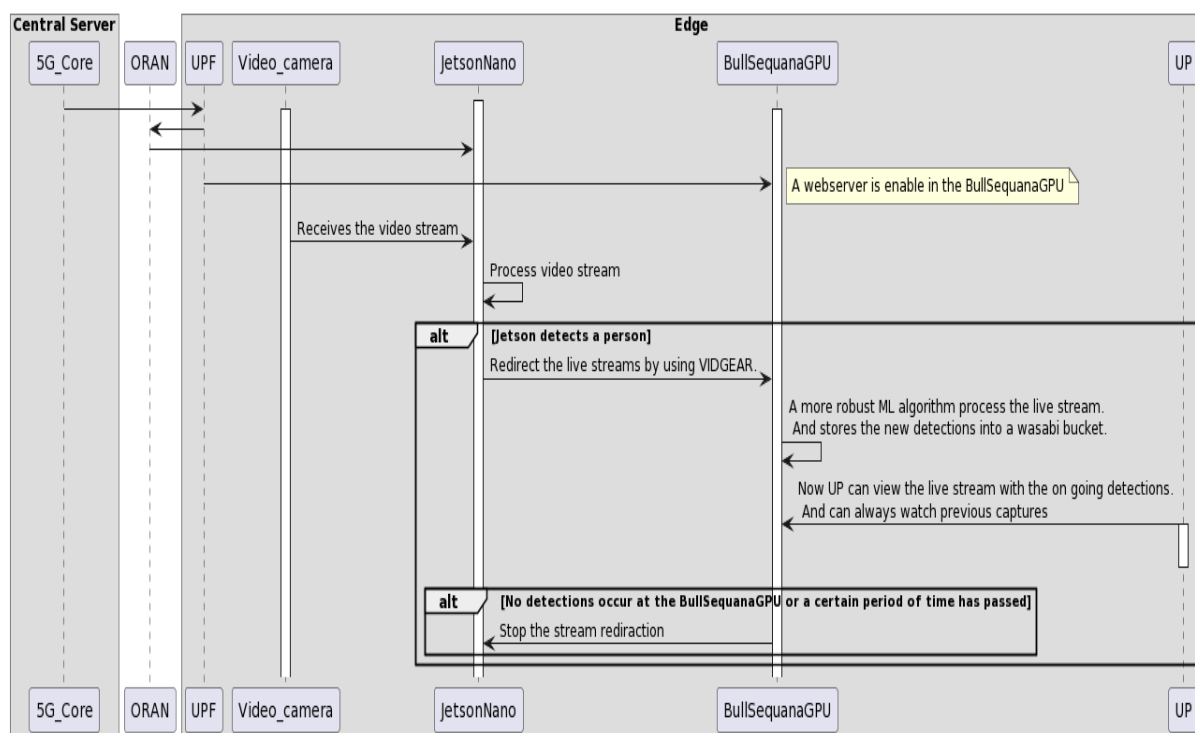


Figure 68 ML Algorithm workflow

To better depict how the different components involving this service work, the following organizational diagram was created. Moreover, the enumeration between the components intends to suggest an order that may not be followed.

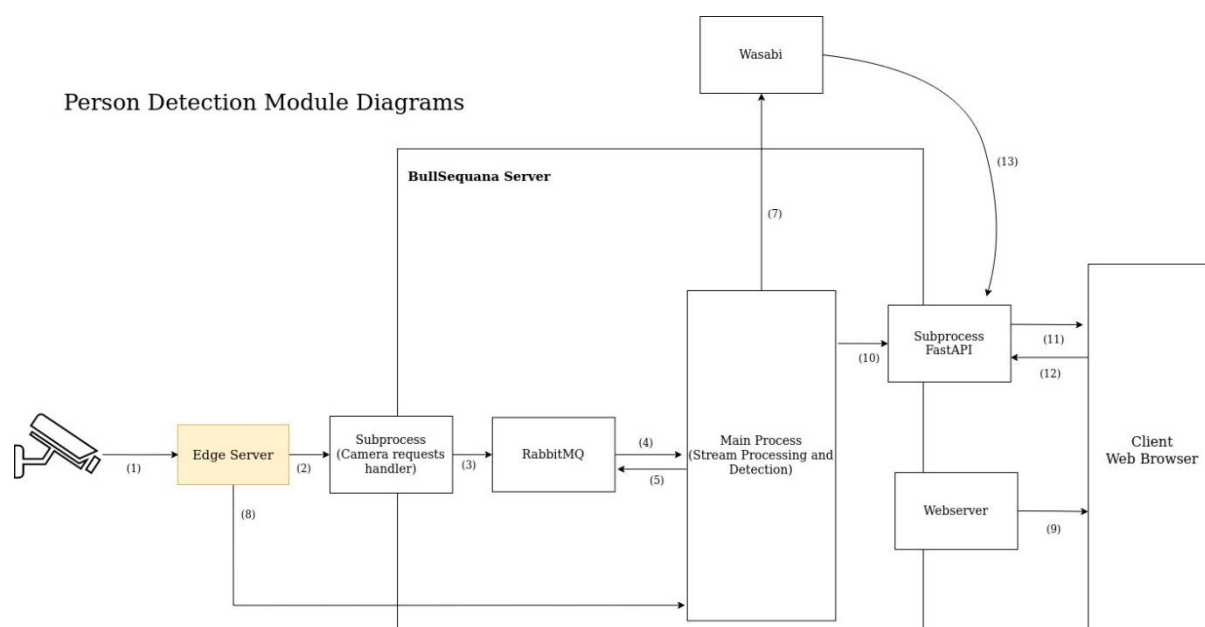


Figure 69 Smartcity components involving service work

This component has the following functionalities:

- Edge server: transmits the live stream and receives the responses if it should transmit the information to the BullSequana Serve. This latter capability was omitted to simplify the illustration.
- RabbitMQ: receives and transmits the reply to the Edge Server, if it has permission to perform the transmission.
- Main Process: performs the detection and saves the captures to Wasabi.
- SubprocessFastAPI: Provides an API for requesting camera streams and the saved videos.
- Webserver: Provides a web interface for viewing the streams made available by the FastAPI

## 6.3 Integration and Deployment Status

### 6.3.1 Dynamic service

The current iteration of the Dynamic Service is still under analysis to ensure the optimal communication between the Jetson Nano and the security guard's smartphone. As well as to define the minimal requirements regarding the dataset of the lost child to have a fully functional ML algorithm, also, these requirements should address the mitigation of false positives and false negatives.

As previously mentioned, the best approach for the security guard to receive the notification of a lost child is by a native application or a web browser approach is being investigated.

### 6.3.2 Static service

In relation to the progress of the Static Service implementation, the software project for the Jetson Nano is dockerized, has access to the GPU for faster processing and takes advantage of a Machine Learning algorithm to detect people. It can also be connected to a camera via Real-Time Streaming Protocol (RTSP) or through a physical connection.

As for the integration with the different partners, a task force was created to facilitate communication and promote synergy to have a faster development cycle and deployment. The significant milestones accomplished for this task are summarized in the following table.

Functionality	Status
Live feed transmission to the Cloud.	Done
Request and receive port to stream to the Cloud.	Done
Stop transmission to the Cloud and resume its detection	Done
Integration of the software solution with Docker	Done
Verification of the type of camera it is connected	Done

Regarding the deployment at the BullSequanaGPU, the dockerized environment is under development. A FastApi [25] web server was implemented to read streams posted by the Jetson Nanos to the RabbitMQ [26] instance. This will allow the webserver to retrieve frames from multiple cameras and stream them to the user in a grid layout or from a single camera specified by the user.

The following table summarizes the implemented features and their current development status.



Functionality	Status
Receive a request from the Edge and assign a new port if any are available.	Done
Integration of a Machine Learning algorithm for detecting people	Done
Visualization of the detection performed by the cloud machine learning algorithm	Partially Done
Processing streams in parallel taken from different edge nodes.	Done
Storing the frames in a media file and into a wasabi bucket.	Partially Done
Implement re-identification of individual(s).	To be Done

An essential integration with Urban Platform [27] was done, which consisted of the use of authorization tokens to verify that the user has the required permissions to access the live stream. It needs further development once all current issues are addressed.

Since the transmitted information is sensitive, VidGear [28] was configured to use the secure mode in order only to allow servers/clients to communicate if they have a valid certificate, thus safeguarding the streaming from nefarious agents. A RabbitMQ broker was deployed with user authentication and Transport Layer Security (TLS) not to allow unauthorized instances to communicate or spoof the data.

## 6.4 Integration and Deployment Status in THI Platform

The current iteration of the Edge is dockerised, still only CPU-bound, and takes advantage of a machine-learning algorithm to detect people. It can also connect to a Real-Time Streaming Protocol (RTSP) connected camera or to one that is physically connected. The significant milestones for this part are represented in the following table.

A part of the CVAE services will be also demonstrated in the FPGA platform provided by THI. In particular, the people detection algorithm will deploy in the Zynq platform in which a multithreaded instance of the NEOX accelerator will be realized. Apart from the hardware IP, THI will also utilize the NEOX SDK for optimizing the CNN models in terms of memory footprint and execution time. The goal is to explore and showcase that the person detection algorithm can be executed with the required performance in a far edge platform (e.g., a low-cost camera) characterized by scarce resources (both in terms of memory and computational capabilities) operating at the same time under tight power constraints. The significant milestones for this part are represented in the following table.

Functionality	Status
Development of a lightweight DNN-based person detection application	Done
Integration of the person detection of application in the AI-SDK compression framework	Partially Done
Integration of the person detection of application in the AI-SDK deployment framework	Partially Done
Deployment of person detection in NEOX accelerator	Done

Camera connected to the FPGA platform	Done
Zynq FPGA platform connected to cloud or edge	Not Done

The person detection CNN model is deployed in THI FPGA platform and effectively parallelized in at least 32 threads, while the CNN is compressed by a factor of 5x compared to the initial size of the model. The execution of each inference step is executed in less than 1 sec at a power consumption less than 3.5mWatts.

## 7 MISSION CRITICAL SERVICES

### 7.1 Introduction

“Emergency Communications” Pilot objective is to integrate, deploy and demonstrate 3GPP-compliant MCS services on top of the Affordable5G private network responding to cloud native functions of monitoring, flexible deployment, scaling and resources allocation.

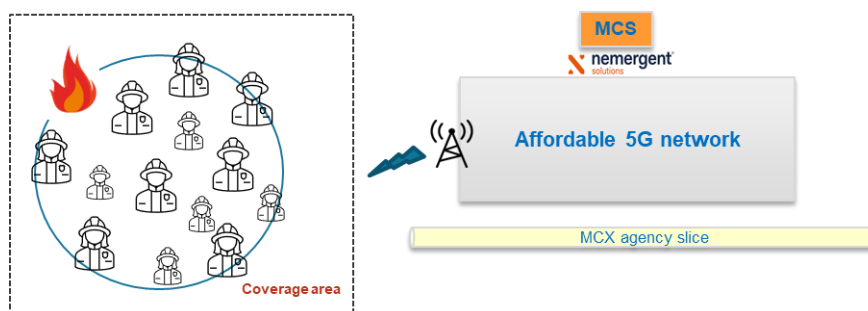


Figure 70 Simplified vision of Pilot1

As part of the Affordable5G network, there is the possibility of deployment over different point of presence (PoP), being able to deploy the service in generic main infrastructure or in closer and more optimized placement edge locations. This underlying infrastructure gives the possibility of not only applying modifications to the service once deployed in a single PoP but also the chance of moving the service by re-instantiating in another PoP if the monitored metrics meet the re-instantiation threshold (seen as the need for a better place to host the service).

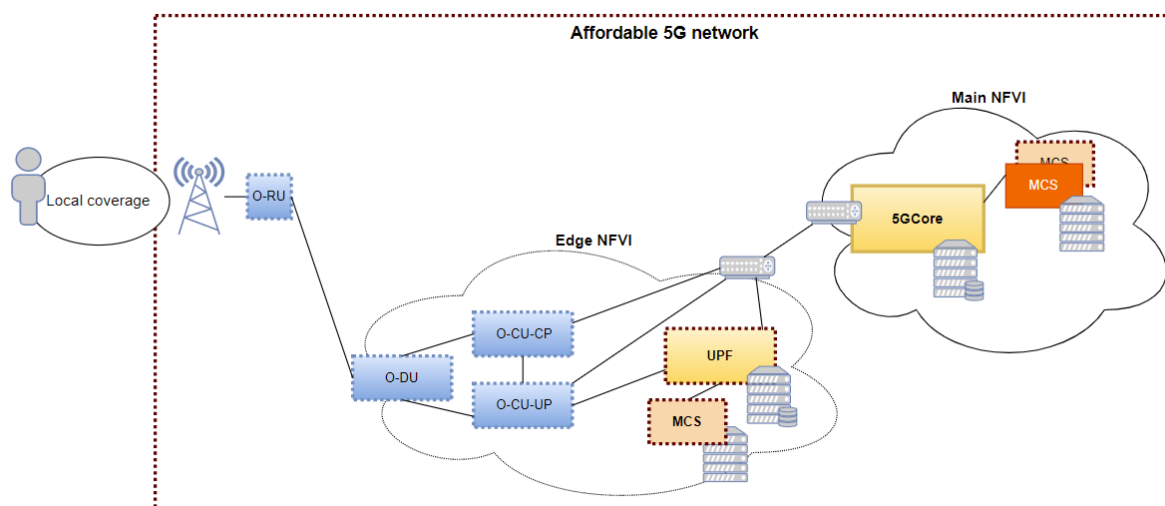


Figure 71 Pilot1 in different PoP

The two use cases that are going to be demonstrated in the validation phase are the ones corresponding to service scalability and service re-instantiation.

Whenever the system detects a service KPI degradation due to an increasing number of connections or a load increase, an MCS scaling mechanism is implemented to deploy a new

MCS CNF/KNF. For the case of high latency if an edge service deployment and a MC service instantiation may be possible, another MC instance in the edge will be implemented.

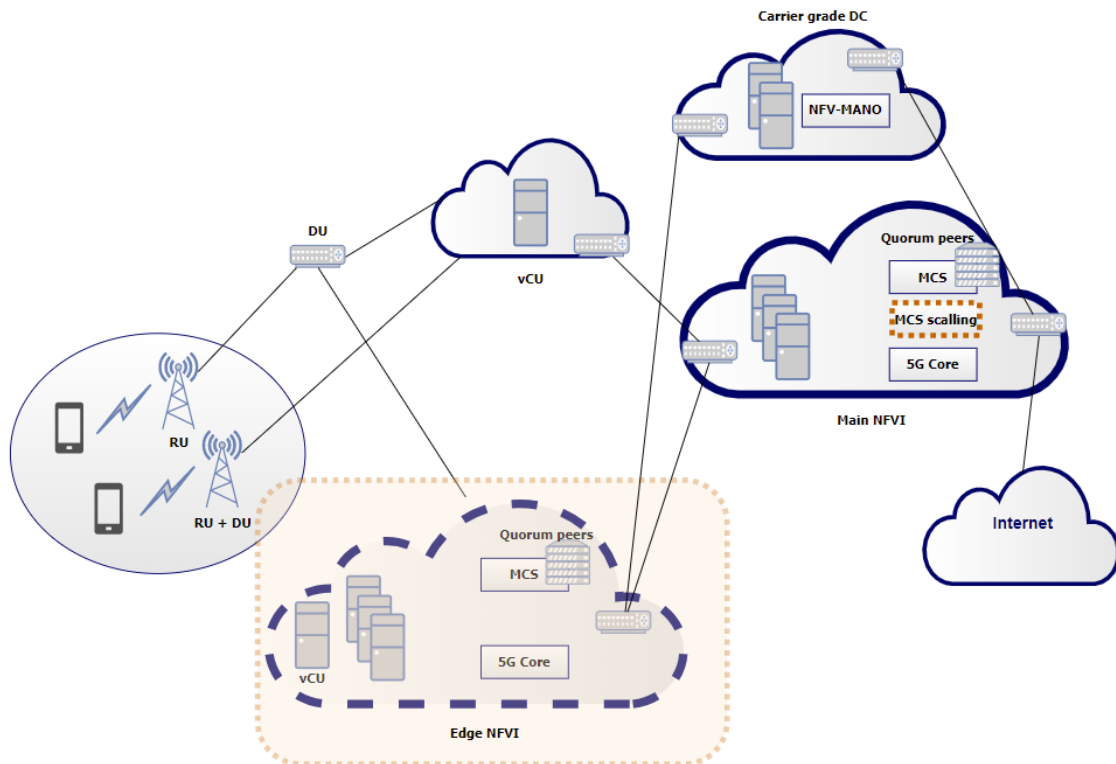


Figure 72 MCS service in Affordable 5G network

## 7.2 Pilot Building Blocks

The initial conception of the interactions between the main building blocks which intervene in the pilot has been described as follows:



Figure 73 Simplified interaction between modules in Pilot1

Depending on the events happening in a certain area, both system and MCS service KPI are collected by the System Monitoring. Alarm and detection systems are required to notice any variation of the nominal values from the system and MCS KPI. A communication between the orchestrator, the MCS Service and the infrastructures is considered. The orchestrator shall trigger the actions (in our case, service scalability and service re-instantiation) depending on the alarms received from the monitoring module. The orchestration layer shall provide management and operation of network slice creation across the whole infrastructure and the services instances as well.

In Figure 74 below we can see a UML diagram, from “D1.2: Affordable5G building blocks fitting in 5G system architecture” [1] depicting the interactions that were initially considered and that have been simplified during project lifetime. The analytics subscription routine, monitoring and detection routine and scale-up routine are shown between the main components described in the previous paragraph.

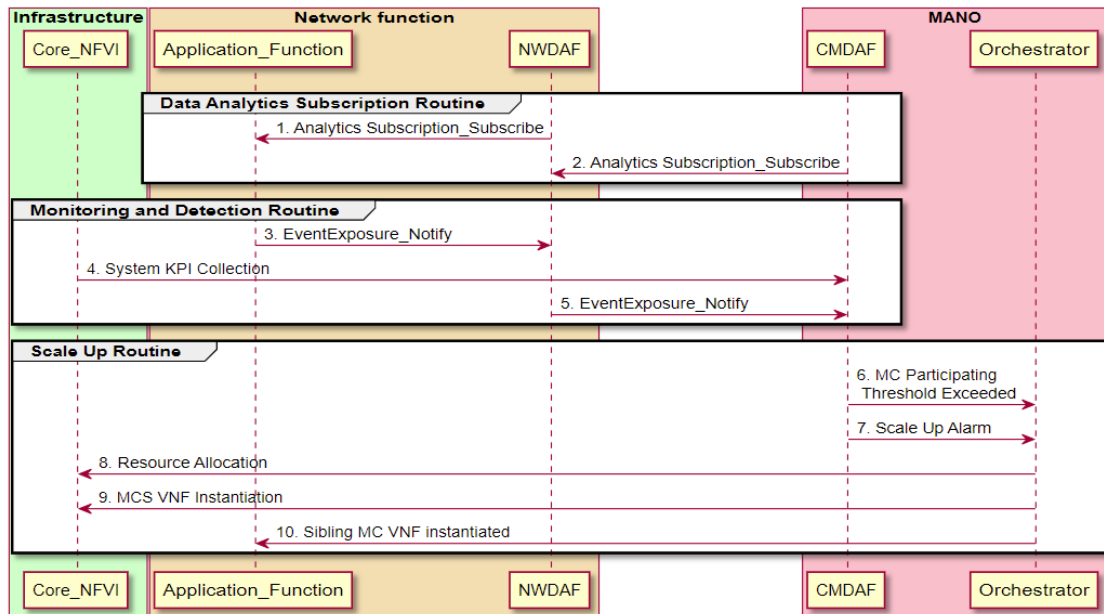


Figure 74 Scalability use case workflow

As the integrations have been materialized and the validations have started, it has been decided to integrate the MCS service, monitoring module and orchestrator as follows.

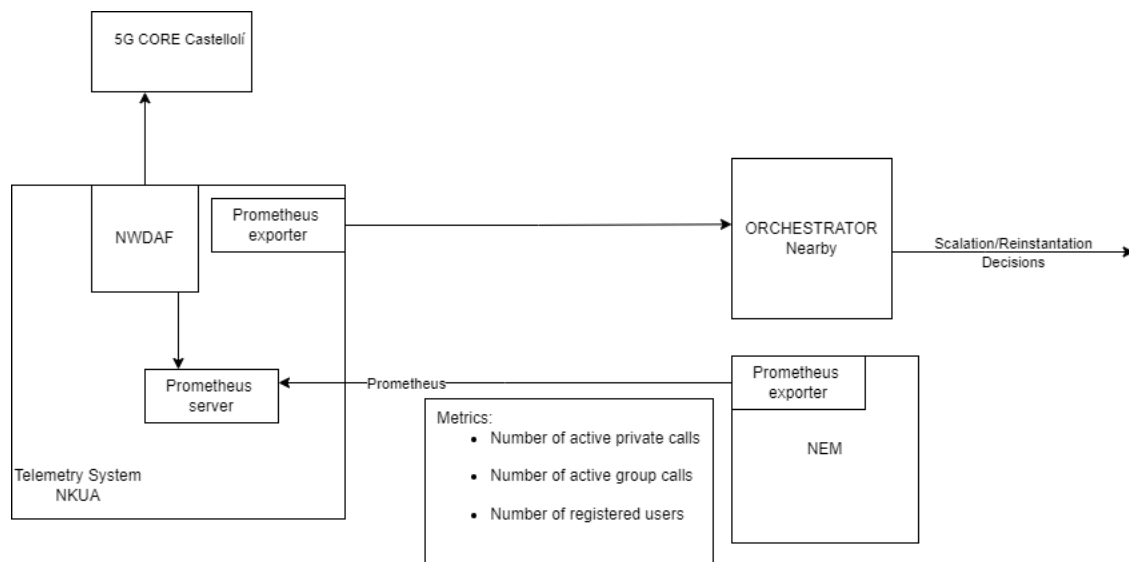


Figure 75 Interactions between MCS service, monitoring and orchestrator

Nemergent Services will expose some internal metrics using Prometheus exporter to NWDAF or Network Analytics module developed by NKUA. NWDAF will process and aggregate this data and expose this data through Prometheus. This data will be consumed by Orchestrator developed by Nearby Computing and according to that data the Orchestrator will trigger the required action to fulfill each use cases requirements in terms of service re-instantiation.

### 7.3 Integration and Deployment Status

The Emergency Pilot in Castellolí contains a containerized version of the 3GPP-compliant Mission Critical Services (MCS/MCX) that involves in the two ends of the service the docker components of the “server-side” and the Android App in the “client-side”. The presence of server and client side is important to understand the status and context of the deployment and the encountered constraints along the path.

Figure 76 illustrates in a simplistic way the different dockerized components that are deployed with the Nemergent HelmChart and the connection points or exposed ports with the client side of the service. In this regard, the Castellolí platform offers from the very beginning the NodePort networking exposure function that the Nemergent service uses in the HelmChart and considers for evolutions of the service like scaling-up methods (technical consideration of having same IP and different port for all “external” components in NodePort). Currently the platform also offers the possibility of deploying with LoadBalancing capabilities and Nemergent is adjusting the deployment and service intrinsic relationships to completely support it (technical consideration of having same port and different IP for all “external” components in LoadBalancer).

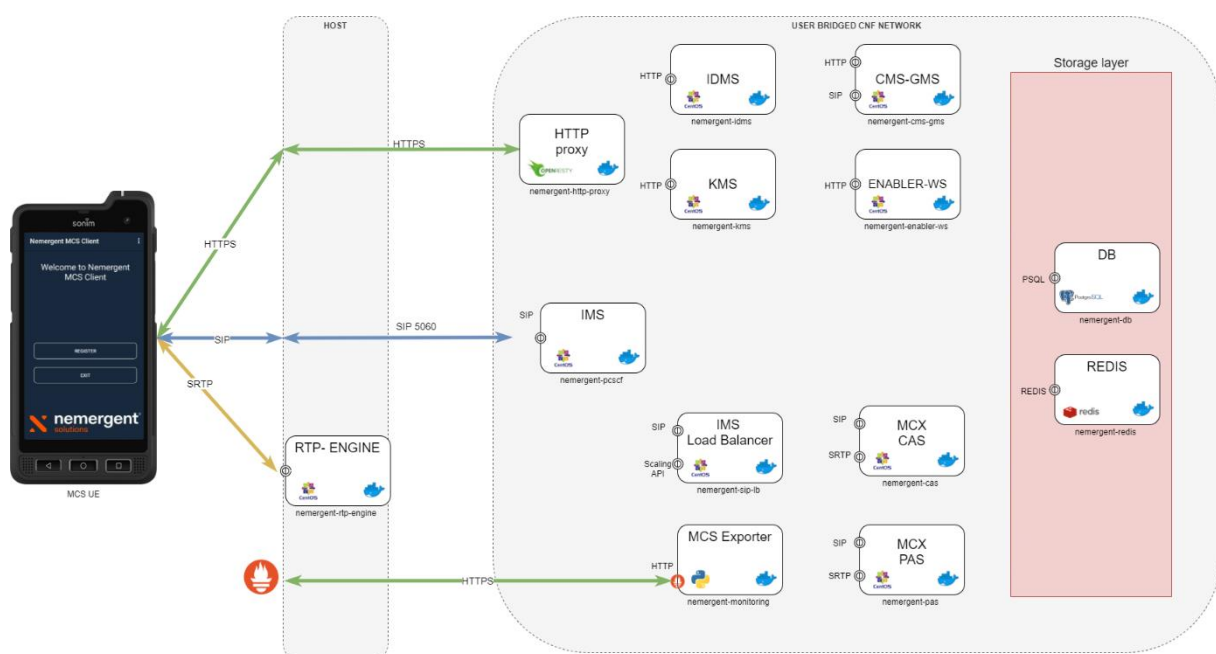


Figure 76 Dockerized MCS service

Taking into account the abovementioned figure and the used NodePort method to externalize certain components to be reachable by the client side, the next table sums-up the utilized ports while deploying in Castellolí. The reason to highlight such table is due to the fact that the service has required hardcoding adjustments in terms of used ports for Configuration Management Server (CMS) and Identity Management Server (IdMS) that usually make use of ports 80 and 8080. This time, the ports were modified to be 20001/20003 and 20002/20004 (in bold) to avoid present blocked ports in the Castellolí infrastructure and complex networking environment.

Table 3 Emergency Communications port exposure in NodePort mode

Protocol	DST PORT	MCS/MCX
TCP	20001/ 20003	CMS
TCP	20002/ 20004	IdMS
TCP/UDP	5060	P-CSCF (IMS)
TCP	20000	xMS (HTTP)
TCP/UDP	30000-40000	RTP traffic

All constraints and adjustments considered, the Emergency Pilot can be easily deployed using the HelmChart provided by Nemergent and modified to point to NearbyOne docker registry. Figure 77 shows the “nemergentmcs” service deployed in “slice-nemergent” Namespace in Castelloli.

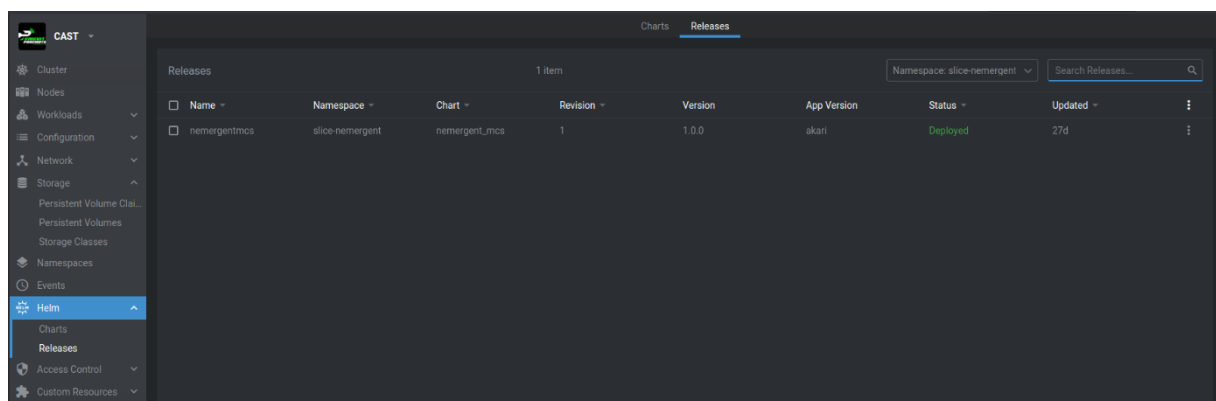
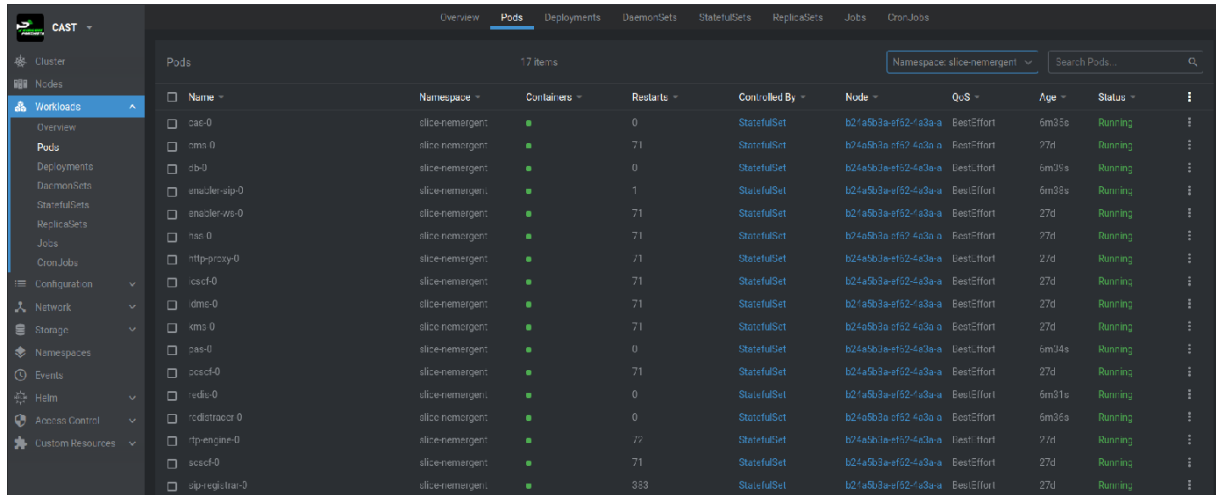


Figure 77 Nemergent MCS service deployed in Castelloli

The service brings all linked components in the HelmChart and are easily accessible as isolated Pods. Tools like the one used in the screenshots (Lens) helps in the troublesome process of troubleshooting specific Pods in terms of available logs or checking the internals.

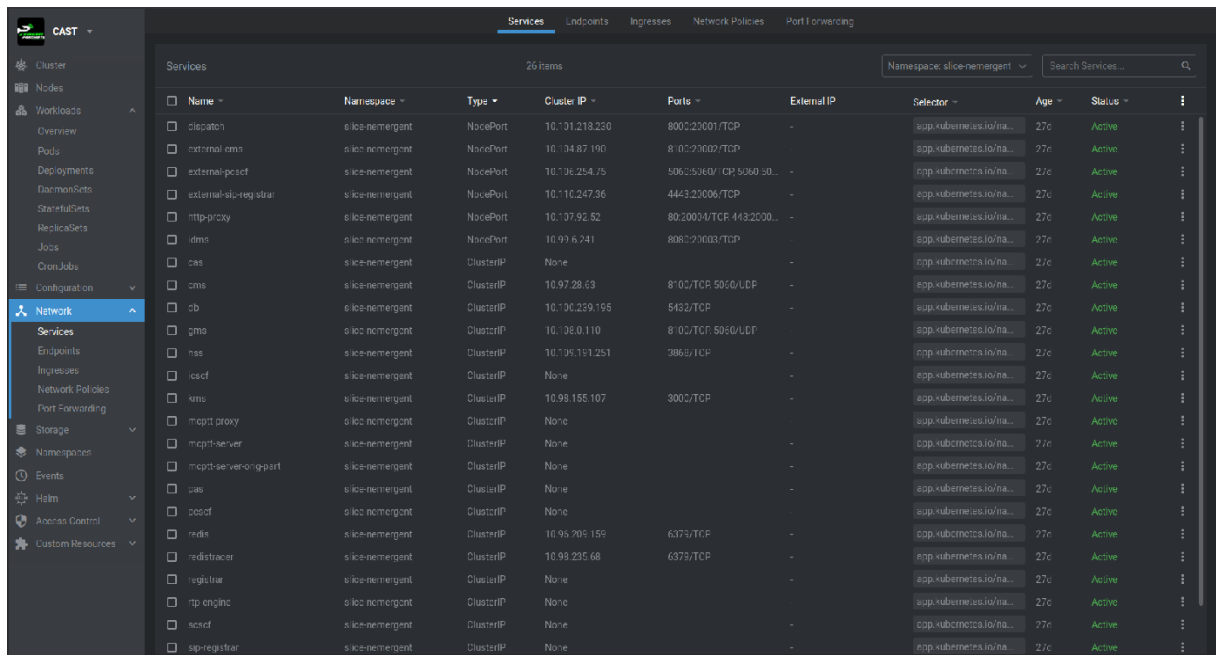




Name	Namespace	Containers	Restarts	Controlled By	Node	QoS	Age	Status
cae-0	slice-nemergent	1	0	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	6m35s	Running
cms-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
db-0	slice-nemergent	1	0	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	6m35s	Running
external-cms	slice-nemergent	1	1	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	6m35s	Running
external-slice-reg-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
has-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
http-proxy-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
icscf-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
ids-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
kms-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
pas-0	slice-nemergent	1	0	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	6m34s	Running
poscf-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
redis-0	slice-nemergent	1	0	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	6m31s	Running
redistracer-0	slice-nemergent	1	0	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	6m35s	Running
rfp-engine-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
sasf-0	slice-nemergent	1	71	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running
slice-registrar-0	slice-nemergent	1	333	StatefulSet	b2/a5b3a-f52-4a3a-a	BestEffort	27d	Running

Figure 78 Nemergent MCS service Pods deployed in Castelloli

As explained before, this entire MCS/MCX service provided by Nemergent is deployed at the moment using NodePort in Castelloli and this has a direct impact on the assigned “external” IP and port for those components that require external access from the client side. Figure 79 shows the service part of the deployment where the cluster IPs are shown alongside the mapping “external” port in the Ports field for all “NodePort” Type service.



Name	Namespace	Type	Cluster IP	Ports	External IP	Selector	Age	Status
dispatch	slice-nemergent	NodePort	10.101.218.230	8002/TCP	-	app.kubernetes.io/name=...	27c	Active
external-cms	slice-nemergent	NodePort	10.104.87.190	8102/TCP	-	app.kubernetes.io/name=...	27c	Active
external-poscf	slice-nemergent	NodePort	10.106.254.75	5062/TCP	-	app.kubernetes.io/name=...	27c	Active
external-slice-reg-0	slice-nemergent	NodePort	10.110.247.36	4443/TCP	-	app.kubernetes.io/name=...	27c	Active
http-proxy	slice-nemergent	NodePort	10.107.92.52	80/TCP	-	app.kubernetes.io/name=...	27c	Active
ids	slice-nemergent	NodePort	10.109.6.241	8082/TCP	-	app.kubernetes.io/name=...	27c	Active
pas	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
cms	slice-nemergent	ClusterIP	10.97.28.63	8102/TCP	-	app.kubernetes.io/name=...	27c	Active
db	slice-nemergent	ClusterIP	10.100.239.195	5432/TCP	-	app.kubernetes.io/name=...	27c	Active
gms	slice-nemergent	ClusterIP	10.108.0.110	8102/TCP	-	app.kubernetes.io/name=...	27c	Active
has	slice-nemergent	ClusterIP	10.108.131.251	3862/TCP	-	app.kubernetes.io/name=...	27c	Active
icscf	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
kms	slice-nemergent	ClusterIP	10.99.155.107	3002/TCP	-	app.kubernetes.io/name=...	27c	Active
mcpt-proxy	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
mcpt-server	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
mcpt-server-on-gp	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
pas	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
poscf	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
redis	slice-nemergent	ClusterIP	10.96.205.159	6379/TCP	-	app.kubernetes.io/name=...	27c	Active
redistracer	slice-nemergent	ClusterIP	10.99.235.68	6379/TCP	-	app.kubernetes.io/name=...	27c	Active
registrar	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
rfp-engine	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
sasf	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active
slice-registrar	slice-nemergent	ClusterIP	None	-	-	app.kubernetes.io/name=...	27c	Active

Figure 79 Nemergent MCS service deployed in Castelloli with NodePort

Additionally, the Emergency Communication pilot also takes advantage of the LongHorn storage class for those Pods that need stateful information stored in case of Pod restarts or stability. To this end, Nemergent uses 6 different persistent volumes that are directly requested or claimed in deployment time.

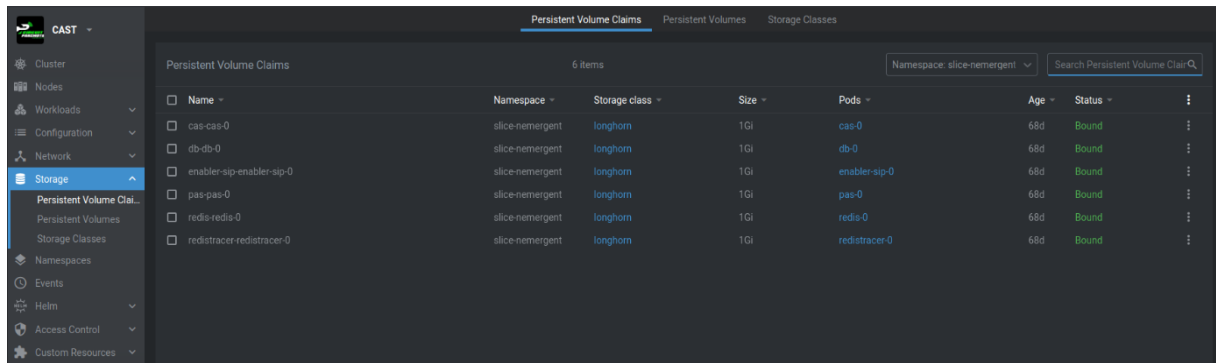


Figure 80 Nemergent MCS service deployed in Castellolí with storage classes

Once the service is completely deployed, the end-to-end testing can be fulfilled. In this regard Nemergent has conducted a thorough analysis of the service and has encountered several limitations when it comes to the infrastructure networking that directly impacts the proper reception of packets for the service. Next table summarizes the driven steps and the encountered blocking points.

Table 4 Status update of end-to-end testing of deployed Pilot 1 in Castellolí

Step	Proto.	UDP mode	TCP mode	Comment
UE-init config	HTTP	OK	OK	
IDMS	HTTP	OK	OK	
KMS key material	HTTP	OK	OK	
REGISTER	SIP	NOK	OK	MTU and fragmentation issue in UDP
SUBSCRIBE to CMS	SIP	Ongoing	OK	
Download user-profiles	HTTP	Ongoing	OK	
Download ue-config	HTTP	Ongoing	OK	
SUBSCRIBE to GMS	SIP	Ongoing	OK	
Client Auth. PUBLISH	SIP	Ongoing	NOK	Dropped packets in firewall
Additional subscriptions	SIP	Ongoing	Ongoing	

Until the networking burden is solved, Nemergent has proceed with a Plan B to factually assess the end-to-end service in a separate cluster controlled by NearbyOne. Once double-checked, this should be equally transposed to the actual Castellolí platform.

Nemergent continues working on the scaling-up mechanism and re-instantiation method so that both procedures are merged in the baseline service deployed in Castellolí.

## 8 CONCLUSIONS

---

This deliverable is an update of the previous deliverable in this work package: D4.1 Integration and Affordable 5G rollout plans. It includes the upgrade of both Malaga and Castellolí platforms achieved during the second year of the project lifetime, including the installation and deployment of enhanced products, developed prototypes and open platforms to carry out complete system tests. The installation and deployment are based on the methodology that has been described in detail in the previous deliverable, which is crucial for the execution of these tasks.

The focus of this document has been on the integrations performed for Affordable 5G components, the different Test Cases defined individually and also the integrations that already took place in the different platforms as well as initial services running on top of these 5G NPN testbeds in Malaga and Castellolí. As described along the document, numerous successful integrations have been performed, as the different partners demonstrated the result of hard work done along the project. Even in the cases that presented some slight deviations, alternative Test Cases were demonstrated.

Additionally, several taskforces (consisting of groups of partners) have started working on the different pilots, as explained in the sections 6, 7, 8 of this document, where the interplay and the requirements for the integration of components and services have been described in detail. Each pilot has the purpose of challenging the platform, not only to test the connectivity, but also each component and the overall performance of the systems. For example, the Time-Sensitive Networking over 5G offers a relevant innovation in terms of network evolution, being able to solve its main challenges: Translation from TSN domain to 5G, prioritization over 5G and time synchronization. The Smart City Pilot uses the same indoor network to validate a real environment scenario with strict requirements on latency and bandwidth demand, and the Mission Critical Services Pilot evaluates the outdoor private network in an emergency situation, using edge computing services.

Finally, this deliverable will be updated in D4.3, as the final deliverable of this Work Package. It will include the details of the end-to-end integrations and the validation results of the two Pilots demonstrations.

## 9 REFERENCES

- [1] Affordable5G, Deliverable D1.2: Affordable5G building blocks fitting in 5G system architecture [Online].
- [2] Affordable5G, Deliverable D4.1: Integration and Affordable5G roll-out plans [Online] - <https://www.affordable5g.eu/download/d4-1-integration-and-affordable5g-roll-out-plans/?wpdmdl=1230&masterkey=623307620ba59>, August 2021.
- [3] Affordable5G, Deliverable D3.2: Software developments release [Online] - <https://www.affordable5g.eu/deliverables>, March 2022.
- [4] Affordable5G, Deliverable D2.2: Hardware solutions release
- [5] 5G-ACIA White Paper, Integration of 5G with Time-sensitive Networking for Industrial Communications [Online] - [https://5g-acia.org/wp-content/uploads/2021/04/5G-ACIA\\_IntegrationOf5GWithTime-SensitiveNetworkingForIndustrialCommunications.pdf](https://5g-acia.org/wp-content/uploads/2021/04/5G-ACIA_IntegrationOf5GWithTime-SensitiveNetworkingForIndustrialCommunications.pdf)
- [6] SLI: O-RAN.WG1.Slicing: O-RAN Working Group 1 Slicing Architecture, v02.00, July 2020.
- [7] CLORAN: O-RAN.WG6.CAD: Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN, v02.01, July 2020.
- [8] OPNET: Stratum [Online] - <https://opennetworking.org/stratum/>, Accessed: June 2022.
- [9] OSPROG: P4 Open Source Programming Language [Online] - <https://p4.org/>, Accessed: June 2022.
- [10] Ublox modem specification - [https://www.u-blox.com/sites/default/files/SARA-R5-R4-GNSS-Implementation\\_AppNote\\_%28UBX-20012413%29.pdf#page=67&zoom=100,68,572](https://www.u-blox.com/sites/default/files/SARA-R5-R4-GNSS-Implementation_AppNote_%28UBX-20012413%29.pdf#page=67&zoom=100,68,572)
- [11] NEOX: <https://www.think-silicon.com/neox-graphics>.
- [12] NEMA: <https://www.think-silicon.com/nema-bits>
- [13] MEBN: 821fa74b50ba3f7cba1e6c53e8fa6845-Paper.pdf (neurips.cc).
- [14] CMC: Evaluation Metrics — Open-RelD documentation (cysu.github.io).
- [15] MAP: Mean Average Precision (mAP) Explained: Everything You Need to Know (v7labs.com).
- [16] JTSN: NVIDIA Jetson Nano Developer Kit | NVIDIA Developer.
- [17] REID: 2001.04193.pdf (arxiv.org).
- [18] BLLSQ: BullSequana S (atos.net).
- [19] WASABI: Creating a Bucket (wasabi.com).
- [20] YOLO: GitHub - theAIGuysCode/yolov4-deepsort: Object tracking implemented with YOLOv4, DeepSort, and TensorFlow.
- [21] TFX Airflow Tutorial, [https://www.tensorflow.org/tfx/tutorials/tfx/airflow\\_workshop](https://www.tensorflow.org/tfx/tutorials/tfx/airflow_workshop).
- [22] TFSD: TensorFlow Serving with Docker, <https://www.tensorflow.org/tfx/serving/docker>.
- [23] RDFSH: <https://www.dmtf.org/standards/redfish>.
- [24] WKRD: [https://en.wikipedia.org/wiki/Redfish\\_\(specification\)](https://en.wikipedia.org/wiki/Redfish_(specification)).

- [25] FAPI: <https://fastapi.tiangolo.com/> -
- [26] RABMQ: <https://www.rabbitmq.com/> -
- [27] UBPF: <https://urbanplatform.city/> -
- [28] VDGR: <https://abhitronix.github.io/vidgear/v0.2.5-stable/>