



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



AFFORDABLE 5G

D4.3 Pilot validation report

Revision : v.1.0

Work package	WP 4
Task	Task 4.5
Due date	21/12/2022
Submission date	22/12/2022
Deliverable lead	UMA
Version	1.0

Abstract

This final report describes the system tests and pilot validation execution and the final comprehensive technical KPI analysis and validation, done in T4.5. These system tests and pilot validation is mainly performed at Malaga and Castellolí platforms, starting with a network characterization of both and, afterwards, validations of TSN over 5G PoC and Smart City and MCS pilots.

Keywords: 5G Non-Public Networks, O-RAN, AI/ML-based network optimization, pilot validation.

List of Contributors

Partner	Short name	Contributor(s)
ATOS SPAIN SA	ATOS	Sergio González Josep Martrat Borja Otura García
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 Ronzani Daniele Munaretto
THINK SILICON EREYNA KAI TECHNOLOGIA ANONYMI ETAIRIA	THI	Georgios Keramidas
RUNEL NGMT LTD	REL	Israel Koffman
NEMERGENT SOLUTIONS S.L.	NEM	Andoni Díaz de Cerio Iván González Eneko Atxutegi Marta Amor
UBIWHERE LDA	UBI	Roni Sabença Rita Santiago
MARTEL GMBH	MAR	Gabriele Cerfoglio, Andrea Falconi
NEARBY COMPUTING SL	NBC	Oscar Trullols Angelos Antonopoulos
UNIVERSIDAD DE MALAGA	UMA	Pablo Herrera Javier Jiménez Francisco Luque Pedro Merino
ETHNIKO KAI KAPODISTRIAKO PANEPISTIMIO ATHINON	NKUA	Sotiros Spantideas Panagiotis Trakadas
FUNDACIO PRIVADA I2CAT, INTERNET I INNOVACIO DIGITAL A CATALUNYA	I2CAT	Juan Sebastian Camargo Raul Blanxart, J.J. Aleixendri, Daniel Bautista
EURECOM	EUR	Ilario Favero
UNIVERSITAT POLITECNICA DE CATALUNYA	UPC	Jordi Pérez-Romero

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	13/10/2022	ToC	UMA
V0.2	03/11/2022	First round of contributions	ACC, ATH, NEM, UBI, NKUA, UMA
V0.2	17/11/2022	Second round of contributions	MAR
V0.3	01/12/2022	Third round of contributions	UBI, NEM, NBC, ATH, NKUA, THI, ACC, UMA
V0.4	15/12/2022	Final round of contributions	UBI, NEM, NBC, ATH, NKUA, ACC, UMA, EURE, I2CAT, UPC, MAR, ATOS
V0.5	16/12/2022	Final contribution from UBI	UBI
V0.6	21/12/2022	Final contribution from CEL	CEL
V0.9	21/12/2022	Version after internal review	ATOS, NKUA, UMA
V1.0	22/12/2022	Quality check & submission to EC	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

Nature of the deliverable:		R	
Dissemination Level			
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

The Affordable5G project aims to provide cost-efficient deployments of private 5G networks able to support a variety of pilots. To achieve that, the consortium is providing the necessary infrastructure, based on the integration of different 5G components that have been developed during the project, improved products, and some 3rd party COTS elements to conform the final end to end 5G SA solution (O-RAN, Edge, 5GC and orchestration capabilities). In the end, two different instantiations of the Affordable5G network solution are presented on the two testbed platforms, one in Castellolí circuit and another in Malaga campus.

This deliverable describes the final picture of both testbeds, including the developed building blocks that are integrated, but with the focus on the pilots and their validation. The first one, Mission Critical Services (MCS) pilot is deployed in Castellolí. This pilot shows the integration of Mission Critical services in a 5G network, taking most of the 5GCore integration in an efficient way. The second one is the Smart City pilot, which has been deployed in Malaga and, using 5G network, demonstrates real time video analytics for mobility surveillance based on ML algorithms. Finally, the third case so called TSN over 5G Proof of Concept, is not complete pilot, but a first PoC approach to deterministic communications over a real 5G network which is of great relevance for industrial scenarios.

System integration work is a critical process that cannot underestimated when addressing a multi-vendor environment. It is not easy to put together many developments to converge in a full operational system level solution. One of the main issues was the difficulties to elaborate a mature O-RAN solution. Alternative O-RAN elements were explored for both testbeds which in turn came with new integrations challenges that conditioned the smooth deployment of pilots. Even so, Affordable5G has allowed partners to improve their technologies, to enhanced products and to evolve and test innovative 5G features within project's framework.

Regarding validation, not only a pilot's validation is included, but also a network characterization of both platforms, which is necessary to know network's performance limitations before the pilots are tested. Finally, pilot's validation itself is included, defining system-level test cases, and different scenarios to measure relevant KPIs.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	4
TABLE OF CONTENTS.....	5
LIST OF FIGURES.....	7
LIST OF TABLES	10
ABBREVIATIONS.....	11
1 INTRODUCTION.....	13
2 UPDATED PILOTS DEFINITION.....	14
2.1 TSN over 5G PoC	14
2.1.1 O-RAN	14
2.1.2 Additional UE and updated functionality on TSN translators	14
2.1.3 Integration of Ball Balancing Table.....	15
2.2 Smart City.....	15
2.3 MCS	16
3 VALIDATION METHODOLOGY	19
3.1 OpenTAP.....	19
4 INTEGRATIONS OUTSIDE THE MAIN SITES	21
4.1 RU-DU integration approach.....	21
4.1.1 Eurecom Testbed Components	21
4.1.2 RunEL preparations for the RU-DU Integration over O-RAN 7.2 interface.....	24
4.1.3 Issues encountered and further steps	25
4.2 Telemetry Module Deployment in Far Edge Devices	25
4.3 O-RAN Fronthaul & S-Plane: ACC Labs	26
5 MALAGA TESTBED	30
5.1 Final picture.....	30
5.2 Network characterization	31
5.3 TSN over 5G PoC	35
5.3.1 Building blocks.....	35
5.3.2 Configuration and automation interfaces.....	37
5.3.3 Configurable parameters and KPIs	37
5.3.4 Test cases and results	38
5.3.5 Comparison with Nokia RAN.....	41
5.3.6 Innovations and conclusions	42
5.4 Smart City Pilot.....	43
5.4.1 Building blocks.....	43
5.4.2 Configuration and automation interfaces.....	50
5.4.3 Configurable parameters and KPIs	50
5.4.4 Test cases and results in THI platform	51
5.4.5 Test cases and results	52
5.4.6 Innovations and conclusions	65
6 CASTELLOLI TESTBED.....	66
6.1 Final picture.....	66
6.2 Fronthaul challenges	68
6.3 Network characterization	69
6.3.1 Slice 1 integration tests and final results	70
6.3.2 Slice 2 integration tests and final results	72
6.4 MCS Pilot.....	76
6.4.1 Building blocks.....	76

6.4.2	Configuration and automation interfaces.....	78
6.4.3	Configurable parameters and KPIs	82
6.4.4	Test cases and results	86
6.4.5	Innovations and conclusions	98
7	CONCLUSIONS.....	100
	REFERENCES	101

LIST OF FIGURES

Figure 1: TSN over 5G PoC final architecture.....	14
Figure 2 : Actions steps representation of scenario 1	17
Figure 3: Action steps representation of scenario 2.	18
Figure 4 : Action steps representation for extended scenario 2	18
Figure 5: OpenTAP interface.....	19
Figure 6: OpenTAP Instruments interface	20
Figure 7: OpenTAP Results configuration.....	20
Figure 8: Eurecom testbed	21
Figure 9: Message sequence chart in FHI integration.....	22
Figure 10: Message sequence chart for TX buffer	23
Figure 11: Sample 7.2 Wireshark traces	23
Figure 12: RU interfaces.....	24
Figure 13: RU internal structure	24
Figure 14: Affordable5G: Initially designed Castellolí fronthaul & S-Plane infrastructure 27	
Figure 15: Simplified Affordable5G Castellolí fronthaul for minimized delay/jitter	27
Figure 16: Spectrum analyzer output. Drift of frequency with TSN switch.....	28
Figure 17: Benetel 550 radio unit	30
Figure 18: Physical server hosting O-RAN solution.....	30
Figure 19: Final architecture of Malaga platform.....	31
Figure 20: Malaga testbed baseline latency	32
Figure 21: Malaga testbed baseline jitter	32
Figure 22: Malaga testbed UDP DL Throughput	33
Figure 23: Malaga testbed TCP DL Throughput.....	33
Figure 24: Malaga testbed UDP UL Throughput	34
Figure 25: Malaga testbed TCP UL Throughput.....	34
Figure 26: Setup of TSN over 5G PoC	35
Figure 27: ADVA FSP 150 equipment x2.....	35
Figure 28: Workstations which run TSN translators.....	36
Figure 29: Ball Balancing Table.....	36
Figure 30: Network latency using Nokia RAN	42
Figure 31: Network jitter using Nokia RAN.....	42
Figure 32: Static Service current deployment	44
Figure 33: Overall static service workflow	45
Figure 34: Prioritization over 5G network in Missing Child scenario.....	45
Figure 35: High-level architecture of Missing Child scenario	46

Figure 36: High-level dataflow of Missing Child scenario.....	46
Figure 37: Dynamic service building blocks.....	47
Figure 38: Detailed view of overall dynamic service system	48
Figure 39: Deployed service in OSM.....	54
Figure 40: Service x logs in the OSM.....	54
Figure 41: Logs presented by the /metrics endpoints that showcase the metrics of the service 54	
Figure 42: Timestamp of the time it takes the frame to be processed by the CPU by the Edge 55	
Figure 43: Timestamp of the time it takes the frame to be processed by the GPU by the Edge 56	
Figure 44: Core videgear client, booting the process of receiving the frame from the Edge. With CPU enabled	57
Figure 45: Fronted service, showing the processed frame	58
Figure 46: Core logs with the CPU configuration enabled.....	58
Figure 47: Security guard smartphone in the higher prioritisation profile.....	60
Figure 48: Security guard smartphone during a lost child occurrence, in lower prioritisation profile	60
Figure 49: logs of the cropping and detection operation	61
Figure 50: Some cropped detections of individuals detected by the YOLOv5	63
Figure 51: Missing kid photo match, with threshold of 0.8.....	63
Figure 52: Provided missing kid photo	63
Figure 53: Mobile application interface	64
Figure 54: Ground truth process.....	64
Figure 55. Final architecture of Castellolí platform.....	66
Figure 56: Node #1 RAN installation	67
Figure 57: Node #9 RAN installation	67
Figure 58. Control Room	68
Figure 59: Successful message of UE attachment.	70
Figure 60: Details of successful message, showing the supported slices.	70
Figure 61: Example of a failed UE attachment with UE model DOGEE 5G	71
Figure 62: Smartphones trying to connect to the Affordable 5G network.....	72
Figure 63 NearbyBlocks list showing Athonet UPF Deployment and Generic Slice Descriptor.....	73
Figure 64: Vendor Independent Slice Descriptor Block.....	73
Figure 65: Athonet Dynamic UPF Deployment Block.....	74
Figure 66: Athonet upf2 dashboard with upf active.....	74
Figure 67: Athonet upf2 dashboard with upf inactive	75
Figure 68: Castellolí platform Network test.....	75

Figure 69: Castelloli's platform Network characterization parameters	75
Figure 70 : Building blocks architecture for scenario 1.....	76
Figure 71: Building blocks architecture for scenario 2.....	77
Figure 72: Building blocks architecture for extended scenario 2	78
Figure 73: Software components involved in slice provisioning.....	79
Figure 74: Workflow followed by CUOM to deploy a new SNSSAI	79
Figure 75: Provisioning of a new operator in CU-UP and CU-CP.....	80
Figure 76: Provisioning of new slices under existing operator in CU-UP	80
Figure 77: Resulting configuration in CU-UP	81
Figure 78: Slice Manager's response.....	82
Figure 79: Slice configuration log after the slice creation	82
Figure 80: RTT MCX system KPI call-flow	83
Figure 81: MCPTT Access Time (KPI1) call-flow.....	84
Figure 82: MCPTT E2E Access Time call-flow.....	84
Figure 83 : Dimensionality of the Deep Neural Network trained to predict the MCS service overload	85
Figure 84: NWDAF deployment through the designer tab in Nearby's k8s cluster.....	86
Figure 85: NWDAF component's list of deployed pods.....	87
Figure 86: MCX deployment through the designer tab in Nearby's k8s cluster.....	87
Figure 87: MCX system's list of deployed pods.....	87
Figure 88: MCX user registration procedure.....	88
Figure 89: Ongoing MCX group call	88
Figure 90 : Confusion matrix regarding the validation accuracy of the ML model training process.....	89
Figure 91 : Log showing the model prediction outputs for given 9-valued inputs.....	89
Figure 92: Prometheus illustrating the prediction results in the upper plot (switching from 0 to 1), the number of registered users in the middle plot (increasing from 0 to 4) and the number of active group calls in the lower plot (from 0 to 1)	90
Figure 93: MCX "cas" and "pas" pods are being reinstantiated due to the prediction received by the orchestrator	91
Figure 94: MCX system's running pods cas-0, pas-0 and pending cas-1, pas-1	91
Figure 95: Steps for multi-PoP feature accomplishment.....	93
Figure 96: Detailed diagram describing the interaction in multi-PoP validation.....	94
Figure 97: Boxplot representing the KPI obtained.....	97

LIST OF TABLES

Table 1: Telemetry integration in THl platform25

Table 2: Slicing and associated SIMs configuration schema. Note there is a SIM having access to both slices (highlighted in bold).69

Table 3: Scenario 1 related functional test case sequence86

Table 4: KPI results from Scenario 1 - Time elapsed between the prediction is received and the MCX service is reinstantiated92

Table 5: Scenario 2 related functional test case sequence92

Table 6: Detailed KPI values for the iterations carried out97



ABBREVIATIONS

3GPP	Third Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation
5GC	5G Core
AI	Artificial Intelligence
AMF	Access and Mobility Function
AMR	Autonomous Mobile Robots
AUSF	Authentication Server Function
CP	Control Plane
CU	Central Unit
DL	Downlink
DN	Data Network
DNN	Data Network Name
DS-TT	Device Side Translator
DP	Data Plane
DU	Distributed Unit
E2E	End to End
eMBB	enhanced Mobile Broadband
FHI	Fronthaul interface
IP	Internet Protocol
KPI	Key Performance Indicator
MCS	Mission Critical Service
MCPTT	Mission Critical Push-To-Talk
MEC	Multi-access Edge Computing
ML	Machine Learning
mMTC	massive Machine Type Communications
NF	Network Function
NFV	Network Function Virtualization
NN	Neural Network
NPN	Non-Public Network
NR	New Radio
NT	Network Telemetry
NWDAF	Network Data Analytics Function
NW-TT	Network Side Translator
OAI	Open Air Interface
OS	Operative System
OSM	Open Source Mano
PCF	Policy Control Function
PLMN	Public Land Mobile Network
PM	Performance Monitoring
PPDR	Public Protection and Disaster Relief
P4	Programming Protocol-independent Packet Processors
QoS	Quality of Service
RAN	Radio Access Network
RU	Remote Unit
SA	Standalone
SST	Slice/Service Type
SD	Slice Differentiator
SMF	Session Management Function
TCP	Transmission Control Protocol
TSN	Time Sensitive Networking
UDM	Unified Data Management

UDR	Unified Data Repository
UE	User Equipment
UP	User Plane
UPF	User Plane Function
URLLC	Ultra Reliable Low Latency Communication

1 INTRODUCTION

This deliverable is the final report with respect to WP4. Its main focus is on pilot's final development and validation. To this end, the document starts in section 2 with the update of pilot's definition, where all work done during last months to define the final structure of the pilots is presented. Later, in section 3, a brief overview of the validation methodology that is going to be followed in test validation is described. This methodology was already presented in D4.1 [2], so in case a more detailed view is needed, the suggestion is to see the corresponding document. After that, section 4 is dedicated to integrations that are not located in Malaga or Castellolí testbeds. These are integrations that, for reasons that can be found in the section, could not be integrated in the final locations. In section 5, Malaga testbed's final status is presented. First, the final architecture view is depicted, highlighting updates since last report. Secondly, network characterization based on most relevant KPIs is provided. Once network architecture and characterization are presented, both pilots deployed in Malaga testbed can be addressed. On the one hand, TSN over 5G Proof of Concept is described in detail, presenting building blocks that compose the pilot architecture, configuration and automation interfaces and scenarios based on combinations of KPIs to be measured and parameters to be modified. Finally, related test cases definition and results are provided. A last subsection regarding innovations and conclusions about the pilot is included. On the other hand, Smart City pilot is also presented, following the same substructure of sections as defined for the previous pilot. With that, Malaga testbed is totally described. Afterwards, section 6 includes similar information as described for Malaga platform but regarding Castellolí testbed. The structure is quite similar, but the only pilot deployed in Castellolí is MCS Pilot. A wide description of such pilot, which involves the collaboration of several partners, can be found in such section. Lastly, section 7 includes the conclusions of this deliverable.

2 UPDATED PILOTS DEFINITION

2.1 TSN over 5G PoC

In the case of Time Sensitive Networking (TSN) over 5G Proof of Concept (PoC), the final architecture was already defined and was presented in previous deliverable D4.2 [1]. The architecture can be found in Figure 1.

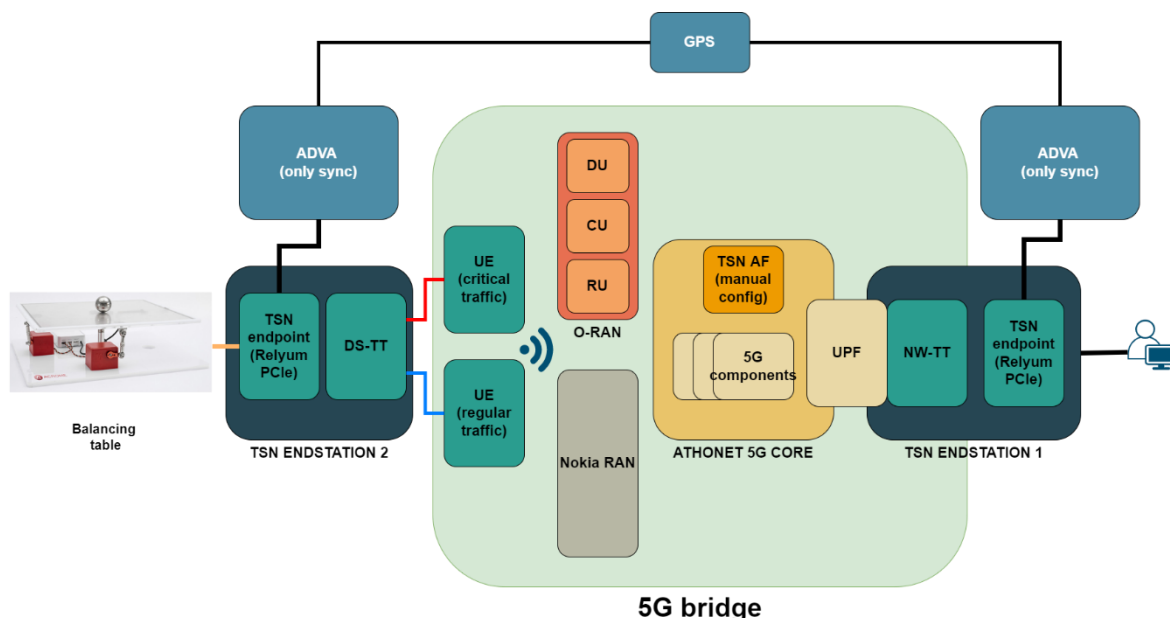


Figure 1: TSN over 5G PoC final architecture

This architecture was defined as the final version but in D4.2 it wasn't deployed yet. Since then, several updates have been performed. However, we won't go through the details of the functionality, related to translation, synchronization and prioritization, as they were already explained in D4.2 and can be checked there. In this section, we will focus on these updates, explaining in detail the developments and integration efforts.

2.1.1 O-RAN

O-RAN has been successfully integrated with ATH 5GC and end to end (e2e) functionality has been tested with our user equipment (UE) Telit fn980m. It is important to remind that this O-RAN solution has been fully provided by ACC. More details about this solution are provided in section 5. This is the main RAN that has been integrated during the project in the Malaga platform; however, the already existing Nokia RAN still remains operative just to perform some comparative tests and check differences between both solutions in the scope of this PoC.

2.1.2 Additional UE and updated functionality on TSN translators

An additional UE has been integrated in this PoC. As explained in D4.2, in the final architecture two UEs are needed, one for critical traffic and another one for regular traffic. This is the way the mapping between TSN domain and 5G domain is achieved, allowing the management of different priorities through the 5G network. In addition, to support this additional UE, an update in the TSN translators (NW-TT and DS-TT) has been performed as well. It consists of the addition of new rules to differentiate the traffic coming from one UE (critical traffic) to the other one (regular traffic). Moreover, this PoC was expected to work only in downlink (sending traffic from NW-TT to DS-TT). However, during the integration of the Ball Balancing Table which, as explained in D4.2, will act as visual demonstrator for this PoC, it was noticed that it was also

necessary to define the Uplink (UL) path. To achieve that, complementary rules have been configured in the TTs to support such traffic.

2.1.3 Integration of Ball Balancing Table

As mentioned above, the integration of Ball Balancing Table (BBT) has inserted a challenge in terms of routing and translators' update. Specifically:

- New routes have been added on the 5GC to use the Telit UE for critical traffic as Gateway (GW) to the two Raspberry Pi controlling BBT. Basically, this indicates to the core that all traffic with destination BBT has to be sent through Telit UE.
- Update of the TT to support IP matching and to include UL rules to send ball position to the controller. In particular, DS-TT has to differentiate the traffic to the Raspberry Pi controlling servomotors and the Raspberry Pi controlling touchscreen.

2.2 Smart City

The SmartCity pilot aims to showcase the usage of a 5G private network in emergency scenarios. Through the development and integrations phases, the pilot underwent multiple iterations but always maintained the core focus on ML algorithms and the 5G network capabilities.

As such, the pilot definition has persisted, being the demo scenario, a host in-door loss of a child in a crowded shopping mall. This scenario is challenging for the ML person detections since a cramped location makes it difficult to have fluid detection without causing substantial FPS loss and downgrading in the 5G network quality, since there is expected to be a great amount of UEs belonging to the people which are consuming and transmitting high amounts of data.

Thus, for this scenario two services were developed: a dynamic and a static one.

The dynamic service aims to find a missing child thanks to a re-id ML algorithm, as such several components were developed being the main ones:

- Smartphone App: When a missing child alert is active, it will transmit the captured frames via websocket to the Yolo detector.
- Yolo detector: that will detect, crop the person and send it to the re-id ML.
- Re-id: that compares the missing child photo with the detections performed by the Yolo detector.

The parents of the lost child will provide the picture of the kid to serve as a dataset as well as any relevant information to an API. Which in turn, will propagate the information to the other services changing the prioritization in the network for the Security guard smartphone. To ensure a better quality of data transmission, it will be needed to temporarily put his device in flight mode.

Then proceeds to search for the kid, pointing the smartphone camera to any pedestrian he faces. If a match occurs, a pop-up message will appear with the photo of the lost children as well as the image that provoked the match. Requesting the confirmation if both images represent the same person, if the detection is confirmed the response will be propagated to the remaining services and the Security Guard will temporarily put his device in flight mode. Finally, the Security guard will deliver the children to the parents in the security post, optionally it can contact them once the confirmation is done.

Additionally, the application allows to define the frame rate on which the frames are sent to the Yolo detector, this permits to mitigate the limitation of process capability of the hardware.

A more detailed description is shown in the building blocks section.

The static service is an additional tool for the security guard to find the missing children since it consists of a security CCTV system. That allows to visualize previous person detections as well as the ongoing ones.

2.3 MCS

The vocation of the emergency pilot on the Affordable 5G test platforms has undergone several modifications throughout the project lifetime.

In the initial pilot conception, we envisioned to work on three scenarios regarding 3GPP-compliant Mission Critical service, the core of the Emergency Communications use case. In order to respond to service load, infrastructure failure and detected delays, service instantiation, location and service master-slave balance were considered.

At this stage of the project, and having invested time and dedication in the cloud-nativization of the service in order to integrate MC services in 5G networks dynamically and in an efficient and cost-saving way, as well as in obtaining and processing the service KPIs, the most significant scenarios adopted to show the capabilities of both the service and the underlying network are the following:

Scenario 1: **MCS service scaling - capacity**

The current scenario's scope is to give response to a load increase regarding Mission Critical service utilization in case of an emergency event (e.g., increase of the combination of registered users and simultaneous groups calls within the registered users).

Figure 2 below depicts the steps performed for Scenario 1.

At first, a single instance of MCS service docker components in the "server-side" will be deployed in the "server-side" using the HelmChart.

- 1) Metrics of the service such as number of active private calls, number of active groups, calls and number of registered users will be exposed and consumed by the NWDAF/Telemetry System module.
- 2) The NWDAF and associated AI/ML module based on the metrics received will predict changes in the service behaviour and alarms will be triggered to anticipate any change in the service performance.
- 3) The resulting alarms will be communicated to the orchestrator.
- 4) The orchestrator consequently will trigger the required actions (service scalability) to face an eventual load increase in the service. The most critical components in terms of load are usually the PAS and CAS in the MCS service.

- 5) Once the target service components (MCX PAS and MCX CAS) are scaled-up, the load balancer within the service will guarantee the integrity of the data and rebalance the load of the service for all the upcoming traffic.

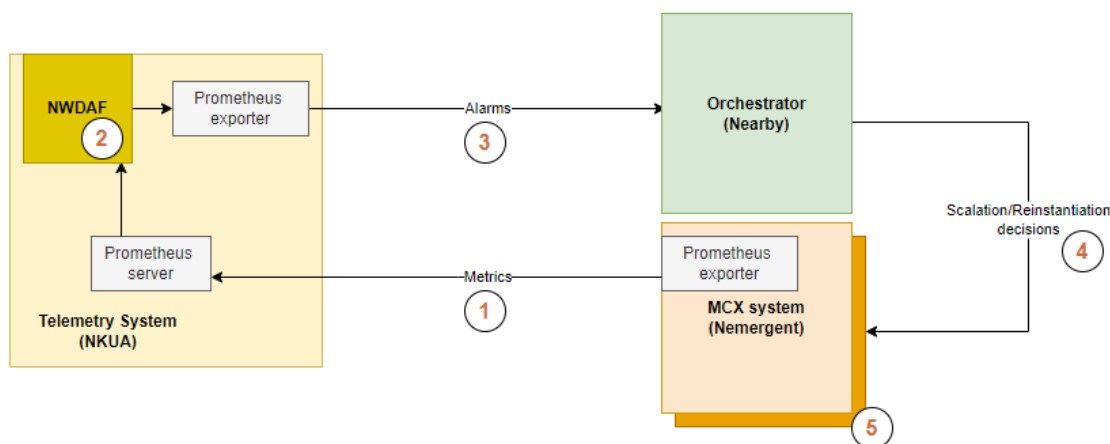


Figure 2 : Actions steps representation of scenario 1

Scenario 2: Dedicated slicing for emergency service bodies

The scope of Scenario 2 is to demonstrate the exploitation of network slicing to support the intervention of a PPDR team for an emergency event on the field. Thanks to the ad-hoc deployment of a dedicated network slice, the UEs of the PPDR team will be granted access to reserved network resources at the edge. This simultaneously guarantees isolation for any sensitive data flowing through the dedicated slice and improved QoS when the slice utilized by all the other users in the network can only offer insufficient best-effort services.

In Figure 3, the entire Scenario 2 is illustrated, with the steps of the storyline indicated in numerical order.

At first, a single slice (Slice 1) is deployed, and normal users are served by it by means of best effort QoS.

- 1) When an emergency event occurs (detected by system KPIs regarding failures or direct notification) it is notified to an emergency control center.
- 2) The emergency control center triggers the appropriate network orchestrator to request a dedicated emergency service.
- 3) At this point, the orchestrator asks the creation of a dedicated slice (Slice 2), interacting with the Central 5GC via API to activate it. The slice will be made available for PPDR users with higher required QoS for their specific emergency service and will be instantiated considering the emergency area's closest UPF.
- 4) This request leads to the actual instantiation of a second UPF (user plane component of the second slice), which has the purpose of serving the emergency traffic.
- 5) After re-shaping the network, the orchestrator deploys the required Mission Critical service instances at the edge for the emergency service bodies or PPDR usage close to the selected UPF in the emergency area.

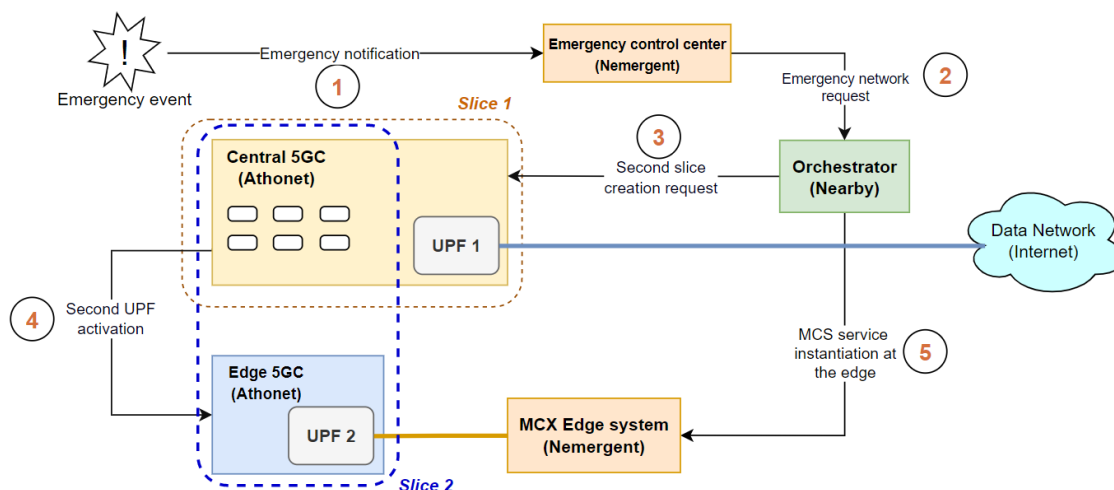


Figure 3: Action steps representation of scenario 2.

Extended scenario 2: MCS service reallocation in Multiple Point of Presence (PoP)

The scope of this scenario is to be able to take advantage of the relocation capabilities of the mission critical service in a more appropriate PoP for more optimized placements closer to the emergency events.

- 1) At first, a single and dedicated slice is present and PPDR users are served by it and can access the cloud-native MCS service.
- 2) For specific reasons according to placements algorithms, manual intervention or cost-related decisions, the service needs to be reallocated in the most seamless way for the PPDR users using the service. The MCS service will consequently be deployed to the destination PoP.
- 3) It will then sync with the service-specific stateful information (all data regarding users, groups and status of the dialogs and procedures).
- 4) And the remaining pods instantiation will occur.
- 5) Once ready it will completely swing and trigger an automated reconnection of the clients, so the clients start using the newly deployed service in the destination PoP and stop using the origin PoP one.

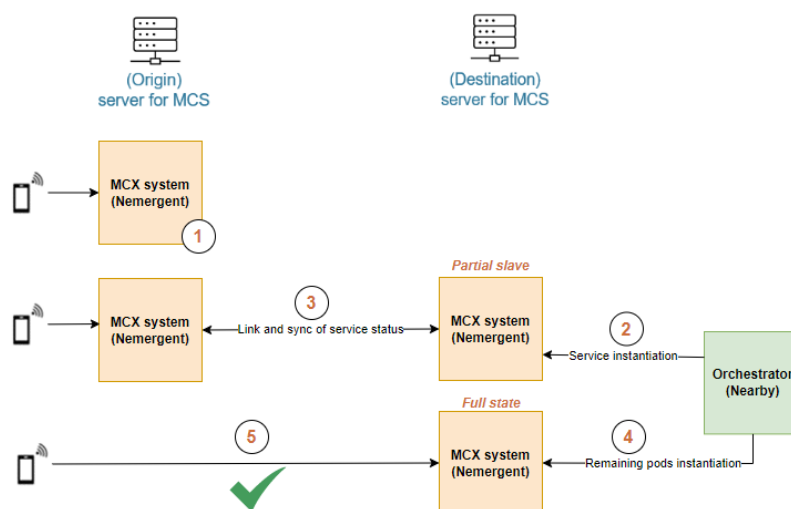


Figure 4 : Action steps representation for extended scenario 2

3 VALIDATION METHODOLOGY

The validation methodology was already explained in detail in D4.1 [2]. However, a reduced and more practical version based on the use of OpenTAP is added in this document in order to be used as an indicative guide for the execution of test cases in both platforms.

The idea is to define a common methodology to secure automation and repeatability in the measurement of the different KPIs. In addition, a discussion about where to store the results is included.

3.1 OpenTAP

OpenTAP is a test sequencing engine that provides functionality for the definition of the test logic, the management of heterogeneous devices (configuration, control and measurement extraction) and the handling of results within a single application.

In order to support a wide variety of components (both hardware and software based), OpenTAP includes a set of plugins that provide support for common requirements out of the box that will be enough for the purpose of this project.

Figure 5 shows the OpenTAP interface with a brief example of a Test Plan.

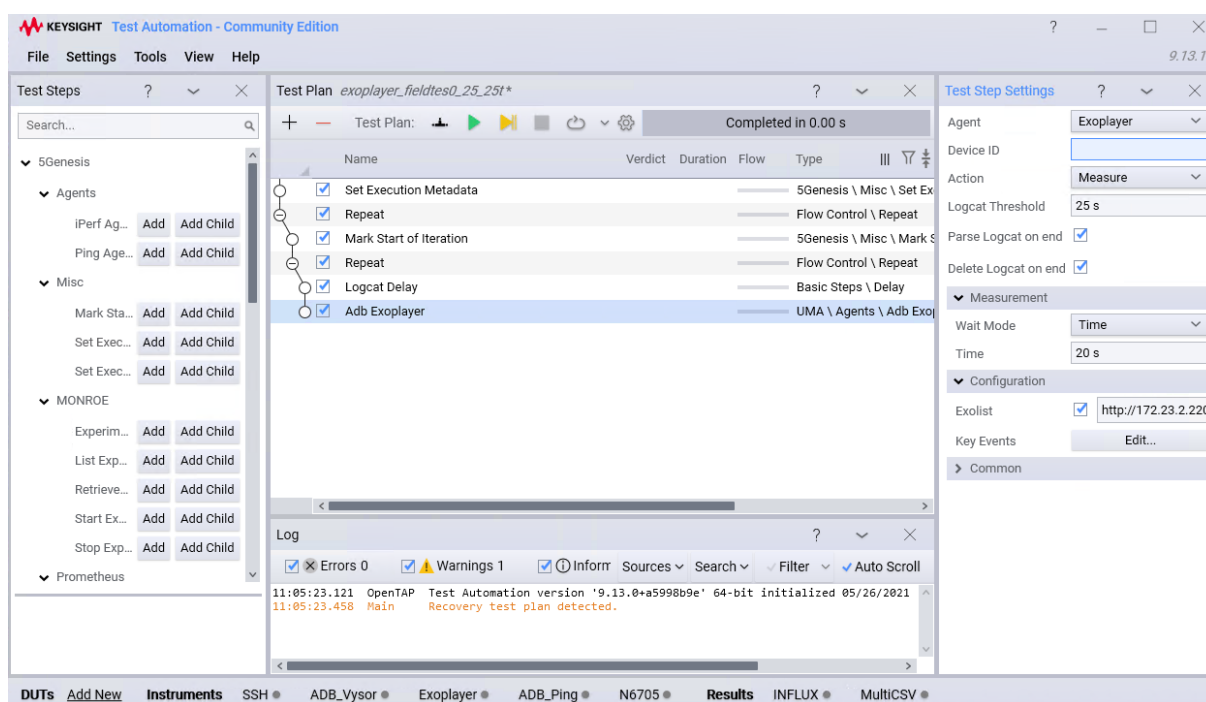


Figure 5: OpenTAP interface

Test cases are implemented in the form of TAP TestPlans. A Test plan is a collection of Test Steps, which in turn define each basic or complex action that is performed during a TestPlan execution, arranged in a tree structure. This structure, along with the usage of specific Test Steps define the order and logic implemented during a TestPlan execution.

In order to abstract the specific details and functionality used for the management of a particular component, OpenTAP separates the concepts of Instrument (Figure 6) and DUT (Device Under Test). Though functionally equivalents, Instruments are normally used to

encapsulate and abstract the use of measurement equipment and other testbed devices, while the DUT concept is reserved for the kind of devices that are subject to the testing procedure.

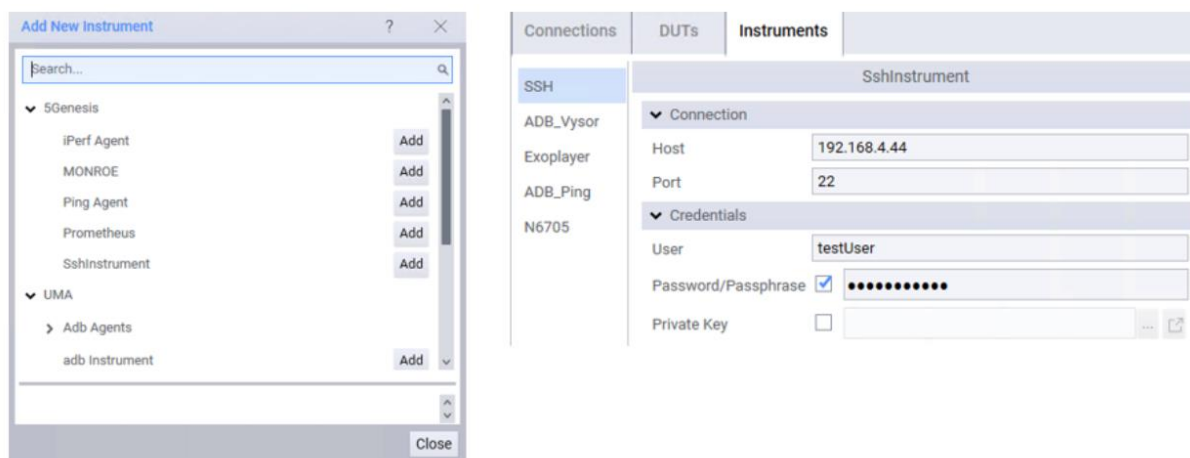


Figure 6: OpenTAP Instruments interface

Finally, in order to handle the storage of results, OpenTAP introduces the concept of *ResultListener*. In OpenTAP, results generation is decoupled from the actual storage of the measurements, which improves extensibility and eases the management of both processes:

- Results are generated by *Test Steps*. Each step may generate results as needed, which are then *Published* in the form of *ResultTables*. These tables may contain any number of rows and columns.
- The published *ResultTables* are retrieved by any number of enabled *ResultListeners* and are handled by each one of them independently.

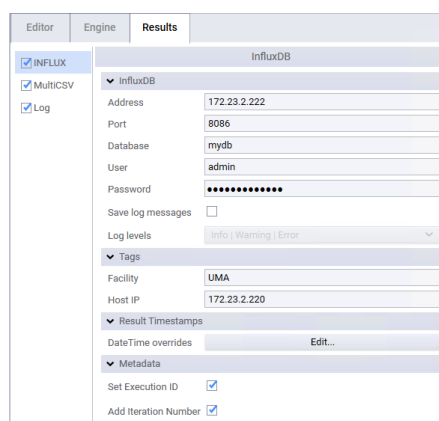


Figure 7: OpenTAP Results configuration

By using this architecture, end users are able to configure and use as many different storage solutions as needed, encapsulating the specific details of each solution to a separate *ResultListener*. For example, Figure 7 shows a setup where all log messages are saved to a text file, while results are recorded at the same time in an InfluxDB database and as multiple CSV files in the hard disk.

4 INTEGRATIONS OUTSIDE THE MAIN SITES

4.1 RU-DU integration approach

Given the novelty of the interoperability required and the complexity of the settings for the testbed that involved third party code like the Fronthaul O-RAN library, we choose a multi-steps gradual incremental approach for integrating the RU from RunEL into the OAI DU. We run the tests with the and without O-RAN Fronthaul library on multiple third-party RUs to have more data and to compare different scenarios. A simulation sample application was also used to validate the O-RAN Fronthaul library and its configuration settings. Finally, we performed the whole integration with RunEL RU and OAI.

4.1.1 Eurecom Testbed Components

Below, in Figure 8, you can find the description of the testbed used that successfully validated the FHI 7.2 interoperability with Commercial Off-the-Shelf (COTS) RUs. The Foxconn RU uses a Nvidia L1 application and L2/L3 OAI software stack, while the STL RU uses the OAI and O-RAN Fronthaul libraries. The RUs from Mavenir, VVDN and RunEL using the O-RAN Fronthaul libraries are still in the process of being tested but will occupy the same positioning of the STL RU in the architecture.

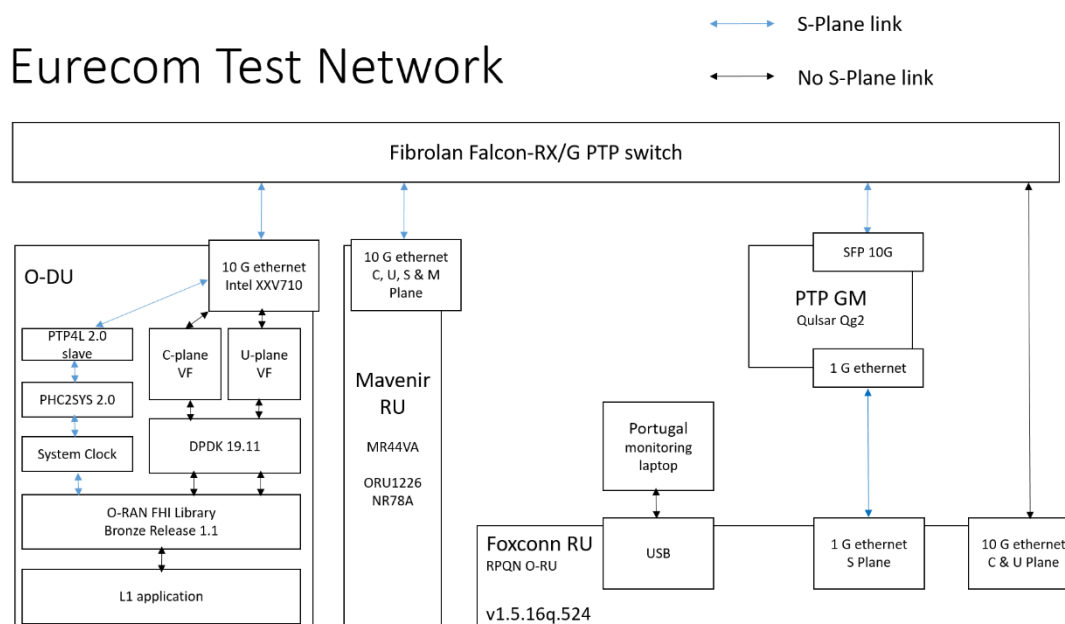


Figure 8: Eurecom testbed

The message sequence chart of this integration is shown in Figure 9. Different steps are performed to start the library, which are:

- Synchronization of the Fronthaul Interface (FHI) with the PPS coming from the Grand master, see Qulsar in the figure.
- Library initialization, that resets all the variables and the buffers used to exchange between the DU and FHI. Example, the bandwidth and compression.
- Open and configuration of the FHI library including setting up the receiver call back and the transmission ring buffer.
- Initialization of the DU.

After this stage, the DU and FHI are ready for Transmission (TX) and Reception (RX). On the RX, the DU will receive the timestamp and data through the call back, and on the transmission,

the DU calculates the TTI as a function of the timestamp, collects the TX data for the next TTI, and then writes the data to the TX buffer corresponding to that TTI. Note that the TX buffer is organized based on the TTI.

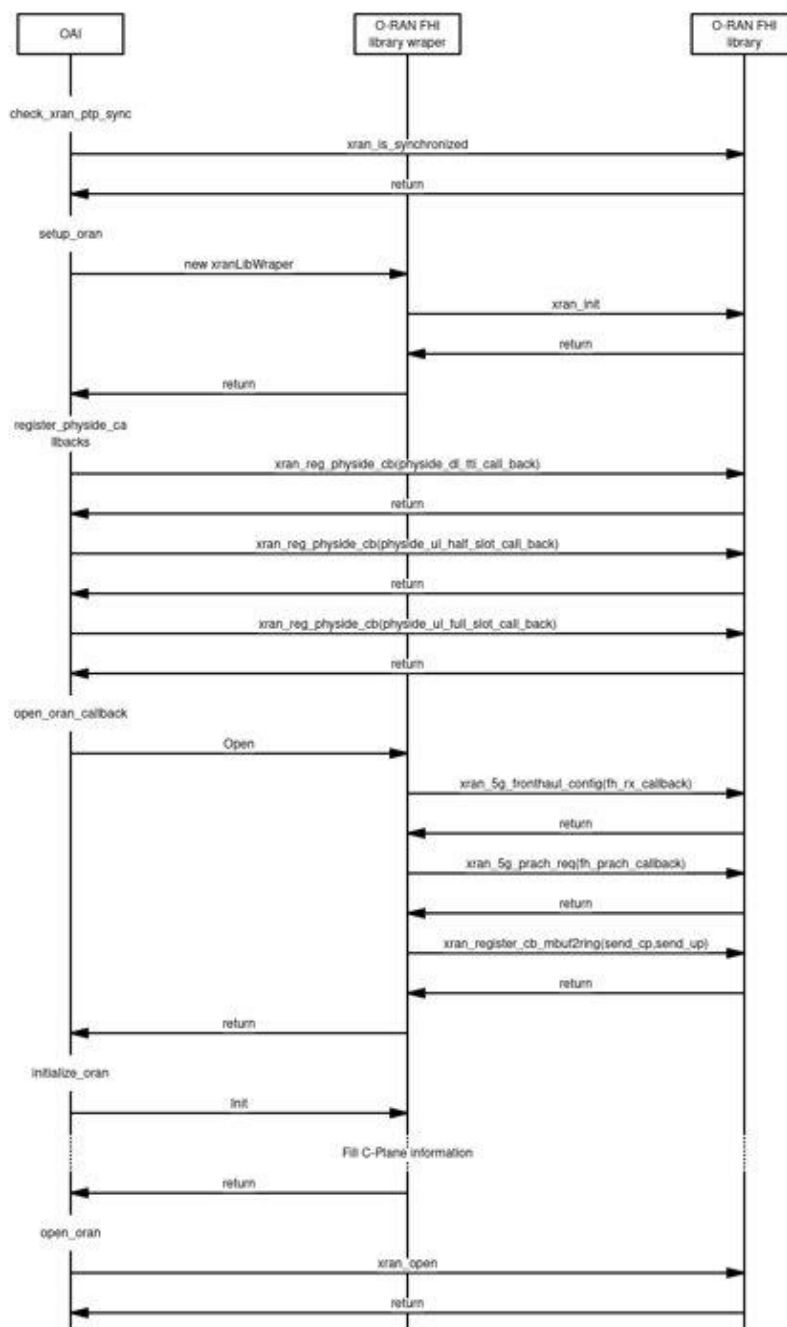


Figure 9: Message sequence chart in FHI integration

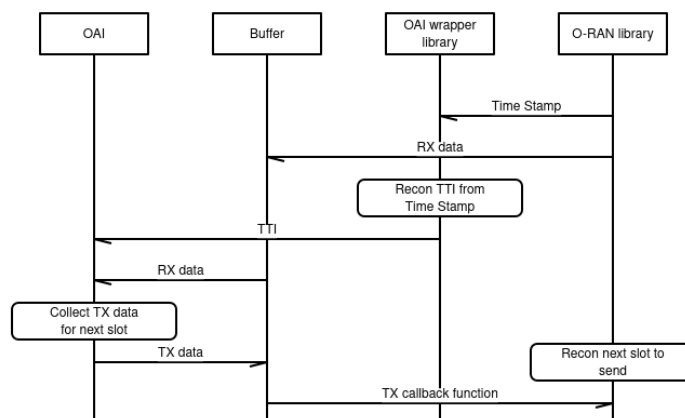


Figure 10: Message sequence chart for TX buffer

We successfully connected a COTS soft UE to the OAI-DU and STL RU both on the CP and UP. A sample 7.2 Wireshark traces corresponding to this is given below in Figure 11.

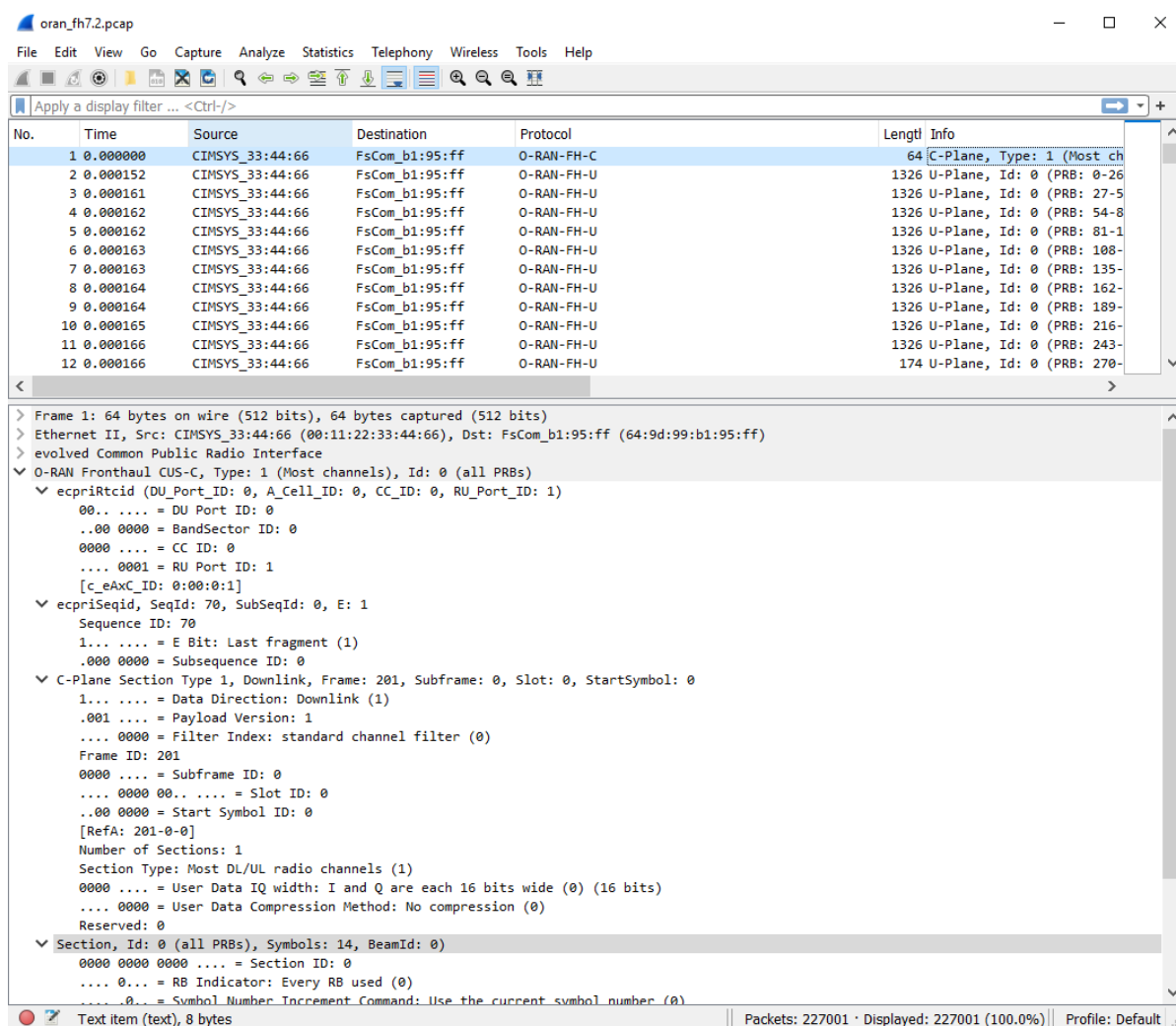


Figure 11: Sample 7.2 Wireshark traces

4.1.2 RunEL preparations for the RU-DU Integration over O-RAN 7.2 interface

RunEL upgraded and prepared for the Sparq-2025-ORU Radio Unit (RU) with 7.2 split PHY interface, for the integration effort with Eurecom DU. The RU is described during this section.

The Interfaces of the RU are depicted in Figure 12.



Figure 12: RU interfaces

The RU internal structure is described in Figure 13.

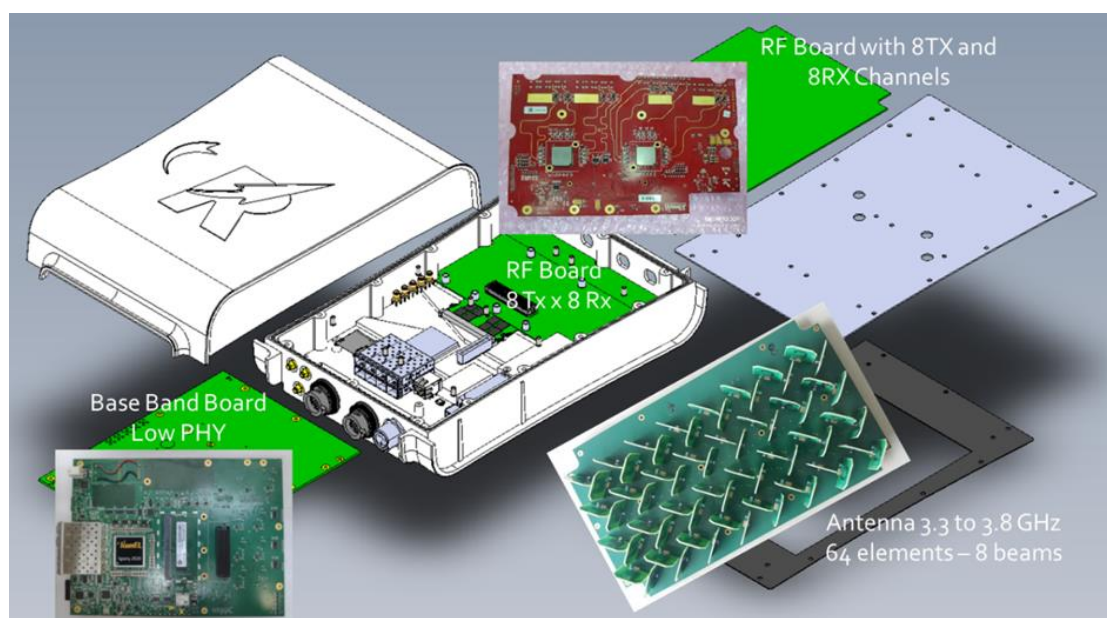


Figure 13: RU internal structure

The RU main features are:

- All-in-one integrated packaging of 5G RF and Baseband (Low PHY) components.
- Full compliance with 3GPP Release 15 Standard.
- Frequency Bands: 3.3GHz to 3.8GHz (n78 5G NR Frequency band).
- Supports MIMO 2x2 or MIMO 4x4.
- Beam Forming of up to 4 dual polarized beams.
- Antenna support - model dependent: either four external antennas or one beam forming internal antenna with 4 dual polarized beams.
- Support for internal GPS receiver for TDD synchronization.
- IEEE-1588 synchronization.

- Flexible coverage capabilities – greater coverage area or greater penetration capabilities.
- Small footprint, single-handed quick installation and simple provisioning
- Seamless and cost-effective integration with 5G DU with ORAN Interface (Option 7.2 Category B).

4.1.3 Issues encountered and further steps

Some issues were detected when performing the integration:

- At the beginning we uncovered an issue in the SRIOV drivers that are responsible for the virtualization of the LAN network (VLAN) in the DU stack. Precisely, the O-RAN Fronthaul libraries supported only the VLAN Ethernet transport while RunEL supported only VLAN UDP transport. RunEL reacted quickly in extending their transport support to ethernet.
- After successfully integrating the STL RU at Eurecom, we proceeded in reproducing that integration with the RunEL RU on the RunEL premises. That integration did not work because of the complexity of the FHI library and its dependencies, whose successful deployment lies on a BOM (hardware, software e.g., DPDK) and settings that are strict.
- Eurecom plans to send a complete server on RunEL premises to accelerate the integration with their RU.
- Both beneficiaries, RunEL and Eurecom, are committed to continue and complete the integration efforts even beyond the completion of the formal Affordable 5G two years period.

4.2 Telemetry Module Deployment in Far Edge Devices

As also described in D4.2 [1], the ML based telemetry module was ported, deployed, executed, and evaluated in the THI platform. THI platform is an FPGA prototype equipped with NEOX accelerator. 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. The goal of this test case 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). More details about the specific test case and the associated results are presented in Table 1.

Table 1: Telemetry integration in THI platform

Test case name	Telemetry integration in THI platform	Test Case id	Test-01-01
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	
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.	
Power - Performance Numbers in THI Prototype Platform		
Compression of AI Model	The ML based model is compressed by a factor of 4x compared to the initial size of the model. Only one compression technique was employed. The application of the second compression technique (low rank factorization) was not able to be employed without noticeable loss of accuracy.	
Parallelization degree	The telemetry module is parallelized in 32 threads and the workload was effectively split in a balanced way.	
Power consumption	The power consumption is less than 8mWatts (measured at gate level).	
Performance	The execution of each inference step is executed in less than 500ms. To achieve the reported execution times the following optimizations have been employed: SIMD processing of the layer level, kernel code optimizations, balanced thread execution, and scratchpad optimizations.	

4.3 O-RAN Fronthaul & S-Plane: ACC Labs

Integration testing in Castellolí identified a number of issues with the O-RAN Fronthaul & S-Plane implementation, causing packet loss and intermittent operation of the 5G RAN. While as much testing as possible was performed by 2 onsite integration visits (of ACC installation engineer, with support of CEL, NBC, ADV, ATOS, ATH, etc), while E2E 5G connectivity was

obtained, it was not stable enough and was noted that the 5G broadcast of the PLMN and consequent connection by UE was not reliable.

This was isolated to be fronthaul & S-Plane issues, initially thought to be jitter sensitivity between the 3rd party RU and DU, exacerbated by the use of a TSN fronthaul switch. P2P connection between the RU and DU using 10 Gbps fiber links improved the operation but there were still 'overflow' and 'underflow' errors latched in the status register of the RU's, indicating downlink packet issues.

Due to the logistical difficulties of testing onsite in Castellolí (infrequent onsite engineering support), it was decided to ship some of the equipment back to ACC labs, to further investigate, debug and resolve these issues.

The initial planned fronthaul, as described in Affordable5G D4.2 [1], is shown in Figure 14

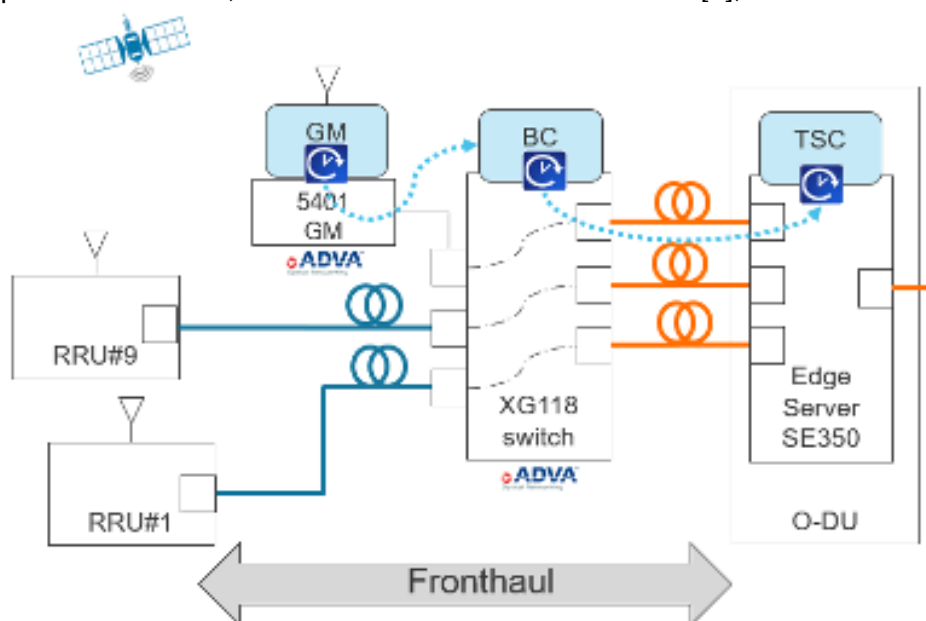


Figure 14: Affordable5G: Initially designed Castellolí fronthaul & S-Plane infrastructure

A simplified fronthaul & S-Plane was used as a baseline for testing in ACC labs, as shown below in Figure 15. This setup has currently provided the most stable results.

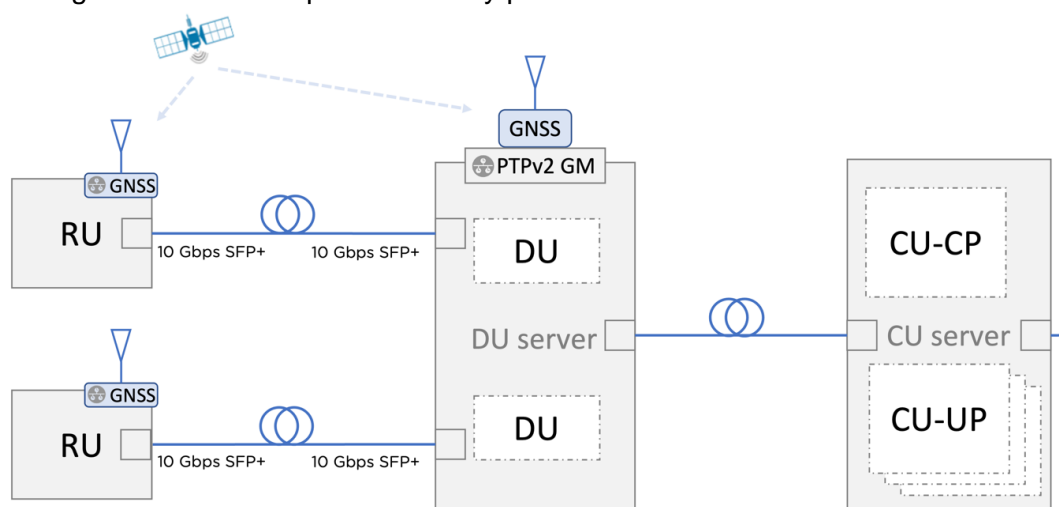


Figure 15: Simplified Affordable5G Castellolí fronthaul for minimized delay/jitter

For testing of fronthaul delay sensitivity, ADVA also shipped 2.5 km fiber spools to ACC, for insertion in the fronthaul links. It was concluded that the actual delay does not significantly impact RU/DU performance (even with the additional 5 μ s delay of the TSN switch), but rather it could be jitter sensitivity causing the issues.

The DU platform was also tuned to minimize jitter (in the BIOS, by switching to ‘performance’ rather than ‘power save’ mode, disabling hyper threading). However, there was very limited capability to characterize and measure the jitter (apart from the aforementioned RU ‘overflow’ and ‘underflow’ errors, which only indicate problems on the downlink). Issues caused by uplink jitter were poorly understood at that point. What was seen, was that this jitter has a direct impact on the 5G-NR RF spectrum, shown below in Figure 16, where the frequency is not stable but steadily increasing before falling back to original value.

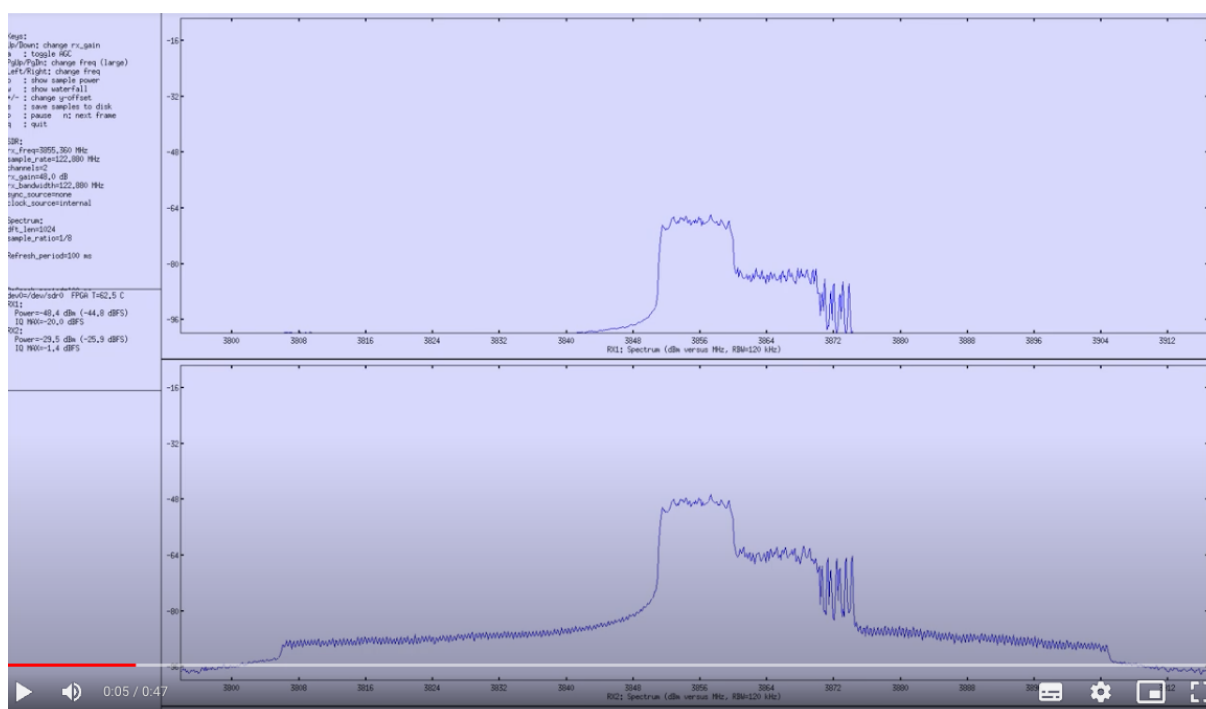


Figure 16: Spectrum analyzer output. Drift of frequency with TSN switch

While the intention, as indicated in Figure 14 and Figure 15, is to use the OSA 5401 PTPv2 Grand Master to time/frequency synchronize the DU & RU (O-RAN S-Plane LLS-C3), in the current setup, the DU slave to the RU uplink packets, with the RU being GNSS synchronized.

At the time of writing, these tests are ongoing, the progress and results of which will be presented in the Affordable5G final review. The currently planned tests are in 2 areas:

- A Netropy Network Emulator [3] has been received for fronthaul jitter emulation, which can be inserted in the middle of a 10 Gbps fiber link, to characterize the sensitivity of the DU/RU jitter, with & without a TSN switch, thereby improving troubleshooting of stability and look for system tuning to rectify. Initial testing has shown the fronthaul to be resilient to both delays and jitter and that the issue may be due to packet loss, which is known to negatively behaviour (for instance if the SSB -Synchronization Signal Block- are not repetitively sent to UE which sync's on these). This testing is ongoing.
- S-Plane DU/RU PTPv2 LLS-C3 time & frequency synchronization testing and debugging, in line with Section 4.2.4 of Affordable5G D4.2 [1].

Note that these RU/DU integrations in Castellolí were alternative 3rd party suppliers as the Affordable5G RU & DU, as described in 4.1, were not available for Castellolí. Fronthaul is a

high throughput and very real-time critical interface. It could be expected that similar issues would also be encountered in any subsequent system integration of those Affordable5G RU & DU network functions. This is especially so in a purely software-based implementation of the High PHY in DU and Low PHY in RU. It is anticipated that HW assisted and accelerated DU & RUs will perform better in this respect. It is noted that the alternative Intel FlexRAN implementation also encounters a number of similar issues and that there is some relevant configuration and tuning information in [4], that will also be considered in further testing.

5 MALAGA TESTBED

5.1 Final picture

The Affordable5G architecture has been instantiated in two platforms, Malaga and Castellolí. In order to remark the differences between platforms, final instances of both are presented in this deliverable. The final architecture of Malaga testbed is depicted in Figure 19, while Castellolí picture is included in section 6. The only modification from the architecture presented in D4.2 [1] is the use of Accelleran O-RAN instead of the O-RAN solution that was intended to be developed within the project (more information about why that solution is not ready can be found in previous section 4.1). Accelleran O-RAN is composed of 1 Benetel 550 indoor radio, depicted in Figure 17.



Figure 17: Benetel 550 radio unit

In addition, it includes a DU package which is composed out of 2 parts provided by Effnet and Phluido and which are integrated by Accelleran, and Accelleran CU and nRT RIC. All of them are physically installed in the server shown in Figure 18.



Figure 18: Physical server hosting O-RAN solution

Information regarding the rest of the building blocks can be found in previous deliverables.

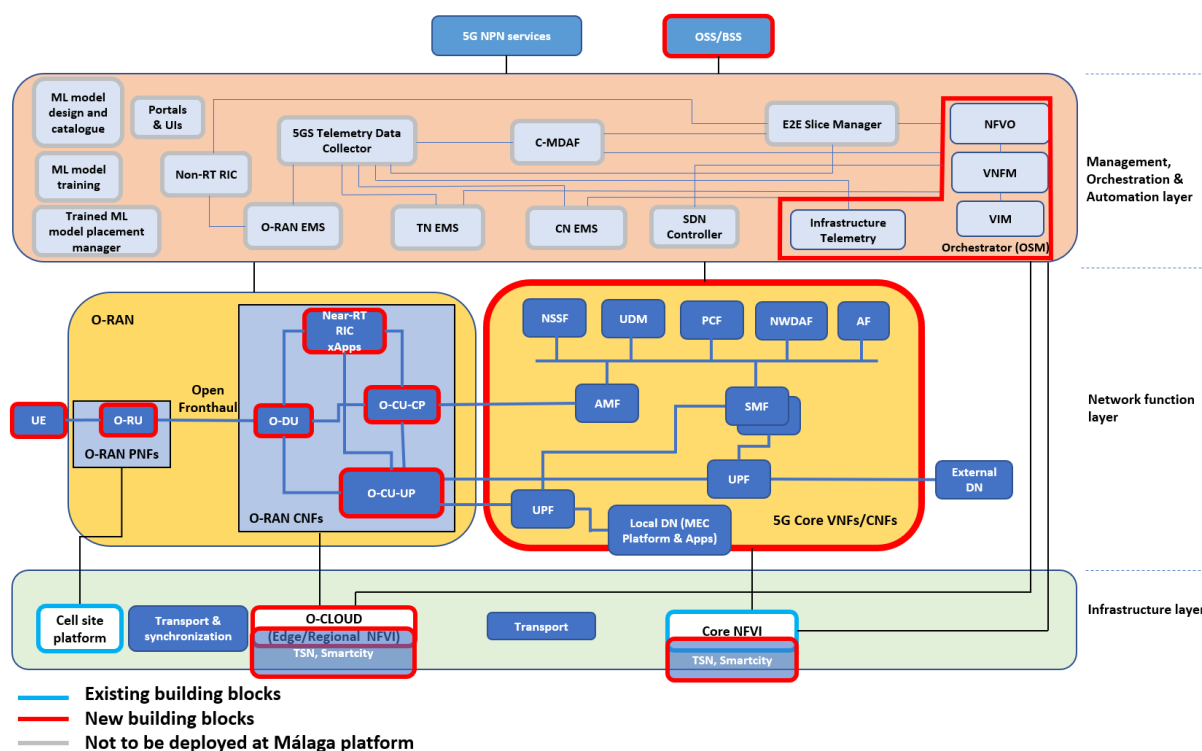


Figure 19: Final architecture of Malaga platform

5.2 Network characterization

Once the picture of the platform is clear, it is important to characterize the testbed itself. For that purpose, some Key Performance Indicator (KPIs) are defined and measured under normal conditions. These conditions are defined as: only 1 UE connected to the network, and default 5QI of 9. The KPIs to be measured are: E2E latency, E2E jitter, UDP DL/UL throughput and TCP DL/UL throughput.

To perform these E2E measurements, Telit fn980m model has been used as UE and a PC connected directly to the UPF is used as a server/client (depending on whether we are measuring UL or DL).

The results have been obtained from measurements during several hours and are all shown in the same format, presenting the probability distribution function (PDF) and the cumulative distribution function (CDF):

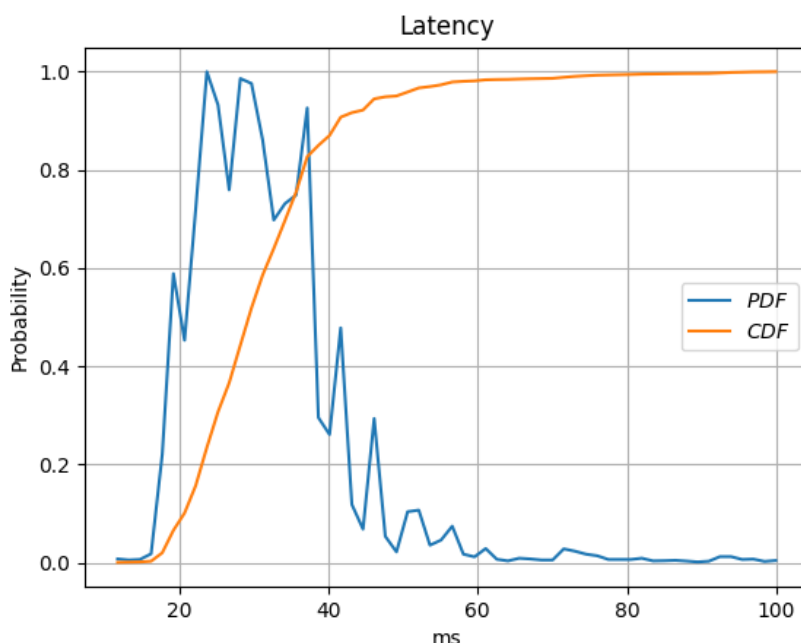


Figure 20: Malaga testbed baseline latency

In Figure 20 can be found the latency characterization of the network. From the PDF and the CDF can be extracted that the latency under normal conditions is between 20 ms and 40 ms. Specifically, the 80% of the distribution is < 35 ms. This shows a non-stable network that, for sure, will required some improvements in a scope beyond the project. This can be also double-checked with the Jitter graphic, shown in Figure 21.

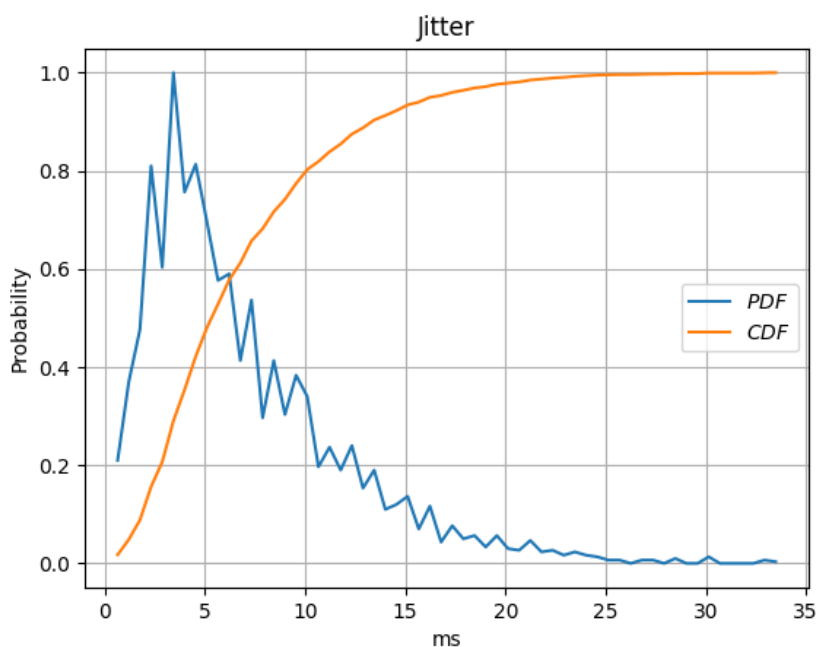


Figure 21: Malaga testbed baseline jitter

In addition, in order to characterize the network, it is important to define the Throughput that can be obtained, both downlink and uplink. We will also measure the difference between using UDP and TCP protocol.

DL Throughput

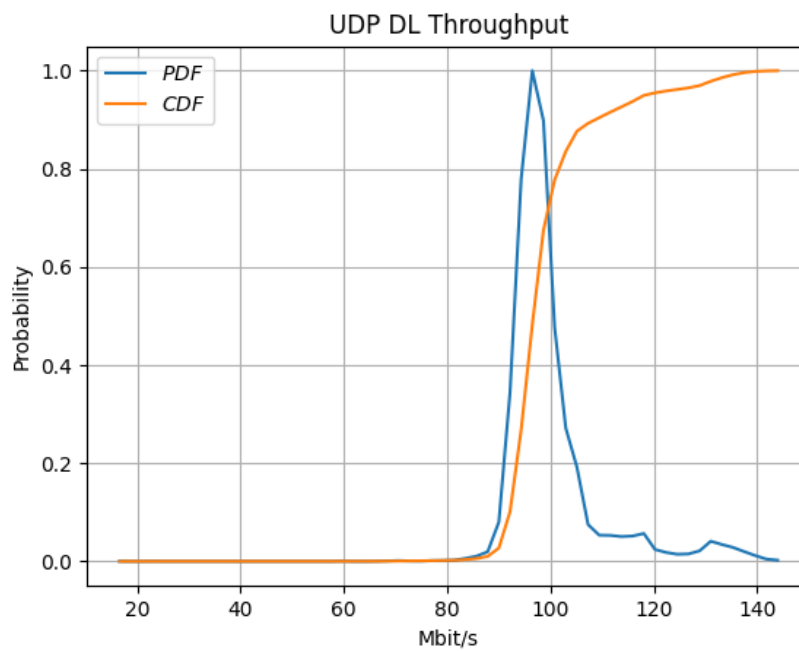


Figure 22: Malaga testbed UDP DL Throughput

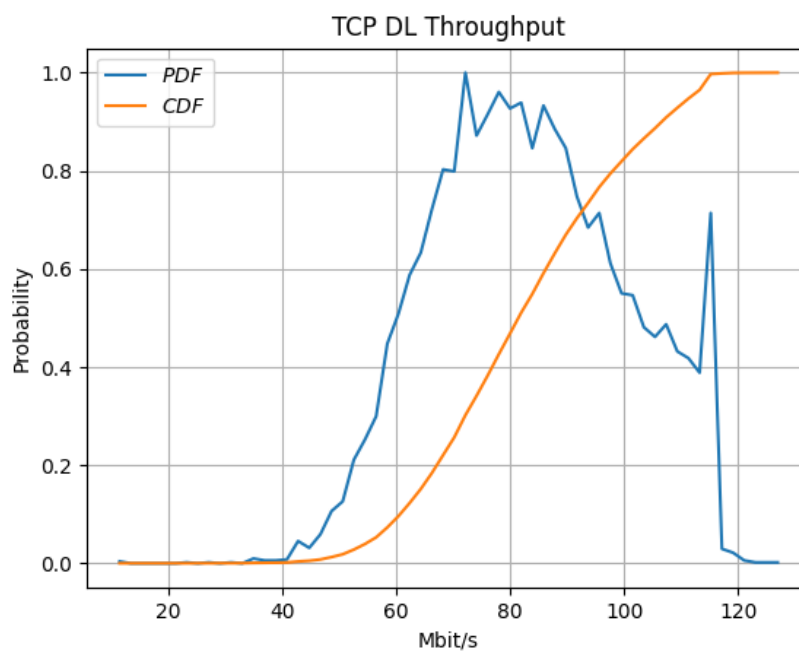


Figure 23: Malaga testbed TCP DL Throughput

In UDP, maximum throughput is 144 Mbit/s and mean throughput is 98.9 Mbit/s. On the other side, in TCP, maximum throughput is 127 Mbit/s and mean throughput is 82.3 Mbit/s. From

Figure 22 and Figure 23 we can conclude that the maximum DL throughput can be achieved with UDP, but the probability to reach such values is very low. However, with TCP we can achieve a relative peak of a bit below 120 Mbit/s, with a higher probability than in UDP. This difference may be due to the automatic windows negotiation in TCP that can adapt the traffic better to variant network conditions. We cannot forget that the environment in which these tests are performed is a dynamic one in which other networks could coexist.

UL Throughput

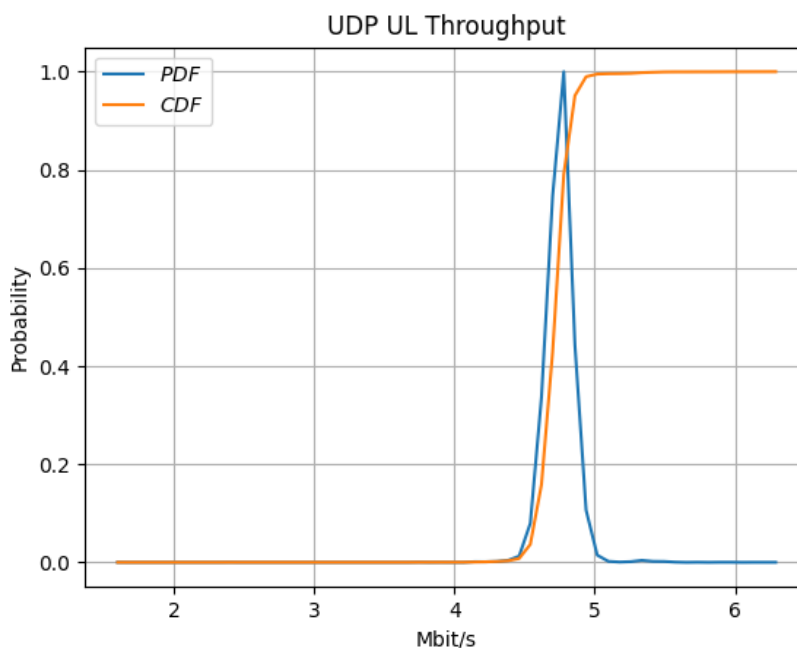


Figure 24: Malaga testbed UDP UL Throughput

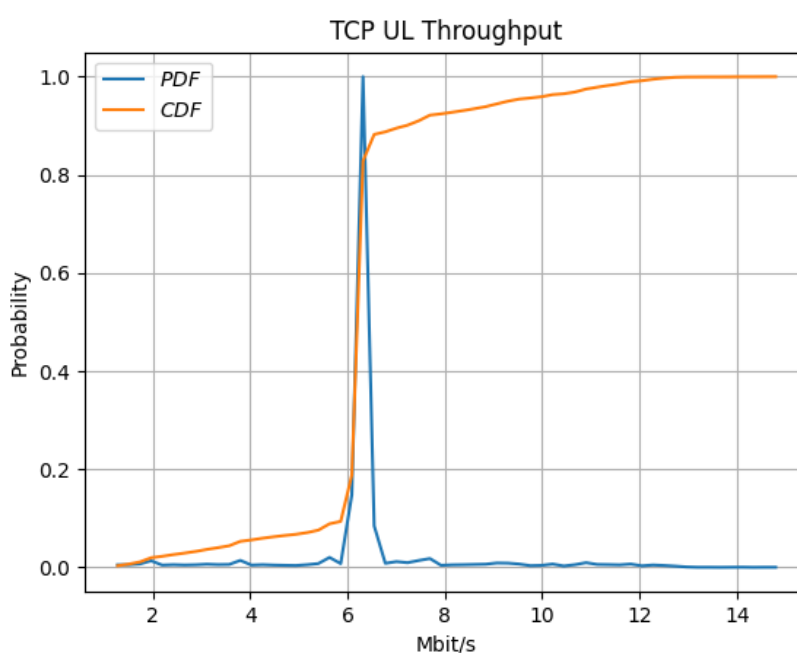


Figure 25: Malaga testbed TCP UL Throughput

Similarly, performing the tests in Uplink, we can obtain maximum throughput of 6.3 Mbit/s in UDP and 14.8 Mbit/s in TCP. In addition mean throughput is 4.7 Mbit/s in UDP and 6.3 Mbit/s in TCP.

5.3 TSN over 5G PoC

5.3.1 Building blocks

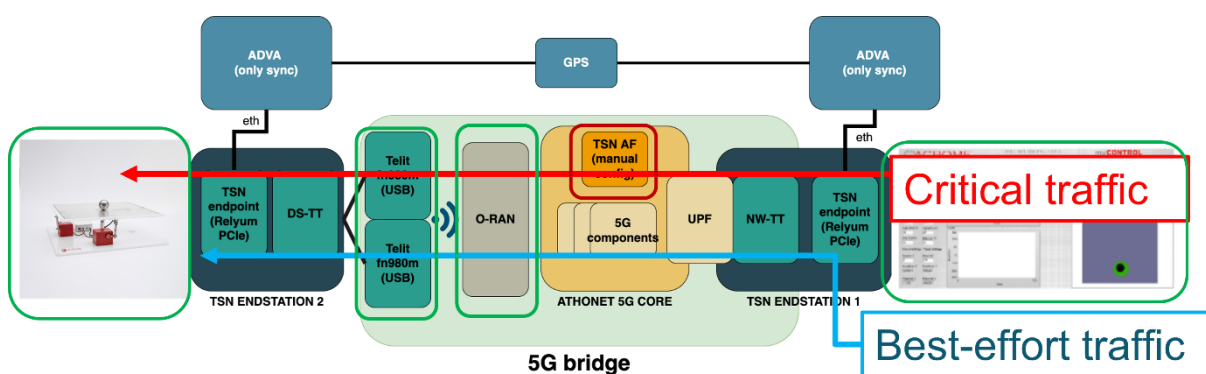


Figure 26: Setup of TSN over 5G PoC

The setup of TSN over 5G proof of concept is presented in Figure 26. It includes all the building blocks that can be separated in:

- Time awareness:
 - GPS
 - 2x ADVA FSP 150

The GPS is used as a unique reference clock for the two ADVA FSP 150, depicted in Figure 27. These devices are, in turn, used as master clocks for Device side and Network side of the solution, as it was explained in D4.2 [1].

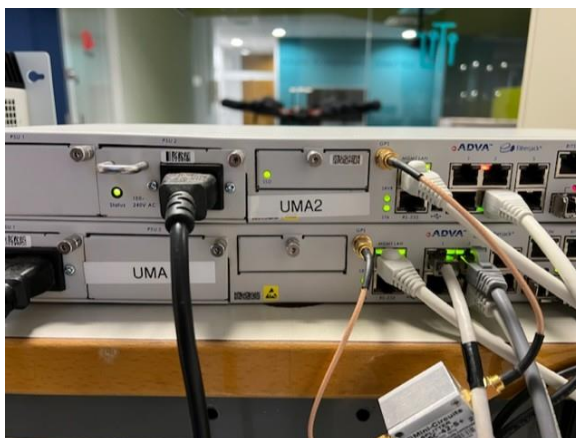


Figure 27: ADVA FSP 150 equipment x2

- 5G Network
 - ATH 5G core
 - O-RAN
 - 2x Telit fn980m UEs

The 5G network is the one defined in the Malaga architecture, composed of ATH 5G Core and the O-RAN solution provided by ACC. In addition, two Telit fn980m are used as UE, one for critical traffic and the other one for best-effort traffic.

- TSN end stations
 - DS-TT
 - NW-TT
 - 2x PCIe Relyum TSN cards

The TSN end stations are the two workstations presented in Figure 28. They both include a translator (DS-TT or NW-TT depending on the side) implemented using P4 language [5], and a PCIe Relyum TSN card to support TSN standards.



Figure 28: Workstations which run TSN translators

- Ball Balancing Table setup
 - 2x Raspberry Pi
 - PC Controller
 - Ball Balancing Table

Finally, the Ball Balancing Table setup, consisting of two Raspberry Pi (for touchscreen controlling and servomotors, respectively), one PC Controller on network side and the ball balancing table itself, which is presented in Figure 29.



Figure 29: Ball Balancing Table

5.3.2 Configuration and automation interfaces

All test cases for this PoC are end-to-end between both TSN end stations, just excluding the blocks outside TSN domain, that are the PC Controller and the BBT, which are not relevant for the measurements.

Thus, the interfaces that we are going to automatize and configure with OpenTAP are:

- In NW-TT, the interface connecting the output of end station 1 with UPF.
- In DS-TT, the interface connecting the output of TSN end station 2 with Telit fn980m UEs.

5.3.3 Configurable parameters and KPIs

The defined KPIs for the TSN over 5G PoC are E2E latency and E2E jitter. It is important to note that all these KPIs were already defined in the network characterization of the platform. However, in this case, the measurements are performed over the specific setup for this PoC. These are the most representative KPIs that, as a set, can define the behaviour of this developed solution.

Additionally, different configurations will be established to perform such measurements. The parameters that will be modified are: delay between packets, payload size, 5QI and network congestion scenario.

These parameters are defined as:

- Delay between packets (20 or 100 ms): Time delay between two consecutive sent packets.
- Payload size (100 or 1000 bytes): Packet size excluding headers.
- 5QI (5 or 9): 5G QoS Identifier is a pointer to a set of QoS characteristics.
- Network congestion scenario (0 or 1): Custom parameter defined as a combination of radio channel bandwidth and user's traffic. In our case, we use an additional UE in the network transmitting more traffic than supported by the network, according to network characterization. 0 = no additional UEs in the network. 1 = additional UE transmitting maximum throughput.

The following scenarios will be tested:

Scenario 1	
Parameter	Value
Delay between packets	100 ms
Payload size	100 bytes
5QI	9
Network congestion scenario	0
Scenario 2	
Parameter	Value
Delay between packets	100 ms
Payload size	100 bytes
5QI	9
Network congestion scenario	1

Scenario 3

Parameter	Value
Delay between packets	100 ms
Payload size	100 bytes
5QI	5
Network congestion scenario	0

Scenario 4

Parameter	Value
Delay between packets	100 ms
Payload size	100 bytes
5QI	5
Network congestion scenario	1

Scenario 5

Parameter	Value
Delay between packets	100 ms
Payload size	1000 bytes
5QI	5
Network congestion scenario	1


Scenario 6

Parameter	Value
Delay between packets	20 ms
Payload size	1000 bytes
5QI	5
Network congestion scenario	1

5.3.4 Test cases and results

As defined KPIs are E2E latency and E2E jitter, we are including in this section specific test cases to measure these indicators. Both test cases are performed using OpenTAP as automation tool to ease repeatability in the execution of a large number of iterations.

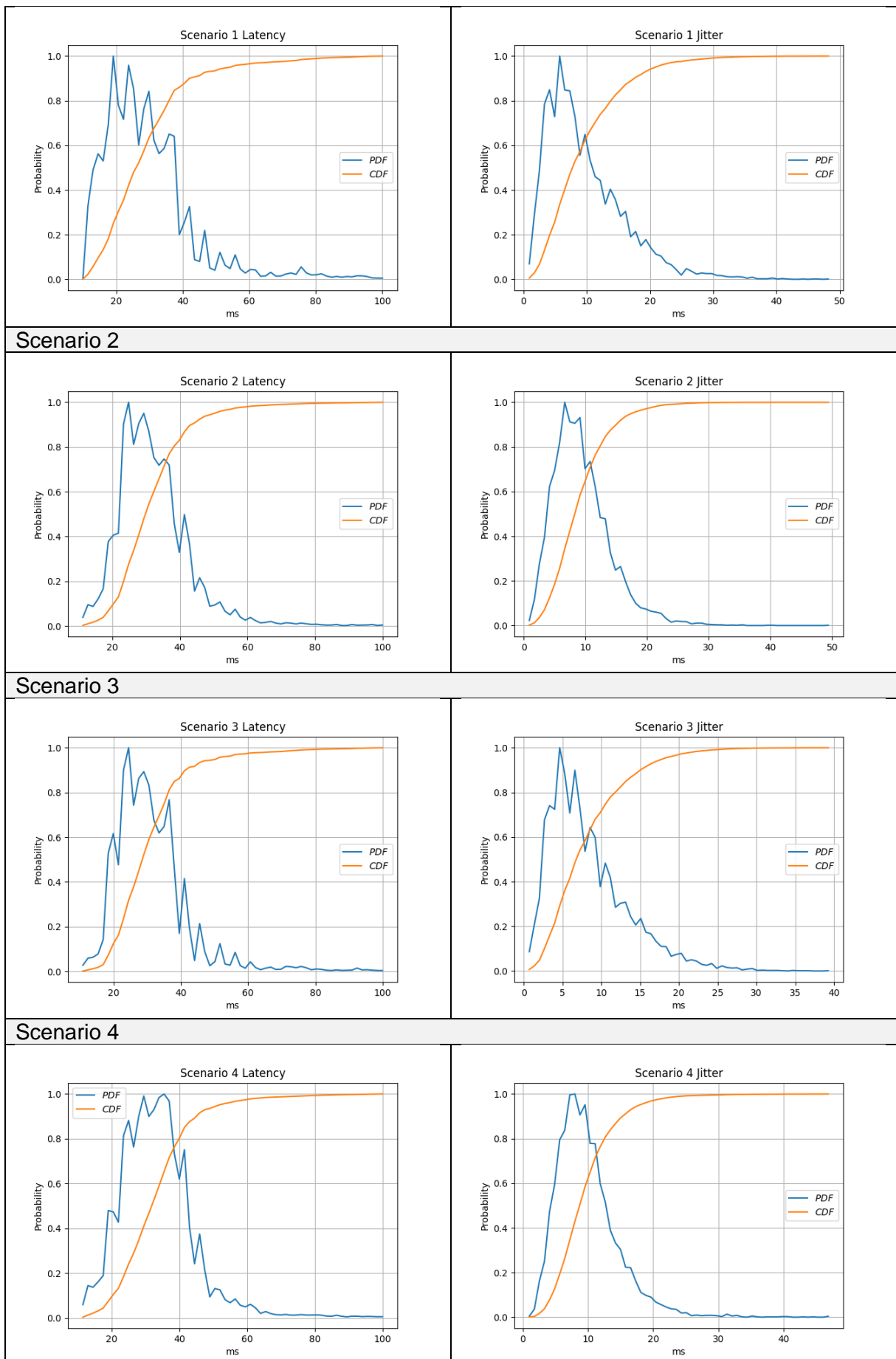
Test case name	TSN E2E latency	Test Case id	Test-02-01
Test purpose	Test end-to-end latency between two synchronized TSN end stations.		
Configuration	TSN end stations configured to be able to be synchronized using ADVA equipment.		
Test tool	fping, OpenTAP		
KPI	E2E latency		
Components Involvement	All building blocks listed above and related to TSN over 5G PoC.		
Pre-test conditions	TSN end points (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 fping and location to save the results on OpenTAP to send packets according to selected scenario, from endpoint 1 (network side) to endpoint 2 (device side), through the 5G network.	The tool is ready to send traffic.
	Step 4	Start Test Plan on OpenTAP to start sending data.	fping is running with selected scenario configuration and results are being saved.
	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	Not defined, it is a comparison between all scenarios.		
Additional Resources			

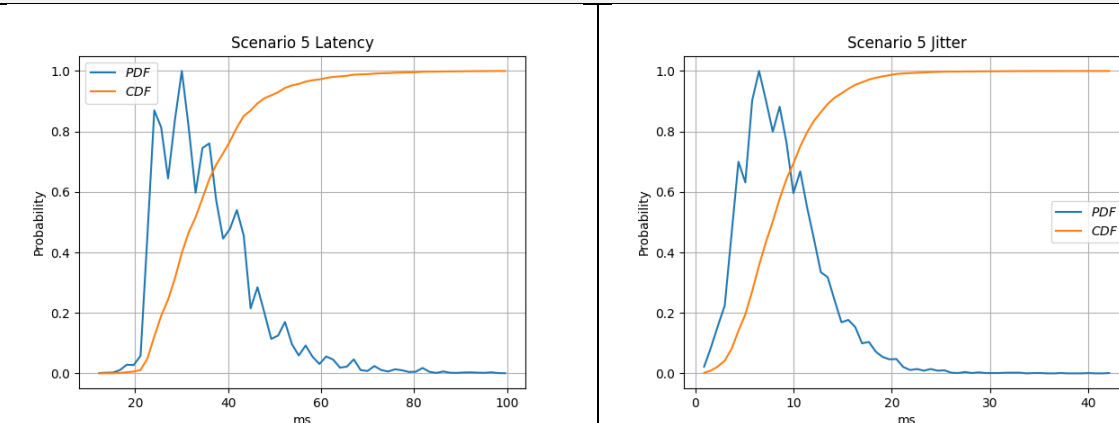
Test case name	TSN E2E jitter	Test Case id	Test-02-02
Test purpose	Test end-to-end jitter between two synchronized TSN end stations.		
Configuration	TSN end stations configured to be able to be synchronized using ADVA equipment.		
Test tool	fping, python program to calculate jitter.		
KPI	E2E jitter.		
Components Involvement	All 5G components, NW-TT, DS-TT, ADVA FSP150 x2, TSN endpoints and Telit fn980m		
Pre-test conditions	Test-02-01 executed		
Test sequence	Step 1	Use the .csv files obtained in Test-02-01	
	Step 2	Process the file with python to obtain jitter.	Jitter results are obtained.
Test Verdict	It is also a comparison between all scenarios.		
Additional Resources	None		

Results

Latency	Jitter
Scenario 1	



Scenario 5



Scenario 6

Scenario 6 results have not been included as they are not representative. Since delay between packets is established to 20 ms in such scenario, and this is quite below the mean latency of the network, the result shows more than 60% of packet loss, and the PDF and CDF associated graphics are biased.

From the obtained results, it can be concluded what it was already exposed in section 5.2: the network is unstable and, even with the synchronization in the TSN domain, the effects of 5G network latency and jitter are notable. However, even concluding that with these conditions it is difficult to see the benefits of TSN synchronization and prioritization over 5G, we can still observe some improvements.

Setting the focus on latency graphics, we can see a common behaviour in scenarios 1, 3 and even 4: The percentage of latency between 20 ms is higher in such scenarios, which can be seen in the relative maximum peaks in the PDF function. This is even higher in scenarios 1 and 3, which fits with the configurations, as these are the scenarios without network congestion. In addition, scenario 4 shows a similar behaviour, but a bit worse, as this scenario includes network congestion. However, the use of 5QI of 5, which gives priority to the critical traffic we are sending, is improving the results. Finally, scenarios 2 and 5 present the lower percentage of latency between 20 ms, which is also representative of tested configurations. In both, 5G network is saturated and, in scenario 2, the 5QI used is the default one, so the traffic is not treated as critical. On the other side, in scenario 5, the payload size has been increased to 1000 bytes, what has turned out to be another cause of latency deterioration.

In addition, from the results we can also extract that TSN mechanisms cannot mitigate the inherent jitter coming from the 5G network, which in a real TSN domain should be ideally 0 ms.

In conclusion, network instability has been proven to be a challenge in order to show TSN over 5G results in this proof of concept. However, during the tests has been noticed that, when network is stable, better results can be obtained thanks to TSN synchronization and prioritization over 5G.

5.3.5 Comparison with Nokia RAN

As explained in previous section, the specific 5G network used during the tests has been a critical part in this PoC. Thus, as commented in section 2.1, some tests will be performed replacing O-RAN solution by Nokia RAN. In particular, latency and jitter tests are presented in Figure 30 and Figure 31, respectively.

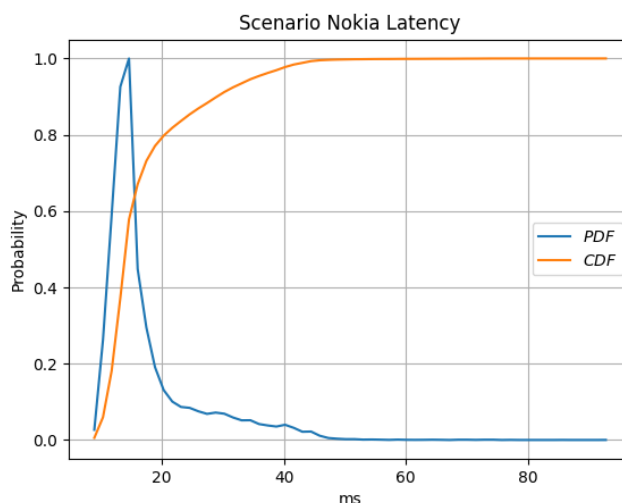


Figure 30: Network latency using Nokia RAN

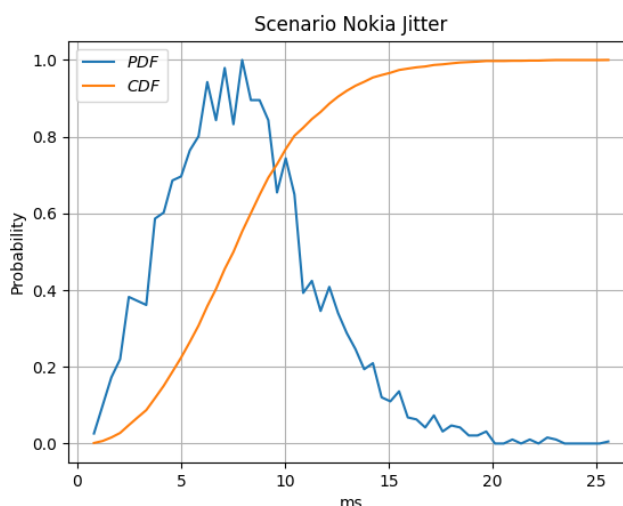


Figure 31: Network jitter using Nokia RAN

First, it is important to note that this test can be compared with Scenario 4, in which the network is saturated (in this case during a big event in which a lot of people was connected to the network) and we are using a prioritized profile with 5QI of 5. From the results, we can see an improvement, mainly in latency, but also in jitter, as the PDF function is very narrow and concentrated below 20 ms, in comparison with previous tests. About jitter, it is also lower than in previous tests, but still far from the goal of a TSN over 5G real solution. This is mainly due to the synchronization over 5G, that cannot be achieved yet, as explained in D4.2 [1]. Thus, we can conclude that the limitations on latency on previous tests were mainly due to radio performance.

5.3.6 Innovations and conclusions

This proof of concept has allowed us to prove some innovations:

In terms of software development, TSN translators have been designed using P4 language based on software switches. This is an innovative action since it allows us to define and modify dynamically custom TSN headers and include priority on the traffic at the same time. This innovation allows us to integrate the 5G network as a bridge in the whole TSN environment.

Another innovation comes regarding time synchronization. The innovative solution which is proposed is based on two TSN master clocks (one on device side and another on network side), and both are using the same GPS signal as a reference. Thus, after validating it with the oscilloscope, we can assure that the synchronization accuracy is valid for this PoC.

In conclusion, this PoC has been useful to demonstrate the benefits of TSN mechanisms applied to a real 5G network. Even with the presence of instability in the network, improvement in latency and jitter have been achieved. This has been obtained thanks to development of the TSN translator, synchronization on both side of the TSN network based on ADVA equipment, and the prioritization over 5G.

5.4 Smart City Pilot

The Computer Vision Analysis for Emergencies (CVAE) service relies on multiple components and logical functions installed in the edge and the network's core. The CCTV cameras deployed at the edge play an essential role as they represent the main input of data from which the CVAE service will operate. As described in previous deliverables, one of the main objectives of the CVAE service is to provide real-time image analysis to detect emergency events.

With this in mind, the Smart City Pilot intends to demonstrate the usage of a 5G private network, and the advances provided by Affordable5G in an emergency scenario occurring at a supermarket. The addressed scenario will be hosted in-door, assuming the loss of a child in a supermarket. As soon as the parent identifies the need to detect his child, the OSM (orchestrator) module from Affordable5G will deploy the detecting system on the edge layer of the 5G network to analyse the overall supermarket's video streaming. The data transmitted over the Radio Area Network (RAN) to the Internet flows through a User Plane Function (UPF) deployed at the edge of the network. Quickly and accurately routing packets to the correct destination on the Internet, the UPF will address the need for the security man to start streaming information over his mobile phone. After the request, the 5G Core and the O-RAN modules will handle a prioritisation feature to give the higher bandwidth possible to the security phone so that the video streaming can be covered over the 5G network. After detecting the lost child, the security guard will be able to identify on his mobile phone that we have found the kid and the OSM will start the necessary services to notify the parents.

As also described in D4.2 [1], a part of the CVAE services will be also demonstrated in the FPGA platform provided by THI. In particular, the people detection algorithm is deployed in the Zynq platform in which a multithreaded instance of the NEOX accelerator will be realized. Apart from the hardware IP, THI is also utilizing 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.

5.4.1 Building blocks

As previously described in the update pilot definition the present pilot is divided into two different services having each one its own innovative modules. The development followed the best practices to avoid the monolithic structure in order to facilitate the deployment of both services.

• Static Service

The static service was expected to be available through two types of computer systems: a BullSequana server capable of performing person detection in multiple video streams provided by connected edge nodes, and a Jetson Nano which performs the role of an edge node, that also perform person detections in a given video stream obtained from a connected camera.

Unfortunately, due to the Jetson Nano architecture, some required packages for the proper execution were not able to be installed in the integration phase, and to circumvent the issue both of the ML algorithms were deployed in the BullSequana. Being the current deployment shown in Figure 32.

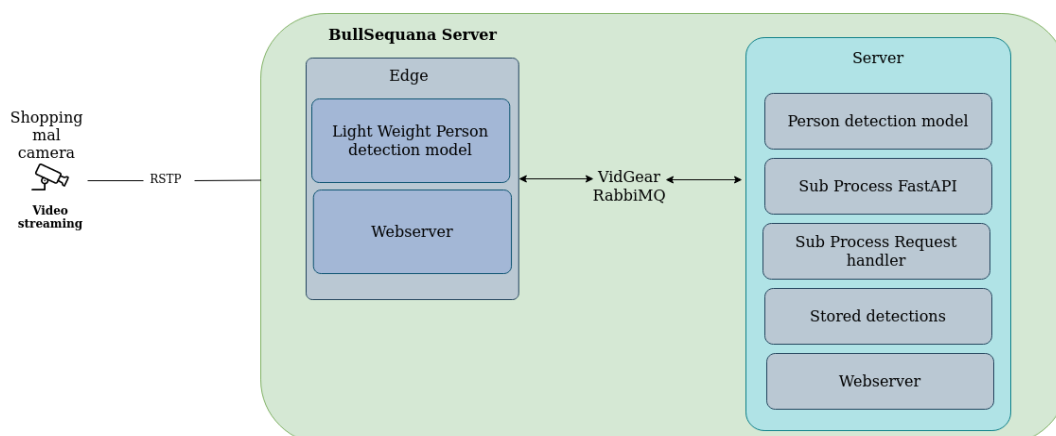


Figure 32: Static Service current deployment

The static service building block has the following functionalities:

- Edge web server: responsible for the communication/synchronization, for when a detection occurs. Sending and receiving the information necessary to transmit the live stream to the Server.
- Light weight Person detection model: Performs the detection of person on the RSTP feed.
- Person detection model: performs the detection and saves the captures to a given directory.
- SubprocessFastAPI: Provides an API for requesting camera streams and the saved videos.
- Webserver (for the Server): Provides a web interface for viewing the streams made available by the FastAPI.

It's referred to as a static service since it doesn't take advantage of the 5G capabilities being the connectivity between the different parts assured by the Wifi. The Edge 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 edge the video stream is re-transmitted to the Server, where a more capable and robust machine learning algorithm processes the stream. On the server a web server is hosted that displays to a user of the video stream with the ongoing detections. These detections were previously expected to be stored in a Wasabi¹ bucket, which is a cloud storage service that provides a high-performance, reliable, and secure data storage

¹ https://wasabi.com/wp-content/themes/wasabi/docs/User_Guide/topics/Creating_a_Bucket.htm

infrastructure. However, the storage of video files in the cloud is not the best practice and to safeguard any ethical question on where the storage of the detections are made, it was decided that all the detections will be stored in the BullSequana.

The ML algorithm running in the Server is based on DeepSORT and YOLOv4 [10], which enables the detection and tracking of individuals within the camera's field of view. The overall workflow of this system is illustrated in Figure 33.

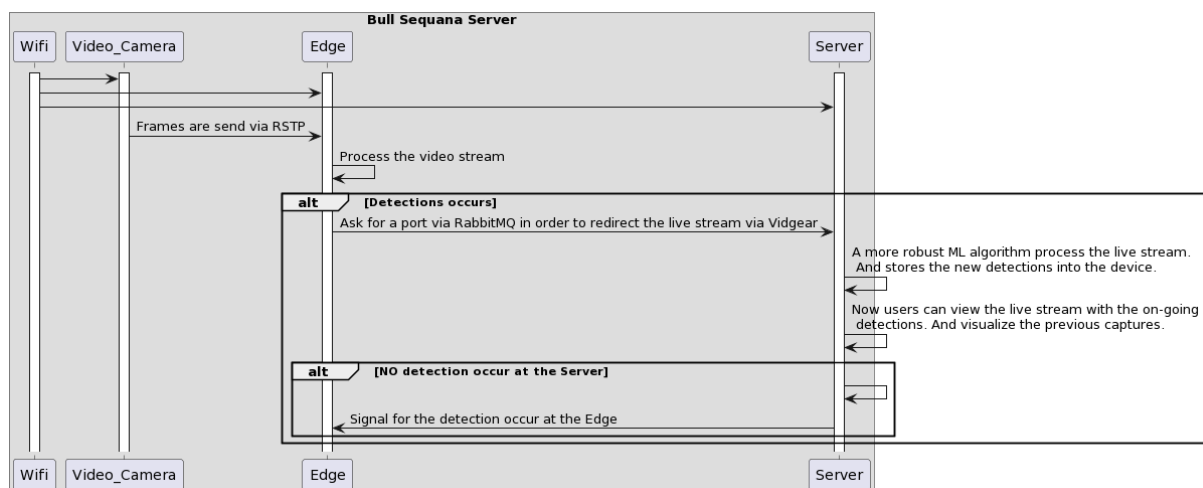


Figure 33: Overall static service workflow

• Dynamic Service

For the building blocks of the dynamic service, the parent of the lost children will report the missing child information in a shopping mall to a Security Guard, by providing a picture and the parents contact information through a dedicated API. This will trigger to perform a request to the Affordable5G's OSM (orchestrator) module, which will deploy the Service_X, located at the edge layer of the 5G private network. This service will propagate the received information to the services instantiated in the BullSequana Server.

While the Service_X is starting, a traffic prioritization of the Security Guard Smartphone will be provided. This action is performed by the 5GC at the Unified Data Management (UDM), Policy Control Function (PCF), Session Management Function (SMF), and UPF level, modifying the dedicated QoS through a 5G QoS Identifier (5QI) selection. Changing the 5QI value allows to guarantee higher priority to the traffic of a specific user profile within the same slice. In this case, the profile of the security guard is prioritized over citizen users with normal *best-effort* service.

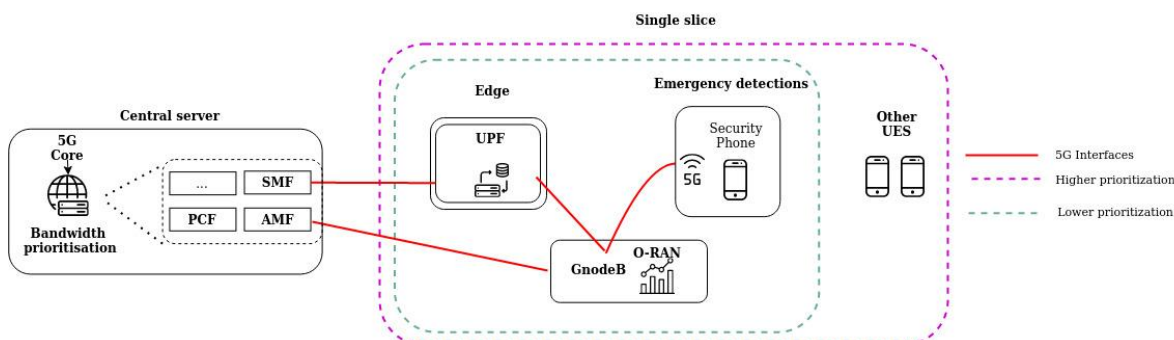


Figure 34: Prioritization over 5G network in Missing Child scenario

Once prioritization is set, 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 without any impact for the rest of the network. The high-level architecture is depicted in Figure 35.

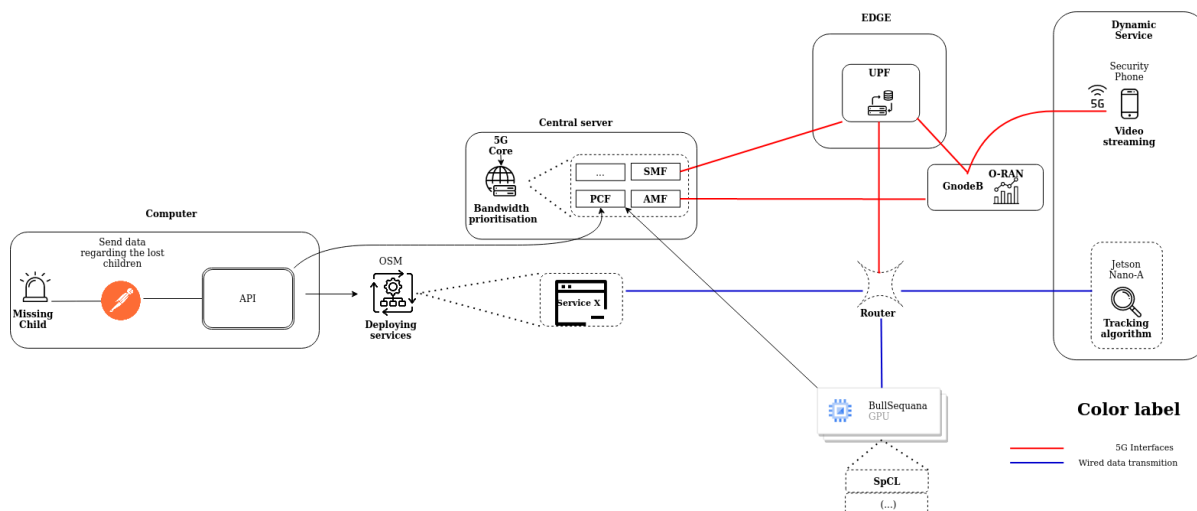


Figure 35: High-level architecture of Missing Child scenario

While a high-level dataflow for this service is represented in Figure 36.

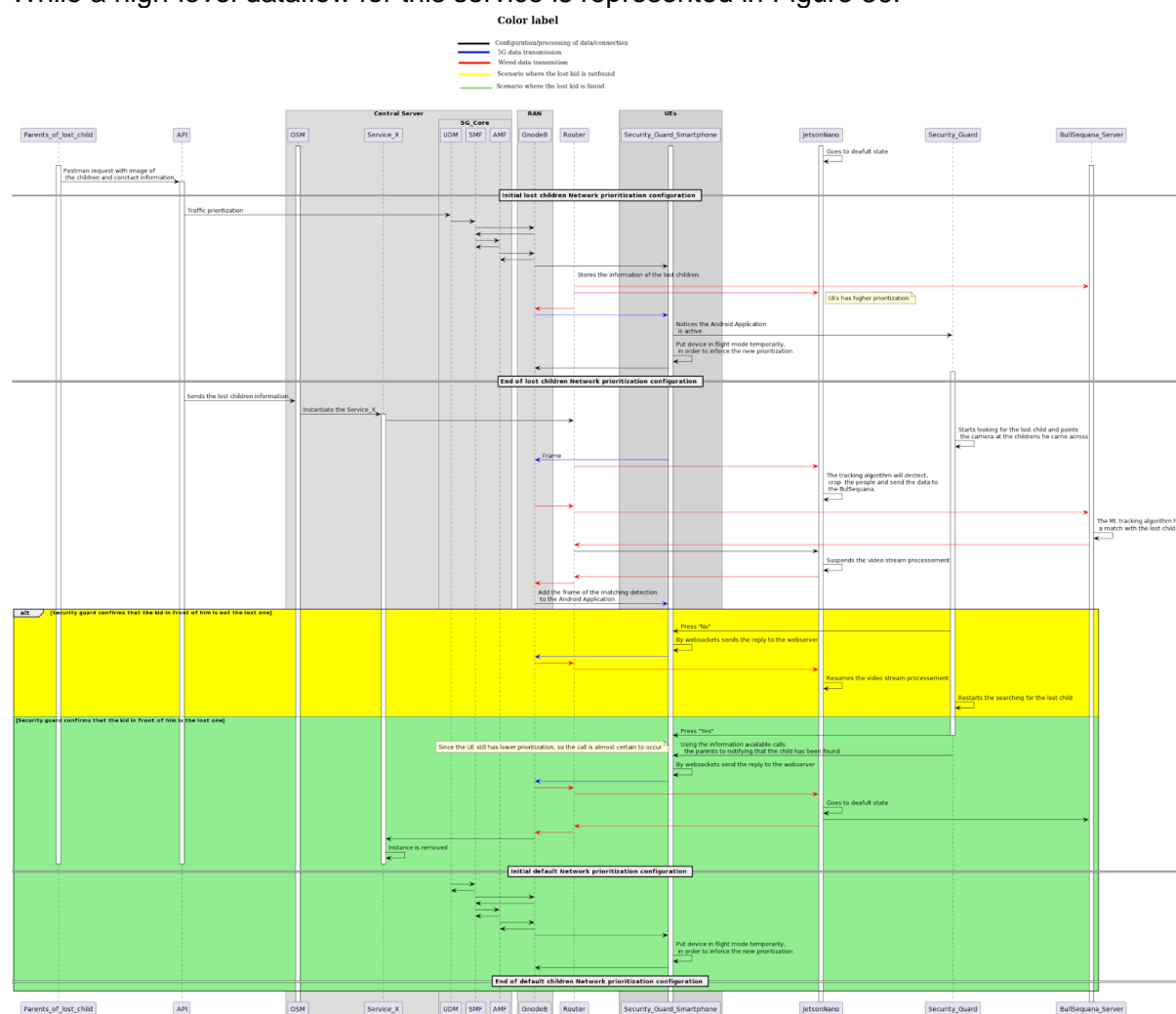


Figure 36: High-level dataflow of Missing Child scenario

The Dynamic Service is comprised of three main operational blocks, as illustrated in Figure 37.

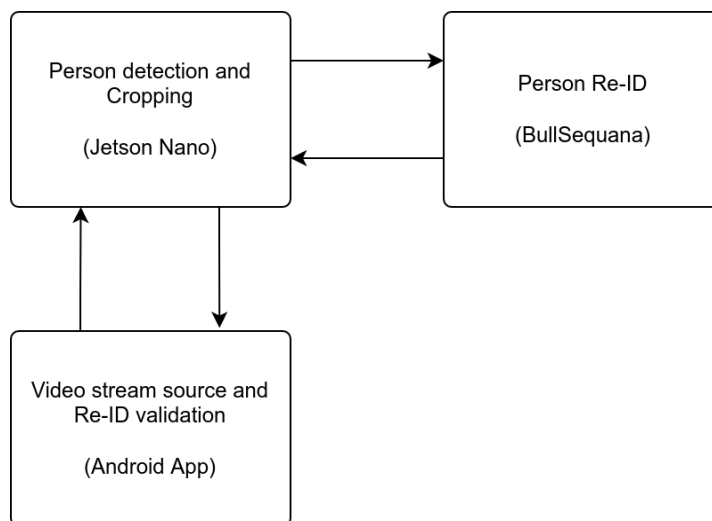


Figure 37: Dynamic service building blocks

- **Android App:** has access to the phone's camera and streams the frames to the Jetson Nano via a websocket connection. When a match occurs in BullSequana, a message with the match information is sent through the Jetson Nano and back to the Android App in order for the security guard to validate it.
- **Jetson Nano:** performs person detection on the received camera frames, and crops and resizes the detections to be used by the re-identification algorithm. After processing the detections, the cropped frames are sent to a Redis stream on BullSequana in order to be read by the process running SPCL.
- **BullSequana:** has an API for receiving requests from Service X to search for a missing child and cancelling a search. Given the missing child's picture, it performs person re-identification using images in the gallery (received via the redis stream) in order to find the missing child.

A more detailed view of the overall system is shown in Figure 38.

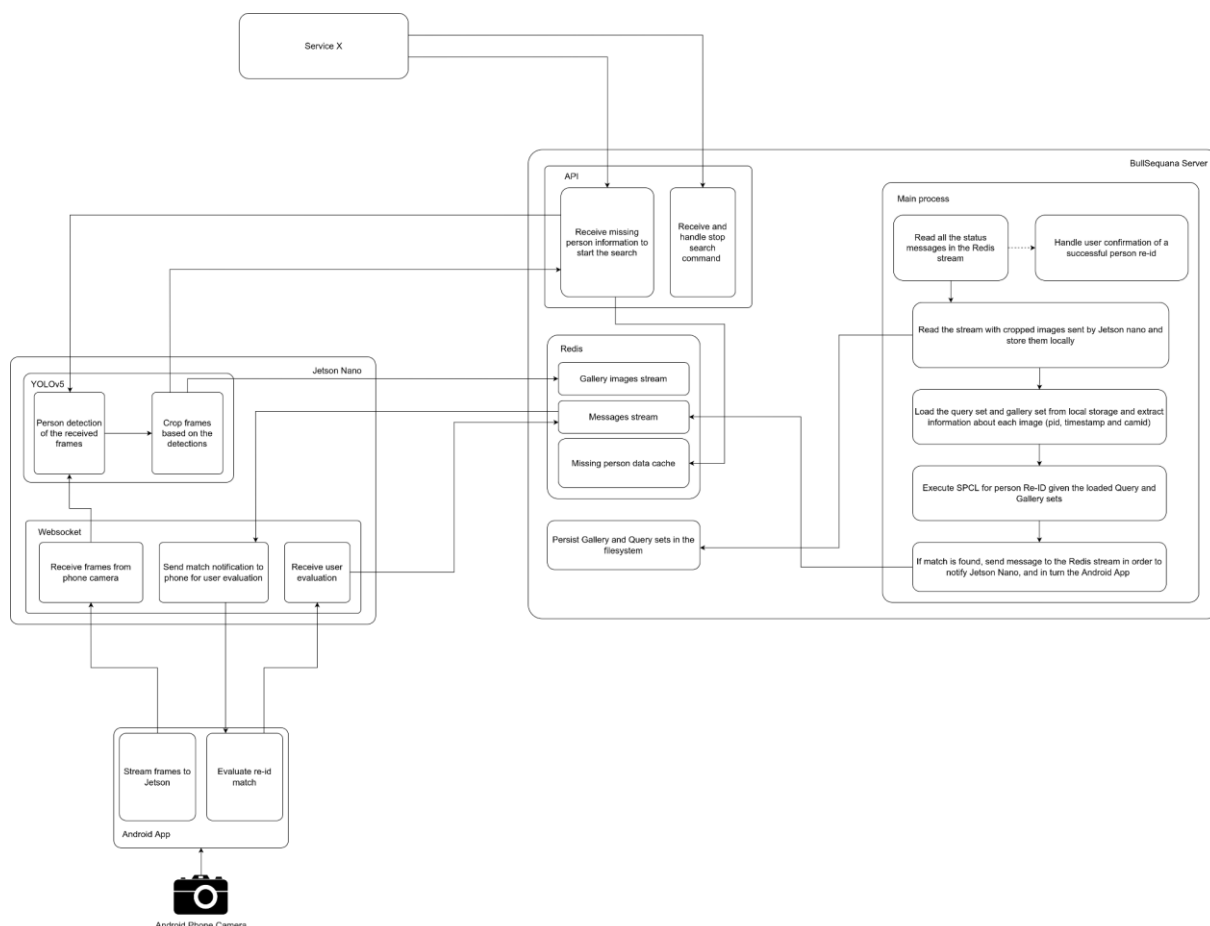


Figure 38: Detailed view of overall dynamic service system

Android App

The Android application is the source of the stream that is sent to and analyzed by the Re-Identification algorithm.

It is divided into 3 sections, one for each provided feature:

- Camera tab for displaying the frame streamed to the Jetson Nano via a websocket connection
- Detection tab to confirm or deny a match found by the SPCL algorithm
- Settings tab for configuring the API server's IP address and port, the camera's (phone) id, and adjusting the stream's framerate. After the child is found, this page will also show the phone number of the parent

The operation flow takes the following steps:

1. The user inputs the API's address and port on the Settings tab, along with the desired frame rate for the stream. After this initial configuration, the user clicks the *Connect* button to establish a websocket connection with the Jetson Nano.
2. The user switches to the Camera tab and points the phone's camera to the surrounding area looking for the missing child.
3. After a match is made by SPCL, deployed in Bull Sequana, the user receives the match information through the websocket connection with Jetson Nano and has to approve or deny the match.

4. In the case of an approved match, the search stops and the user goes back to the Settings tab, where it will be shown the phone number of the child's parent. On the contrary, if the match is not approved, the app automatically switches to the camera tab and the streaming resumes, along with the search.

Jetson Nano

The API is deployed in the Jetson Nano and acts as the bridge between the client and the Re-Identification algorithm. It provides a websocket endpoint for establishing a communication channel with the client and sends cropped person detections and status messages to redis streams, which are consumed by the Re-Id process in Bull Sequana.

The API makes use of the YOLOv5 architecture to perform person detection on the frames that arrive from the Android application, crops the detections according to their bounding boxes and sends the results to the redis stream used for SPCL gallery images.

The API sends matches originated from Bull Sequana to the Android App through the websocket and relays the user evaluation of each match to Bull Sequana.

Bull Sequana

The Re-Identification process is deployed in Bull Sequana, along with an API for controlling the search state, used by Service X, and a Redis instance to manage Redis streams and cache the information about the missing child.

The API has two endpoints:

- one for receiving the missing child's information.
 - It caches the child's data in Redis and calls an endpoint of the Jetson API to perform person detection. The endpoint processes the detection by cropping the bounding box around the child and resizing it to be usable by the SPCL algorithm, which then responds with the processed image. After receiving the processed image, the endpoint stores it in the directory corresponding to the query set, that may contain one or more images of the missing child.
- one for stopping the current search operation.

The main process of the BullSequana server executes the SPCL algorithm according to the following sequence of events:

1. Checks for messages in the appropriate redis stream in order to process service status information. Status information may be a notification that the missing person was found, given the confirmation of the match by the user, or the stop command requested by Service X.
2. Checks the gallery stream for new cropped images obtained from person detection on the Jetson Nano. If present, it stores the images locally in the Gallery directory along with the previously obtained ones.
3. Loads the query and gallery sets from their respective local directories and extracts information about each image. Each image will be associated with a person id, timestamp and a camera id.
4. The SPCL algorithm is executed, taking into account the loaded Gallery and Query sets, in order to find matches with the missing child.
5. If a match is found, a message is sent to the redis stream used for notifications, which is then read by Jetson Nano and sent to the user for evaluation.
6. If the match is approved by the user, a message is sent through the websocket to the Jetson Nano, and then sent to the redis stream used for notifications. When the main process checks the redis stream in the following iterations, it verifies that the child was

found. This prompts it to stop the search and to send a message to service X notifying the outcome.

5.4.2 Configuration and automation interfaces

For the present pilot, most of the used infrastructure and network capabilities were deployed and configured by the partners as described during this section.

The tests for Smart City were performed directly in the Jetson Nano, BullSequana and in the OSM by the Martel team. Without the need to recover to any specific software to analyse and process the data.

5.4.3 Configurable parameters and KPIs

The deployment of the Smart Cities pilot was tested in the UMA infrastructure and to evaluate the performance of the solutions, several tests with distinct configurations were defined.

It's important to state that any analysed metrics of a given ML algorithm is influenced by several factors, for example from the computational power of the device on which it is deployed. To the quality and quantity of the data processed. Is expected a decrease in performance when several detections occur simultaneously, since more resources are being used.

Some tests are transversal for both services, like the impact of the usage of CPU/GPU in the processing capabilities of the ML algorithms. This test enables the analysis of the inference time that the different services will have in the processing of a single frame.

Additionally, for the dynamic service, the following KPIs were considered:

- Global inference time:
 - Person detection and cropping average processing time.
 - The transfer time from the API to the Redis stream was at first considered as a KPI, but later we concluded that it didn't have a major impact to the performance due to the fact that the cropped images received by the Redis Stream are not immediately processed by the backend and the SPCL algorithm. Only when a person is currently being searched and the SPCL algorithm finishes analysing the current gallery directory, then the newly received images cached in the Redis Stream are stored in the gallery directory and are available to be used in the next SPCL search iteration.
- Average precision (AP) which evaluates the object detection model capabilities of a given ML algorithm.
- Average reconciliation time for the Service X, how long it takes for a change in the remote repository to be enforced in the deployed service.
- Bit rate, the quantity of data being transferred from one part of the network to the other in a certain amount of time.

For the Service X, the Reconciliation time, which is the amount of time necessary for a change in a remote repository to be enforced in the current instantiation, was considered as a KPI.

The following scenarios were tested:

Scenario 1	
Service	Dynamic
Parameter	Value
Reconciliation time	0.6 seconds

Scenario 2	
Service	Dynamic and Static
Parameter	Value
YOLOv5 Checkpoint	yolo5s.pt
SPCL Checkpoint UDA	msmt2market_resnet_ibn50a/model_best.pt h.tar
SPCL device	GPU\CPU

Scenario 3	
Service	Dynamic
Parameter	Value
Network	High/Low priority channels.

Scenario 4	
Service	Dynamic and Static
Parameter	Value
YOLOv5 Checkpoint	yolo5s.pt
SPCL Checkpoint UDA	msmt2market_resnet_ibn50a/model_best.pt h.tar
SPCL device	GPU
Quality of detections	AP

5.4.4 Test cases and results in THI platform

To perform the specific test case, a camera has been connected to the Zynq FPGA board and the associated HW IPs and SW drivers have been developed. The HW IP extract a frame (captured by the camera) and sends it (via a DMA engine) to the programmable part of the FPGA where the NEOX accelerator is realized. The FPGA runs a full-blown Linux version, and it can be easily connected to the cloud e.g., via typical ssh and sftp services. The NEOX AI-SDK was used to compress the network (targeting both the convolutions and the dense parts

of the person detection architecture). More details about the specific test case and the associated results are presented in the following table.

Test case name	CVAE low power	Test Case id	Test-03-01
Test purpose	To execute a CNN based CVAE application with less than 3.5mWatts power consumption. Compress the CNN model.		
Configuration	NEOX 2 core configuration.		
Test tool	Linux perf tool.		
KPI	Latency, power consumption, compression ratio		
	Step 1	Development of a lightweight DNN-based person detection application	
	Step 2	Integration of the person detection of application in the AI-SDK compression framework	
	Step 3	Integration of the person detection of application in the AI-SDK deployment framework	
	Step 4	Deployment of person detection in NEOX accelerator	
	Step 5	Camera connected to the FPGA platform	
	Step 6	Zynq FPGA platform connected to cloud or edge	
Test Verdict	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 of 3.04mWatts (for a processing rate of 30 Frames-per-second). To achieve the reported execution times the following optimizations have been employed: SIMD processing of the convolution layers, kernel code optimizations, balanced thread execution, and scratchpad optimizations.		
Additional Resources	None		

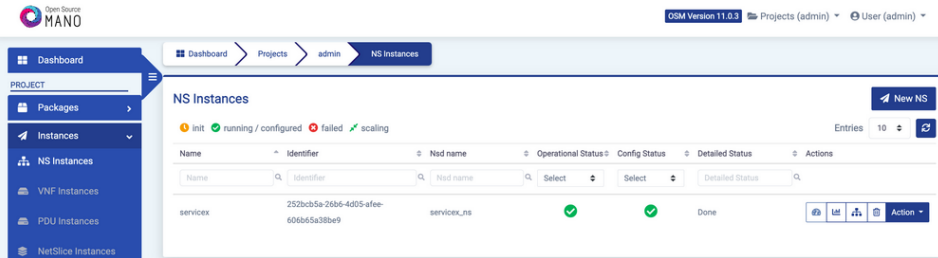
5.4.5 Test cases and results

As described before, several tests were carried out to validate the performance of the smart City Pilot, with the following results obtained.

Scenario 1

Test case name	Reconciliation time	Test Case id	Test-04-01
Test purpose	Verify the time needed for a change in the remote repository to be enforced in the deployed service.		
Configuration	None.		
Test tool	None.		
KPI	Reconciliation time.		
Components Involvement	OSM, Service X.		

Pre-test conditions	None.	
Future improvements	Debug and fix the error, which is responsible for not allowing communication with other services to occur.	
Test sequence	Step 1	<p>The Malaga environment comes with a two-node Kubernetes cluster pre-configured. Being necessary to:</p> <ul style="list-style-type: none"> • install and configure the FluxCD CLI on node1 • deploy FluxCD and OSM Ops services to the Kubernetes cluster • configure OSM Ops
	Step 2	Set OSM ops pipeline, connect the Git repo through FluxCD.
	Step 3	Analyse the reconciliation occurring.
Test Verdict	According to Martel, the average value for the reconciliation took about 0.6 seconds	

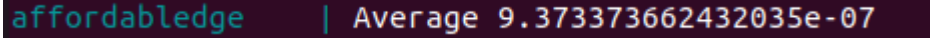
Additional Resources	 <p><i>Figure 39: Deployed service in OSM</i></p> <pre> \$ kubectl -n cf6786a7-cf47-42ce-9dbd-db7ffd509433 logs svc/api INFO: Will watch for changes in these directories: ['/usr/src/app'] INFO: Uvicorn running on http://0.0.0.0:5000 (Press CTRL+C to quit) INFO: Started reloader process [1] using StatReload INFO: Started server process [8] INFO: Waiting for application startup. INFO: Application startup complete. </pre> <p><i>Figure 40: Service x logs in the OSM</i></p> <pre> controller_runtime_reconcile_time_seconds_bucket{controller="gitrepository",le="40"} 2 controller_runtime_reconcile_time_seconds_bucket{controller="gitrepository",le="50"} 2 controller_runtime_reconcile_time_seconds_bucket{controller="gitrepository",le="60"} 2 controller_runtime_reconcile_time_seconds_bucket{controller="gitrepository",le="+Inf"} 2 controller_runtime_reconcile_time_seconds_sum{controller="gitrepository"} 0.6356981930000001 controller_runtime_reconcile_time_seconds_count{controller="gitrepository"} 2 # HELP controller_runtime_reconcile_total Total number of reconciliations per controller </pre> <p><i>Figure 41: Logs presented by the /metrics endpoints that showcase the metrics of the service</i></p>
----------------------	---

Result:

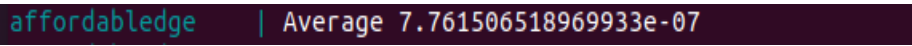
Considering the importance of the Service X, it is necessary that all the changes done remotely are able to be implemented as fast as possible. Having a reconciliation time, lower than 1 second satisfies the need of the Pilot.

Scenario 2

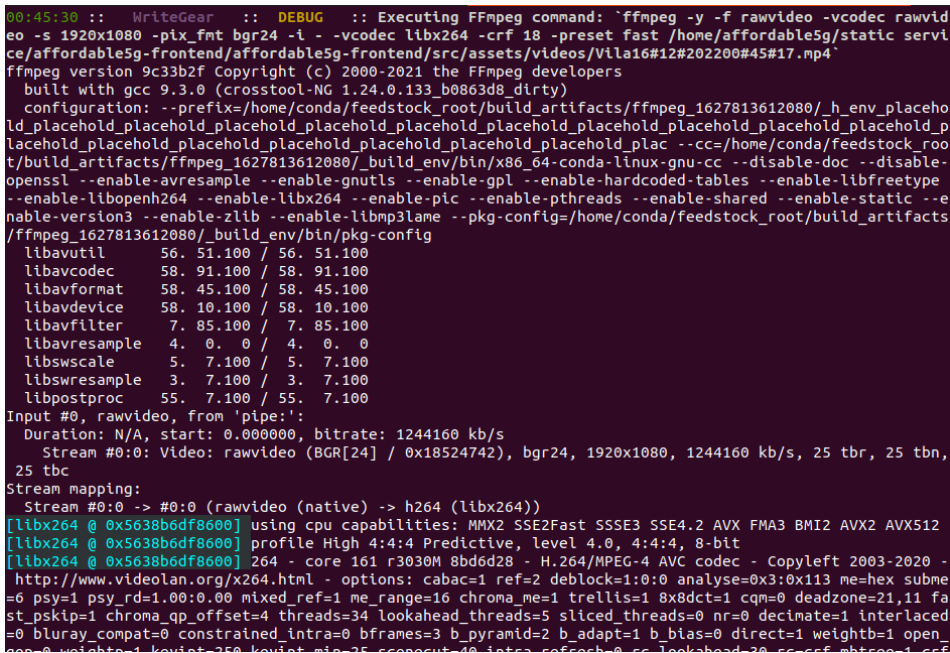
Test case name	Static edge inference CPU	Test Case id	Test-04-02
Test purpose	Verify the inference time, in a CPU-bound configuration, that the Edge solution of the static services takes to process a single frame.		
Configuration	CPU enabled.		
Test tool	None		
KPI	Inference time		

Components Involvement	All the building blocks of the static solution.	
Pre-test conditions	Have a pre-recording video to ensure that the same detections occur in every new test iteration.	
Future improvements	None.	
Test sequence	Step 1	Deploy Server solution in the Bullsequana.
	Step 2	Enable CPU-bound operation in the docker file of the Edge solution. And deploy it.
	Step 3	Analyse the time it takes for the Edge to process a single frame.
Test Verdict	On average a single frame is processed in 9.373373662432035e-07 seconds. As previously stated, the calculated value depends on several factors and the result can differ between tests.	
Additional Resources	 <i>Figure 42: Timestamp of the time it takes the frame to be processed by the CPU by the Edge</i>	

Test case name	Static edge inference GPU	Test Case id	Test-04-03
Test purpose	Verify the inference time, in GPU-bound configuration, that the Edge solution of the static services takes to analyse a single frame.		
Configuration	GPU enabled.		
Test tool	None		
KPI	Inference time		
Components Involvement	All the building blocks of the static solution.		
Pre-test conditions	Have a pre-recording video to assure that the same detections occur in every new test iteration.		

Future improvements	Verify the fidelity of using the newer Vidgear configuration, which allows bi-directional communication while supporting the multi-stream capabilities. Change used libraries to ensure the edge solution is able to be deployed in a jetson nano, in this manner the BullSequana has more resources for the needs of the Core solution.	
Test sequence	Step 1	Deploy Server solution in the BullSequana.
	Step 2	Enable GPU-bound operation in the docker file of the Edge solution. And deploy it
	Step 3	Analyse the time it takes for the Edge to process a single frame.
Test Verdict	On average a single frame is processed in 7.369914480174581e-07 seconds. As previously stated, the calculated value depends on several factors and the result can differ between tests.	
Additional Resources	 <i>Figure 43: Timestamp of the time it takes the frame to be processed by the GPU by the Edge</i>	

Test case name	Static Server inference CPU	Test Case id	Test-04-04
Test purpose	Verify the inference time, in CPU-bound manner, that the Server solution of the static services takes to analyse a single frame.		
Configuration	CPU enabled.		
Test tool	Vidgear debug logs.		
KPI	Inference time		
Components Involvement	All the building blocks of the static solution.		
Pre-test conditions	Have a pre-recording video to assure that the same detections occur in every new test iteration.		
Future improvements	The ideal solution is not based on a CPU-bound configuration.		
Test sequence	Step 1	Deploy Server solution in the Bullsequana.	
	Step 2	Enable CPU bound operation in the docker file of the Server solution. And deploy it.	

	Step 3	Analyse the time it takes for the Server to process a single frame.
Test Verdict	<p>On average a single frame is processed in 0.5130387544631958 seconds.</p> <p>The CPU bound operation provided 0.1 Frames per second, which is not enough for the needs of a real time security system.</p>	
Additional Resources	 <p>Figure 44: Core videogear client, booting the process of receiving the frame from the Edge. With CPU enabled</p>	

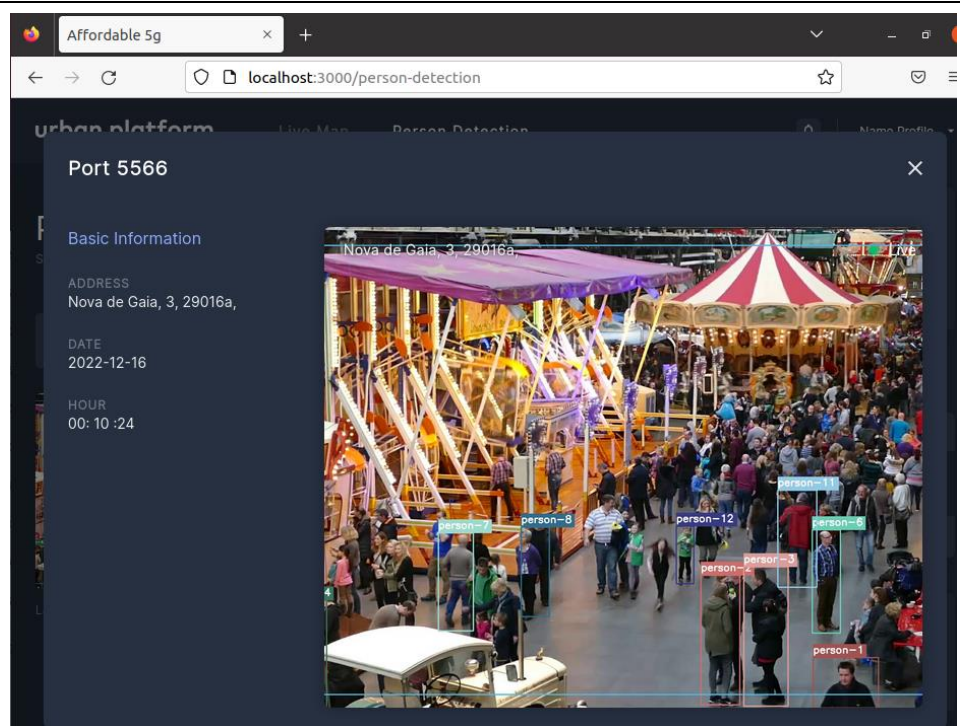


Figure 45: Fronted service, showing the processed frame

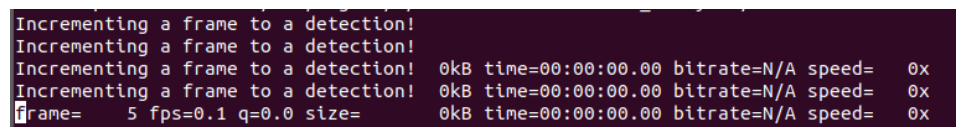


Figure 46: Core logs with the CPU configuration enabled

Test case name	Static edge inference GPU	Test Case id	Test-04-05
Test purpose	Verify the inference time, in GPU-bound manner, that the edge solution of the static services takes to analyse a single frame.		
Configuration	GPU enabled.		
Test tool	None		
KPI	Inference time		
Components Involvement	All the building blocks of the static solution.		
Pre-test conditions	Have a pre-recording video to assure that the same detections occur in every new test iteration.		



Future improvements	Verify the fidelity of using the newer Vidgear configuration which allow bi-directional communication while supporting the multi stream capabilities. Change used libraries to ensure the edge solution is able to be deployed in a jetson nano, in this manner the BullSequana has more resources for the needs of the Core solution. Use the GPU for the ML algorithm processes and perform further debug as to optimise the frame rate.	
Test sequence	Step 1	Deploy Server solution in the BullSequana.
	Step 2	Enable CPU bound operation in the docker file of the Edge solution. And deploy it
	Step 3	Analyse the time it takes for the Edge to process a single frame.
Test Verdict	An issue when enabling the GPU was found, not allowing it correct usage. The present test failed.	
Additional Resources	None.	

Result:

Although it was not possible to fully demonstrate the impact that the GPU has in the processing of ML algorithms. The comparison of the inference time of each frame by the Edge allows us to confirm the important rule that the GPU has in the processing of data.

Scenario 3

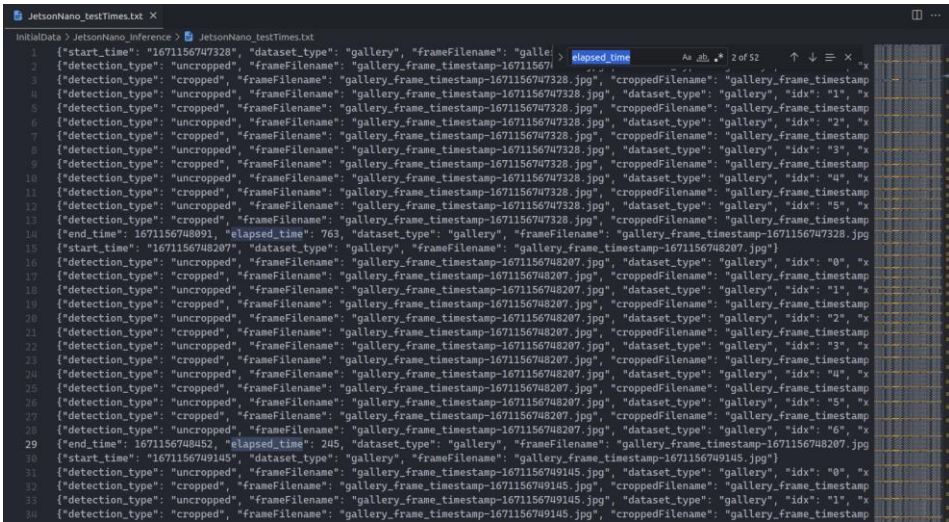
Test case name	Bit rate	Test Case id	Test-04-06
Test purpose	Test the impact of the data transmission speed of the Security Guard smartphone when it is in a lower priority traffic.		
Configuration	Set the priority to a lower one for the security guard.		
Test tool	None.		
KPI	Bitrate		
Components Involvement	5G network, dynamic service		

Pre-test conditions	The security guard smartphone needs to be in a higher priority profile to change to the lower one. And his device needs to be connected to the 5G network infrastructure.	
Future improvements	None.	
Test sequence	Step 1	Have all the dynamic service components up. Verify the number of the SUPIs of the security Guard Smartphone.
	Step 2	Trigger an alert to change the priority.
	Step 3	Analyse the time it takes for the websocket to send the data from the smartphone to the Jetson Nano.
Test Verdict	Since the test was performed with a smartphone connected via VPN to the UMA network. It was not possible to directly access the 5G network and take advantage of the different priority profiles.	
Additional Resources	 <p><i>Figure 47: Security guard smartphone in the higher prioritisation profile</i></p>  <p><i>Figure 48: Security guard smartphone during a lost child occurrence, in lower prioritisation profile</i></p>	

Result: The acquired results don't allow us to extrapolate and show the benefits of the 5G networks capabilities for fast data transmission.




Scenario 4

Test case name	Global inference time for detections time and cropping	Test Case id	Test-04-07
Test purpose	Verify the amount of time that the detection and cropping occur for the solution deployed in the Jetson Nano for the Dynamic service.		
Configuration	None.		

Test tool	None.	
KPI	Inference time	
Components Involvement	Dynamic service	
Pre-test conditions	None	
Future improvements	None	
Test sequence	Step 1	Have all the dynamic service components up.
	Step 2	Trigger an alert and start search.
	Step 3	Analyse the time it takes for the frame to be processed and cropped.
Test Verdict	None	
Additional Resources	 <p>Figure 49: logs of the cropping and detection operation</p>	

Test case name	maP	Test Case id	Test-04-08
Test purpose	Verify the quality of the detection for the Dynamic service.		
Configuration	None.		
Test tool	None.		
KPI	maP		

Components Involvement	Dynamic service	
Pre-test conditions	None	
Future improvements	Develop a manner to crop the lost children photo from the background. In this way only the lost children will be analysed, mitigating false positives caused by the ML algorithm associating the background of the mall (or where the search takes place) in the prediction process.	
Test sequence	Step 1	Have all the dynamic service components up.
	Step 2	Have an active missing children alert.
	Step 3	And finish the experiment after a successful match is found
	Step 4	<p>In order to measure this metric, we first logged all the information required for the calculations, mainly the top left and bottom right coordinates of the detection, the confidence in the classification and the class of the detected object.</p> <p>Then we performed a post-processing step to format and store the inference results with the required data, where each file contains the information of the detections in a single frame.</p> <p>This allowed us to obtain well-formatted inference results.</p>
	Step 5	The next task was to create the ground truth labels, by labelling each frame sent from the mobile phone to the Jetson Nano with bounding boxes surrounding people and associated object class, and structure the data similarly to the inference results, but without the confidence value for each detection.
	Step 6	Given the correctly structured inference results and ground truth, it is now possible to calculate the IoU for each detection and the Precision and Recall values. We use the 11-point interpolation method to plot the Precision/Recall curve that allows us to calculate the Average Precision of the detection of objects of class "person".
Test Verdict	This test would have allowed us to measure the quality of detections for the Dynamic solution. However, some difficulties emerged in the process of calculating the Precision/Recall Curve, which blocked further analyses.	

<p>Additional Resources</p>	<div data-bbox="496 315 1444 560"></div> <div data-bbox="536 584 1414 616"><p><i>Figure 50: Some cropped detections of individuals detected by the Yolov5</i></p></div> <div data-bbox="874 719 1069 1111"></div> <div data-bbox="638 1135 1310 1167"><p><i>Figure 51: Missing kid photo match, with threshold of 0.8</i></p></div> <div data-bbox="884 1196 1059 1570"></div> <div data-bbox="748 1594 1198 1626"><p><i>Figure 52: Provided missing kid photo</i></p></div>
---------------------------------	---

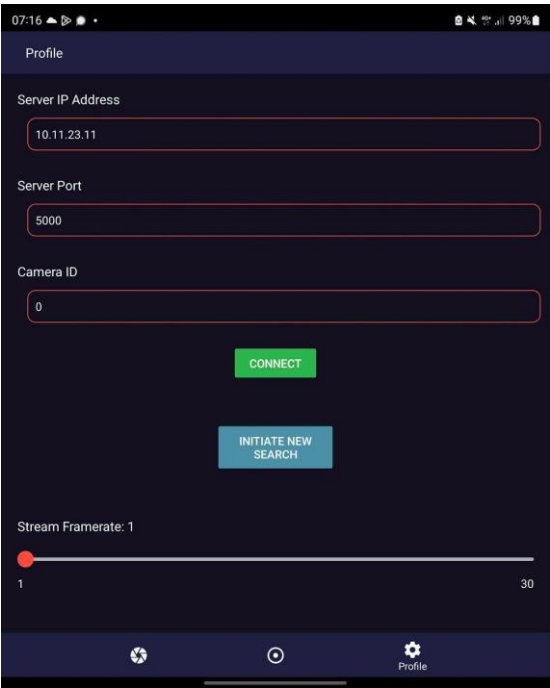


Figure 53: Mobile application interface



Figure 54: Ground truth process

Result: As shown by the captures of this scenario, the base system developed for the Dynamic service is capable of capture, process, crop, and redirect the data between the different components. As previously stated, the quality of the detection of any ML algorithm is dependent on different factors. However, considering the streamline on which the data flow and the current results, the decoupling of the models as well as the development of the solution was a success.

5.4.6 Innovations and conclusions

The Smart City pilot, thanks to the integration between different companies, was able to showcase a small-scale use case for the deployment of 5G technology. Coherently align itself with the objectives and the underlying strategy of the Affordable5G project.

As previously stated, this pilot has to face technical challenges intrinsic to the innovations that build the differentiation and value for the services, which required adequate investigation, action and adaptation of the pilot to reach its goals.

And with the tests mentioned in the previous chapter, we were not only able to test the capabilities of ML algorithms, but also to verify the correct deployment of the services built around them in the different devices. Having each one its own architecture and specific challenges for their proper instantiation. Although the validation of some service parts did not go as expected, it was possible to incorporate several technologies. However, the Smart City Pilot's overall goals were reached, and several innovations were made in the different development phases. For instance, the dynamic service besides being an opportunity to explore the symbiotic usage of ML technology and the fast data transmission of the 5G network at the edge originating the following new innovations:

- Elaboration of architecture with scalability in mind, through the decoupling of SPCL and YOLOv5. In this way, the central server is focused on the re-identification of people, while the detection and cropping of images used as input to the SPCL can be done in multiple edge nodes, not existing a performance bottleneck directly related to the number of chambers in the environment
- Creation of a system which supports the streaming between different layers, using websockets for communication between a mobile App, Jetson Nano and Redis streams. While also supporting the communication between the Jetson Nano and Bullsequana.
- Development of a central system that allows communication, synchronisation between the different parts of the system, and managing searches for missing persons through an API.

While for the static service, the main innovation was the creation of a central system that uses different types of ML algorithms to perform person detections and allow the visualisation of livestream and stored videos in a multi-camera environment.

The described scenario implements several blocks and services capable of working in an integrated and fluid way, making it possible to combine the orchestration and management of a 5G network to improve the response to a crisis scenario, such as the loss of someone in ample space. Thus, the pilot is positioned as quite innovative and capable of promoting the first step in crisis scenarios with automation via computer vision solutions. In the future, it will be necessary to improve the services developed so that they allow greater interaction with the end user and efficient bandwidth management when facing scenarios with a lot of people.

6 CASTELLOLI TESTBED

6.1 Final picture

The Affordable 5G architecture implemented in Castellolí is depicted in Figure 55. The differences that were implemented between these last months of the projects are the ones related to the RU and DU deployment. Castellolí platform decided to install the O-RAN solution from Accelleran and Benetel. Additionally, due to several problems (explained in the following sections) occurred during these integrations with the platform, Adva's switch was decided not to be used.

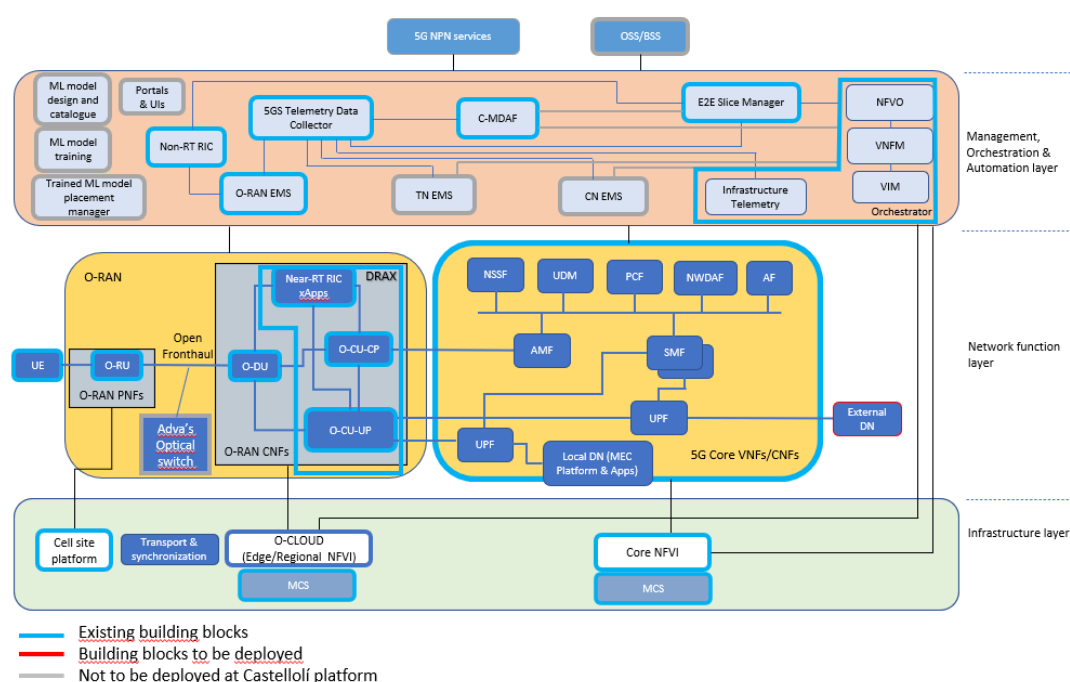


Figure 55. Final architecture of Castellolí platform

The deployment of the RU and DUs were in two different locations in the circuit. The Node #1 and Node #9. The Node selection was determined by its fiber installation. It was only possible to have a good fiber connection from the elected nodes to the control room, where the shared DU was installed. In the following figures, Figure 56 y Figure 57, we can observe the equipment that was used in each case: Antennas from Alpha wireless, GPS antennas from Accelleran and the Radios from Benetel. These elements were interconnected between them, and the Radio was connected into the cabinet through a fiber, that at the same time was connected into the control room.



Figure 56: Node #1 RAN installation



Figure 57: Node #9 RAN installation

In the control room, Figure 58 shows the different equipment used for the project:

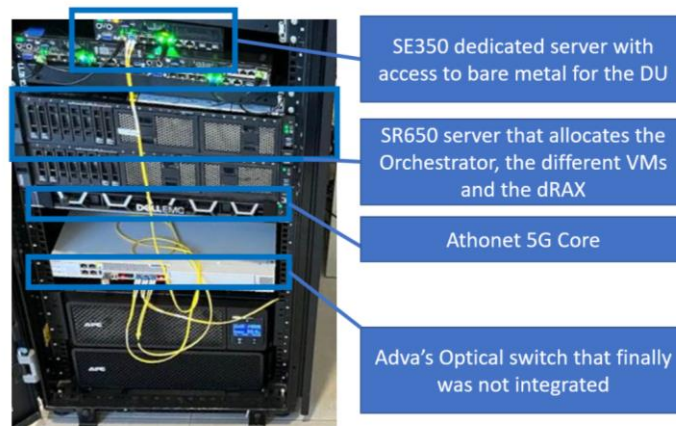


Figure 58. Control Room

A specific server had to be used to allocate the DU that was connected to both of the RU installed in the Nodes. This server had not to only meet several computational requirements but also the DU was required to access to the bare metal.

In the first place, the fibres from the nodes were connected to the DU through the switch, but this caused that the RU and the DU did not see each other. So finally, the fibres were directly connected to the SE350. This integration is explained in section 6.2. With all the elements installed, Castellolí's platform had the full Affordable 5G architecture working, ready to be tested.

6.2 Fronthaul challenges

The RAN deployment scenario at Castellolí site originally included RunEL O-RU and Eurecom OAI O-DU. The fronthaul interface planned was O-RAN split 7.2 (eCPRI).

The integration between Eurecom and RunEL was not ready at the moment, and as a mitigation plan, Accelleran proposed another set of pre-integrated RU-DU components: O-RU RAN650 by Benetel and DU from Phluido. The RAN split is still 7.2, but the fronthaul interface is different from O-RAN defined eCPRI. The fronthaul interface was developed by Phluido and called "revolutionary radio-agnostic fronthaul protocol" (RAFP), similar to CPRI over UDP and compressed (which was proposed to the O-RAN Alliance for standardization but was not the final selection).

After initial fronthaul testing, the integration team observed that the presence of ADVA fronthaul switch between RU and DU caused malfunctioning of the radio link. The problem disappeared when the RU and DU were directly connected via fiber. For future investigation, as described in section 4.3, ADVA fronthaul switch has been sent to Accelleran lab.

Also, Accelleran team observed that even the direct fiber connection depends on a NIC PCIe cards used at DU. For example, Intel X710 NIC causes the problem, while X550 NIC works fine.

The staging at Accelleran lab showed that packet size of the RAFFP is 2608 bytes. Any switching device with “store-and-forward” design introduces an inevitable delay, roughly equal to packet size/line rate. For example, the measurements on ADVA switch showed the latency of 6 μ s and the jitter for a fixed packet stream was below 1 μ s.

To figure out what causes the problem, latency, jitter or packet loss, ADVA has sent to Accelleran 2 spools of 2.5 km fiber. This fiber length introduces approximately 12 μ s constant delay (assuming the propagation delay 5 μ s per km with 1310nm laser) and no jitter. With the long fiber the problem has not been reproduced, so we have concluded that the root cause was the jitter. We cannot assess the maximum allowable value for the jitter though. Further ongoing tests are described in section 4.3.

RAFFP interface seems incompatible with any available fronthaul switches. It means that there is no way to inject PTP stream into this interface. DU must receive the PTP stream via other physical links and not via the fronthaul link.

6.3 Network characterization

With the architecture and elements deployed in Castellolí all clear, the tests started in order to define and characterize the network.

The final network architecture consists of two slices, characterized by two UPFs (one in the central server and one logically located in an edge node, but actually installed in the same server) that share the same control plane and RAN.

Slice 1 represents the main end-to-end network, performing a best-effort service, and has been identified by the two parameters Slice/service Type (SST) and Slice Differentiator (SD) having initial values 1 and *nil* (1, *nil*) respectively, and a Data Network Name (DNN) set to *internet*. Slice 2 represents a mission-critical service, with a guaranteed quality of communication, and has been identified by the two parameters SST and SD having initial values 1 and 000001 (1, 000001) respectively, and a DNN set to *mec*.

A total of 5 SIMs has been provided and provisioned into the 5GC, associated with three profiles. Summarizing, Table 2 illustrates the specific set of configurations:

Table 2: Slicing and associated SIMs configuration schema. Note there is a SIM having access to both slices (highlighted in bold).

Slice	(SST, SD)	Description	DNN	SIMs associated
Slice 1	(1, <i>nil</i>)	Central core	<i>internet</i>	001011001009398, 001011001009399, 001011001009400, 001011001009402
Slice 2	(1, 000001)	Edge node	<i>mec</i>	001011001009401, 001011001009402

Starting from this setting, two series of preliminary integration tests were performed, including the related UE attachment and traffic tests: the first one related to the commissioning of Slice 1, the second one related to Slice 2.

6.3.1 Slice 1 integration tests and final results

The preliminary integration procedures were aimed at the deployment and configuration of the first slice (1, *niI*). The Athonet 5GC has been properly configured to include the profiles of the SIMs associated with that slice and the network plan has been applied based on the internal network architecture of the Castellolí testbed.

The RAN side has been configured accordingly to match the slice identifier and attached to the 5GC.

Here follows a summary of UE attachment tests and results obtained: Successful tests have been performed with the OnePlus NORD 10 model, achieving attachments and data connectivity (see Figure 59). Furthermore, in Figure 60, the two mentioned supported slices can be shown in the confirmation response (Item 1 and 2), meaning that the UE can access those (the presence of Item 0, with SD = 000001, although it is part of Slice 2, was added only for some internal testing, and was not actually adopted for Slice 1).

```

  ✓ NG Application Protocol (NGSetupResponse)
  ✓ NGAP-PDU: successfulOutcome (1)
    ✓ successfulOutcome
      procedureCode: id-NGSetup (21)
      criticality: reject (0)
      ✓ value
        ✓ NGSetupResponse
          ✓ protocolIEs: 4 items
            ✓ Item 0: id-AMFName
              ✓ ProtocolIE-Field
                id: id-AMFName (1)
                criticality: reject (0)
                ✓ value

```

Figure 59: Successful message of UE attachment.

```

  ✓ Item 3: id-PLMNSupportList
    ✓ ProtocolIE-Field
      id: id-PLMNSupportList (80)
      criticality: reject (0)
      ✓ value
        ✓ PLMNSupportList: 1 item
          ✓ Item 0
            ✓ PLMNSupportItem
              ✓ plmnIdentity: 00f110
                Mobile Country Code (MCC): Unknown (1)
                Mobile Network Code (MNC): Unknown (01)
              ✓ sliceSupportList: 3 items
                ✓ Item 0
                  ✓ SliceSupportItem
                    ✓ s-NSSAI
                      sST: 01
                      sD: 000001
                ✓ Item 1
                  ✓ SliceSupportItem
                    ✓ s-NSSAI
                      sST: 01
                      sD: 000000
                ✓ Item 2
                  ✓ SliceSupportItem
                    ✓ s-NSSAI
                      sST: 01

```

Figure 60: Details of successful message, showing the supported slices.

After this first milestone, other UEs were tested: One plus Nord with snapdragon chipset, Ulefone 5G and Dodgee 5G. Additionally, the Sunwave CPX60P (indoor) and CPX80P (outdoor) CPEs that Cellnex have on the Castellolí site were also considered.

Unfortunately, some of them failed to attach (an example is shown in Figure 61), and other successfully attached but after a few seconds the connection was lost.

After some troubleshooting, a partial lack of compatibility between the UE device models and the deployed 5G network was found. In details, some tested smartphone models are designed to always expect a *voice centric* connectivity, thus requiring an available IMS service. Since the IMS service has not been introduced in the testbed, once the device is connected, it drops connectivity after a few seconds, since it does not find any IMS service.

```

  ▾ NG Application Protocol (InitialContextSetupFailure)
    ▾ NGAP-PDU: unsuccessfulOutcome (2)
      ▾ unsuccessfulOutcome
        procedureCode: id-InitialContextSetup (14)
        criticality: reject (0)
        ▾ value
          ▾ InitialContextSetupFailure
            ▾ protocolIEs: 3 items
              ▾ Item 0: id-AMF-UE-NGAP-ID
                ▾ ProtocolIE-Field
                  id: id-AMF-UE-NGAP-ID (10)
                  criticality: ignore (1)
                  ▾ value
                    AMF-UE-NGAP-ID: 8
              ▾ Item 1: id-RAN-UE-NGAP-ID

```

Figure 61: Example of a failed UE attachment with UE model DOGEE 5G

This demonstrated the need to use ‘data centric’ UE that don’t expect IMS. While in some smartphones, this can be achieved by changing the phone settings, in many cases it is not possible to change this behaviour. Industrial CPEs are expected to be more compatible in this respect, but in the timeframe and due to other issues, it was not possible to fully test with all devices and characterize this problem sufficiently.

The UE that was known to work was setup (in the indoor Castellolí control room) to be permanently online and remotely controllable to allow additional system testing.

All of these issues are justified by the lack of full technological maturity that 5G is experiencing today: unlike 4G, which boasts a fifteen-year technological maturity, including its compatible devices, 5G is still a very young technology. First, produced 5G compatible devices do not yet support all the new features and functionalities introduced by the 5G standards. This results in UEs possibly not responding properly to an attachment, or not sending the right packets, or not whitelisting all possible PLMNs, and not handling slices correctly.

While initially both RU’s were operating at nominal downlink Tx power, after a while it seemed that the Radios switched in a very low power transmit mode. This was remotely investigated but no obvious problem or solution was detected. The problem may be due to a hardware failure of the RU power amplifier. This lower power meant that the remotely controllable UE, described above, no longer had 5G coverage of the network and it was hence not possible to continue other testing.

The ongoing solution is to be replace one RU with the spare unit that is onsite and send a fault unit back to ACC labs for further investigation, with possible shipment back to Benetel for repair, if deemed a hardware failure. At the time of writing, this is ongoing.



Figure 62: Smartphones trying to connect to the Affordable 5G network

6.3.2 Slice 2 integration tests and final results

The integration of Slice 2 takes place through the introduction of a second UPF logically separated from the central core that represents Slice 1. This second UPF is connected to a different DNN called *mec*, which has the task of providing connectivity to the emergency services. Slice 2 was configured with the combination (SST, SD) = (1, 000001). Also here, the 5GC has been configured to include the profiles and the specific SIMs associated with this slice. The RAN has been properly configured to support this second slice as well. Notice that the on-site integration and end-to-end network configuration work has resulted very insightful and instructive for the involved partners. In particular, the proposed two-slice setup turned out to be less straightforward integration-wise than the more common one-slice configuration (applied for instance in the other project's testbed at UMA). Aspects that have required specific attention and non-negligible troubleshooting have been the management and exchange of network slice indicators and related information by all network components, the ways UEs handle such indicators and require access to specific slices to the 5GC, and the vendor-specific low-level implementation of the handling mechanisms of such parameters.

First UE attachment tests for Slice 2 failed. Initially, some misconfigurations were the cause of these failures, but once everything was fixed and set up correctly, the following set of tests were unable to have a working end-to-end connectivity. Besides all the 5GC and RAN components were properly configured providing end-to-end 5G connectivity to the first slice, the connection to the second slice through the second UPF deployed at the edge did not succeed. Our initial investigations pointed that such incompatibilities could come from the commercial UE side, that was not able to manage multiple NSSAIs (Network Slice Selection Assistance Information) correctly, hence not supporting multiple network slices and as a result not allowing to properly validate this scenario. Further investigation and further testing with more commercial UEs is required in order to solve this issue, but for lack of time this work will be performed outside of the scope of this project.

In functional terms, the second slice was correctly instantiated with the activation of the second UPF via API by the orchestrator. For this integration, a NearbyBlock **Athonet Dynamic UPF Deployment** has been developed and integrated in the orchestrator for dynamically activating/deactivating Athonet's UPF instances. This block builds on top of another reusable NearbyBlock: **Slice Descriptor** (Vendor Independent). Figure 63 provides the list of NearbyBlocks used in this demo.

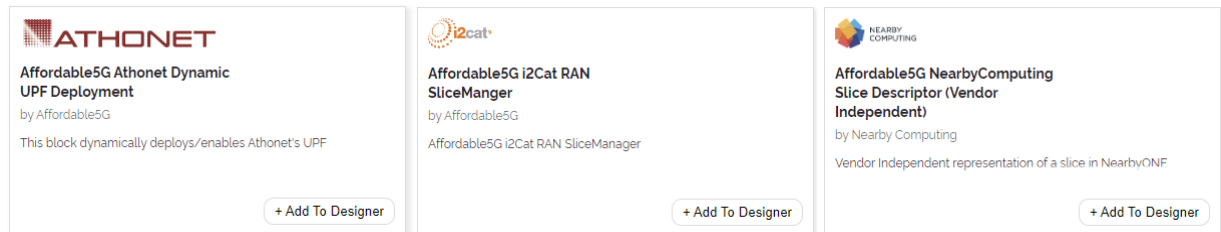


Figure 63 NearbyBlocks list showing Athonet UPF Deployment and Generic Slice Descriptor

The *Athonet Dynamic UPF Deployment* block will constantly monitor the existing instances of *Slice Descriptors*. As soon as it detects any instance, it will use the integration with Athonet 5gG Core API to enable the UPF instance, and when the Slice Descriptor block is deleted, the "Athonet Dynamic UPF Deployment" Block handles the deactivation of the UPF.

Block default values

```

1  Block:
2  InstanceId: 12345678-9012-3456-7890-123456789012
3  deployments:
4  slice:
5    variables:
6    slice_definition:
7      PLMNId:
8        mcc: "001"
9        mnc: "01"
10     dataNetworks:
11       - name: mec
12       - name: ims.mnc001.mcc001.gprs
13     id: 1
14     name: slice-2-ab2112
15     sNssai:
16       sd: 1
17       sst: ab2112
18     slicePolicyFilter:
19       accessTypes:
20         - 3GPP
21       name: filter-datanetwork-1-ab2112
22     subnets:
23       - name: subnet-datanetwork-1-ab2112
24       networkFunctions:
25         - name: amf1
26         - name: smf1

```

Figure 64: Vendor Independent Slice Descriptor Block

Block default values

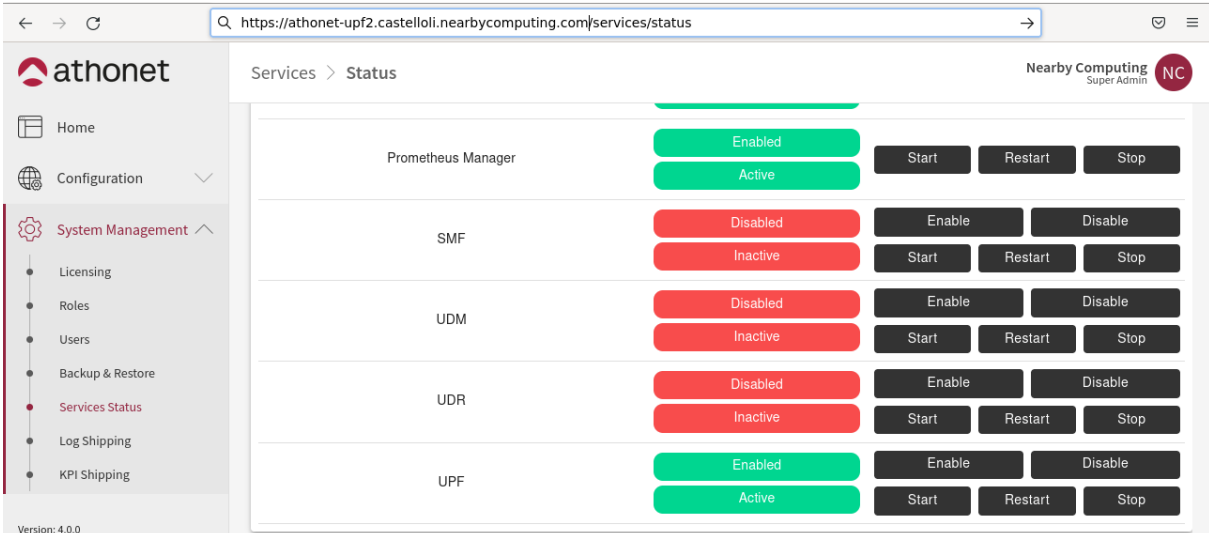
```

22 chart:
23   name: CRUDController
24   repo: https://registry.nearbycomputing.com/chartrepo/apps
25   version: 0.2.0
26 variables:
27   appname: upf
28   baseEndpoint: https://10.17.7.247/api/1
29   customCheckEndpoint: services/resources/upf
30   customDeleteEndpoint: services/resources/upf/stop
31   customLoginURL: auth/login
32   customPostEndpoint: services/resources/upf/start
33   enabled: false
34   image: external/crud-controller:latest
35   password: [REDACTED]
36   user: nearbyadmin

```

Figure 65: Athonet Dynamic UPF Deployment Block

As shown in Figure 64, the vendor independent *Slice Descriptor* block includes general 5GC slicing parameters that are commonly used for a slice creation. In addition, depending on the 5GC vendor, different parameters of the set available are included in the integration, as seen in Figure 65. In this specific use-case, the main functionality to be supported is the activation/deactivation of the second UPF, so the main parameter to be used is the name of the slice to identify the UPF instance whose status needs to be updated. After these parameters' setting, the orchestrator can execute the configured requests to the exposed 5GC API, in order to activate/deactivate the UPF resource. The confirmation of the successful operation can be directly monitored in the provided Athonet's 5GC Dashboard, as shown in Figure 66 and Figure 67, where the UPF component is active (green) and inactive (red) respectively.



The screenshot shows the Athonet dashboard with the 'Services > Status' view. The dashboard lists several services and their status:

Service	Status	Actions
Prometheus Manager	Enabled / Active	Start, Restart, Stop
SMF	Disabled / Inactive	Enable, Disable, Start, Restart, Stop
UDM	Disabled / Inactive	Enable, Disable, Start, Restart, Stop
UDR	Disabled / Inactive	Enable, Disable, Start, Restart, Stop
UPF	Enabled / Active	Enable, Disable, Start, Restart, Stop

The UPF service is currently active, indicated by green status bars. The dashboard also includes a sidebar with navigation options like Home, Configuration, and System Management, and a top navigation bar with the Athonet logo and user information.

Figure 66: Athonet upf2 dashboard with upf active

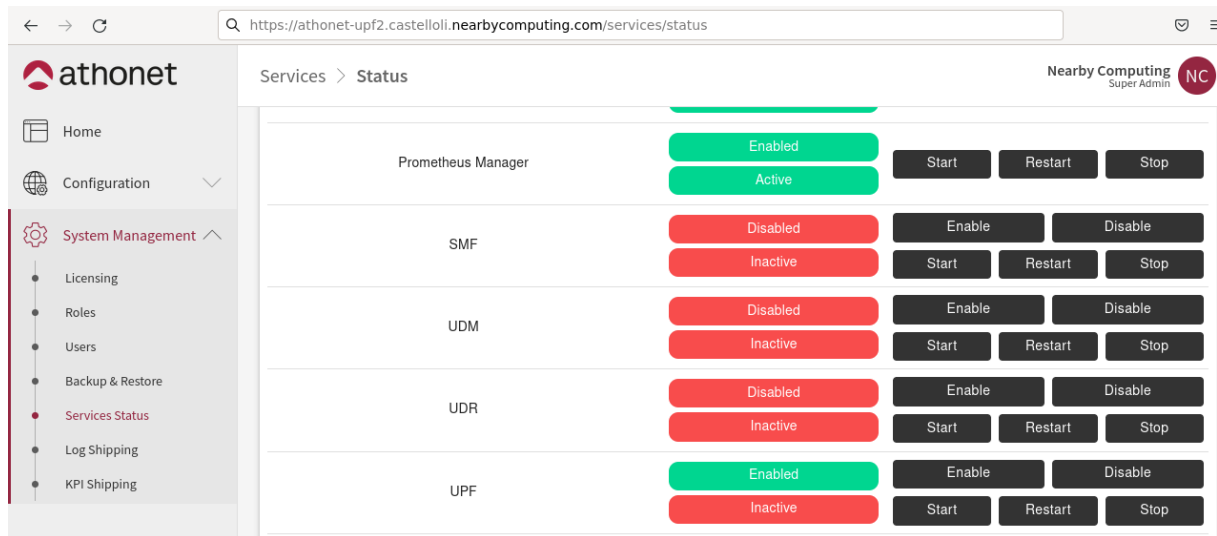


Figure 67: Athonet upf2 dashboard with upf inactive

Due to the different problems encountered into the Castellolí's platform deployment, the tests that could be performed show a network behaviour represented in Figure 68.

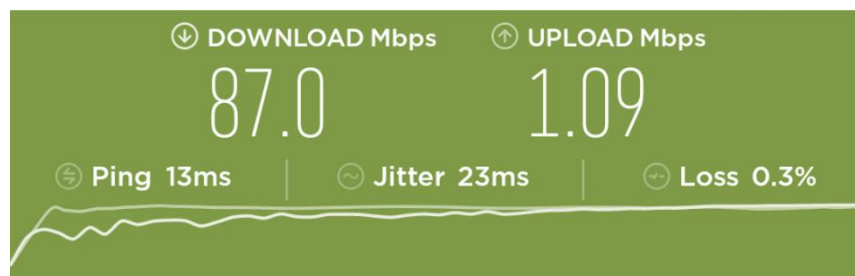


Figure 68: Castellolí platform Network test

The Latency has a variation of 34 ms. The best result was 13 ms and the worst 47 ms. This shows that the network is not as stable as it should be, many improvements should be applied in order to have a reliable network.

The jitter also presents an important variation. The values measured differ from 15ms until 23 ms. The Downlink Throughput is between 81 Mbit/s and 90 Mbit/s and the Uplink Throughput is not higher than 1.5 Mbit/s in most of tests.

From these values we can conclude that the network is not having expected 5G performance especially in radio uplink and overall latency values and, as said, further network investigation and several configuration improvements must be done in a scope beyond the project.

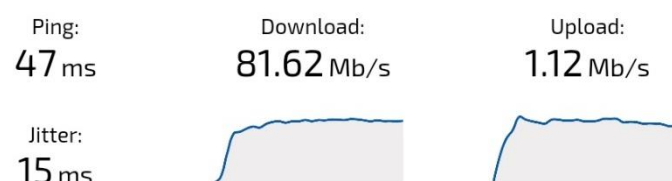


Figure 69: Castellolí's platform Network characterization parameters

6.4 MCS Pilot

6.4.1 Building blocks

Scenario 1

The architectural view of emergency communications pilot's building blocks for Scenario 1 are the ones depicted in the Figure 70 below. For this scenario one single slice across the Affordable 5G network is considered. Mission Critical users in a concrete coverage area are connected via the RAN and the 5GC to the MCS server. In this scenario, the NWDAF telemetry system is implementing machine learning approaches for optimization purposes of the tasks and procedures performed within Mission Critical service. The orchestrator on its side is managing the different network services lifecycle and necessary resources allocation and will be the component responsible of the escalation action in case it is requested.

Mission Critical Service instances allow end-users to communicate via Mission Critical Push To Talk (MCPTT), Mission Critical Video (MCVideo) and Mission Critical Data (MCData), following 3GPP guidelines. The communication between the MCX service, the Telemetry Module (NWDAF) and the Orchestrator is sustained on a Prometheus client server base. The Prometheus instance will be in charge of receiving and storing the metrics information from the MCX service, as well as sending alarms to the Orchestrator in case the Telemetry Module requests it.

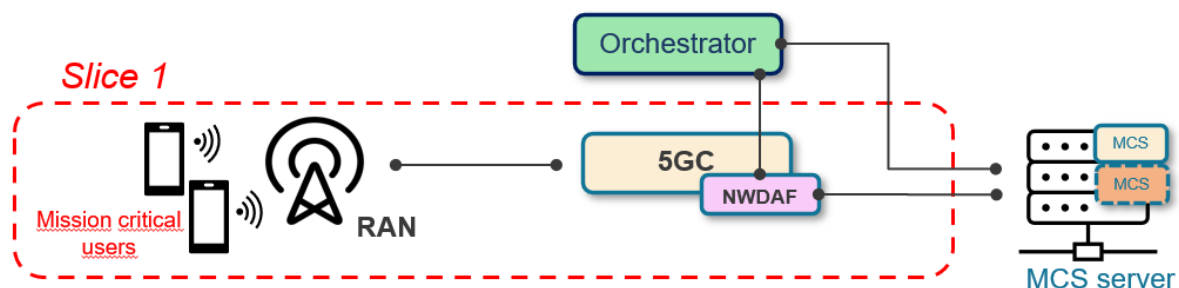


Figure 70 : Building blocks architecture for scenario 1

Scenario 2

The MCS pilot's general architecture is shown in Figure 71. As illustrated, two slices are considered for the pilot execution, representing the best-effort service (served by the central site) and the reserved MCS service (served by the edge site), and both share some of the 5G network components (i.e., RAN and central 5GC).

Starting from the end-user side, the UEs represent the devices communicating through the 5G network. The UEs belonging to Slice 1 have a configuration with support for default best-effort services, while those belonging to Slice 2 are configured to leverage dedicated MCS services, as they are used by the PPDR team to communicate with the MCS edge server (Nemergent).

The RAN is unique and shared by the two slices and is therefore connected to both UPFs and to the central 5GC's AMF.

The 5GC (Athonet) has a control plane containing all the usual network functions, like AMF, SMF, AUSF, PCF, etc. (see deliverables D3.1 [6] and D3.2 [7]), shared by both slices.

Furthermore, on the same physical server, there is an UPF for the best-effort traffic management, which is connected to the default Data Network (Internet). This set (central 5GC and UPF) constitutes the core subnetwork of Slice 1. Another UPF, part of Slice 2, is also installed on the same physical server, but it is installed in a different virtual machine and virtually separated from the rest of the 5GC; it represents an actual edge UPF, still managed by the central control plane.

The central 5GC exposes a specific API for its management, in particular for enabling/disabling the second slice. The 5GC API is leveraged by the Slicing Orchestrator (NearbyComputing), at the request of a Control Emergency center (Nemergent) when an emergency event occurs. Furthermore, the slicing orchestrator, receiving the emergency notification, instantiates the MCS edge system (Nemergent) at the edge, which represents the actual Data Network for the second UPF.

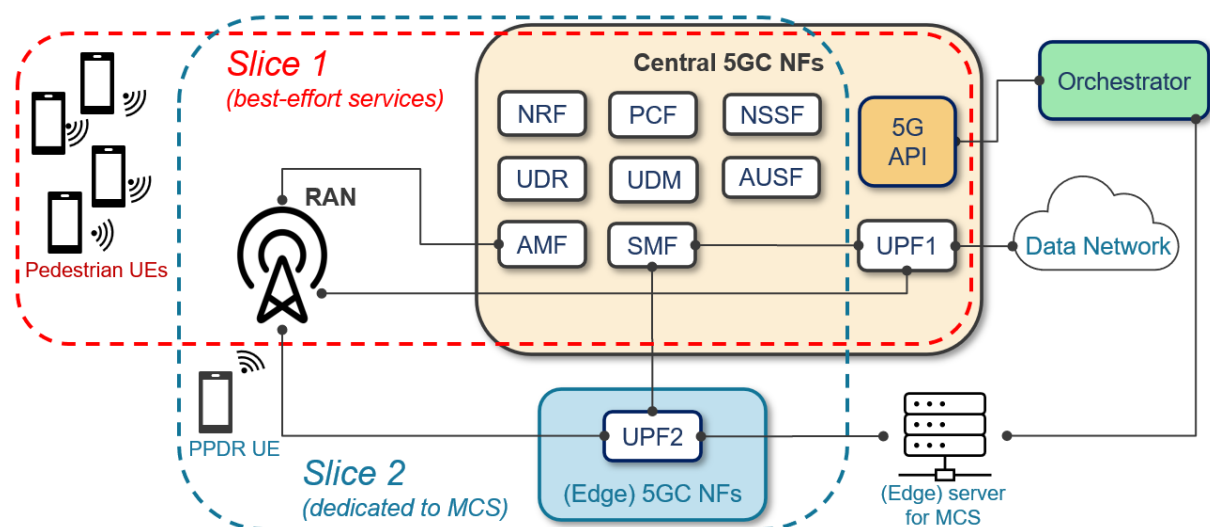


Figure 71: Building blocks architecture for scenario 2.

Extended Scenario 2

The represented building blocks are identical as the ones described in Scenario 2. In this one besides, a single slice and two distinguished MCS servers are involved in order to illustrate the MCX service migration from one server to another in case there is an eventual infrastructure failure.

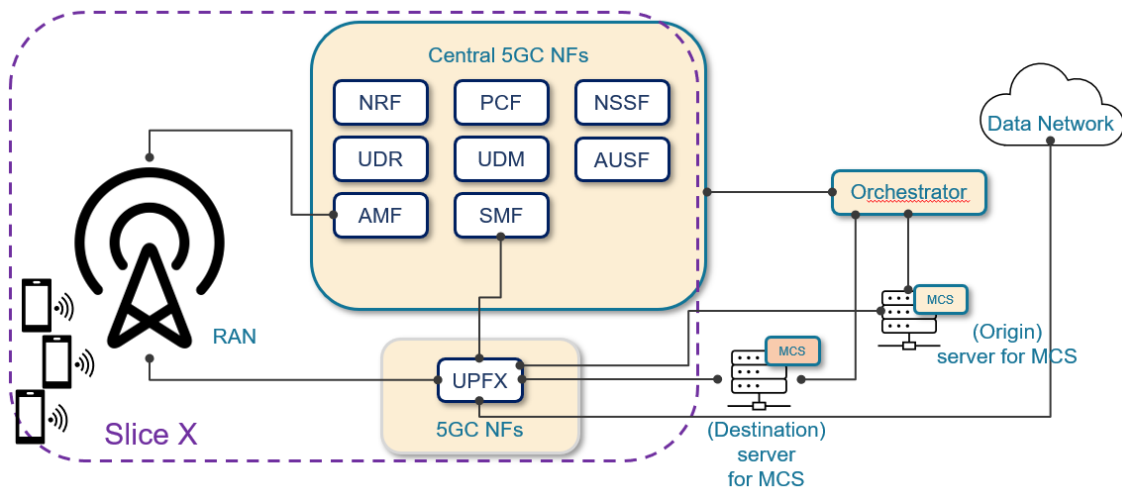


Figure 72: Building blocks architecture for extended scenario 2

6.4.2 Configuration and automation interfaces

6.4.2.1 Slice Manager extensions for SNSSAI support

To support the Castellolí use case the slice manager module proposed in deliverables D3.1 [6] and D3.2 [7] had to be extended to support Single Network Slice Selection Assistance Information (SNSSAI) based slicing. Notice that in our original implementation slicing was based on MOCN functionality, whereby each slice would require a separate 5GC control plane supporting a different PLMNID. Adding SNSSAI support allows us to support multiple slices within a single 5GC.

The required extensions were implemented in two steps, which we report in the next sections:

- Step 1: RAN Controller integration with the Accelleran dRAX.
- Step 2: Slice Manager integration with RAN Controller.

RAN Controller integration with Accelleran dRAX

Configuring a SNSSAI based network slice requires configuring the PLMNID and SNSSAI lists in the CU-CP and CU-UP components. To this end, Figure 73 represents the involved software components, where we highlight the CU Orchestration and Management component (CUOM) that is charge of managing the CU related components in dRAX, thus hiding the complexity² derived from O-RAN disaggregation to the higher layers of the management stack. The CUOM component was originally developed in the 5G-CLARITY project [8] and has been extended in Affordable5G to support configuration of SNSSAIs.

² e.g. need to maintain a mapping between CU-UPs and CU-CPs

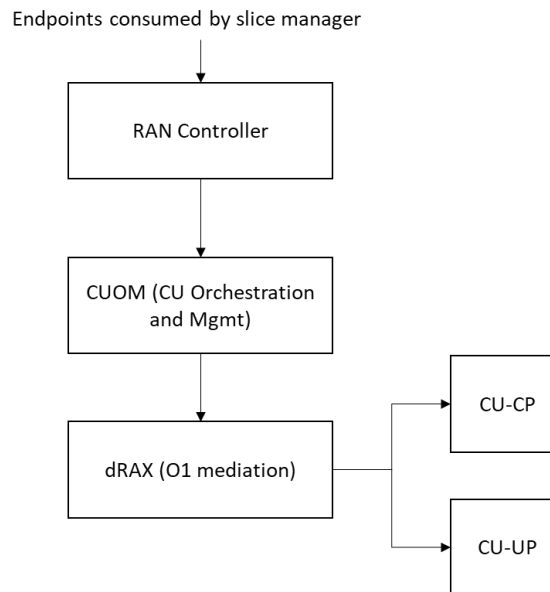


Figure 73: Software components involved in slice provisioning

Figure 74 depicts the workflow followed by CUOM to deploy a new SNSSAI, where we can see how the CUOM behaviour upon receiving a request to deploy a new slice depends on whether an existing PLMNID to serve that slice is already configured or not.

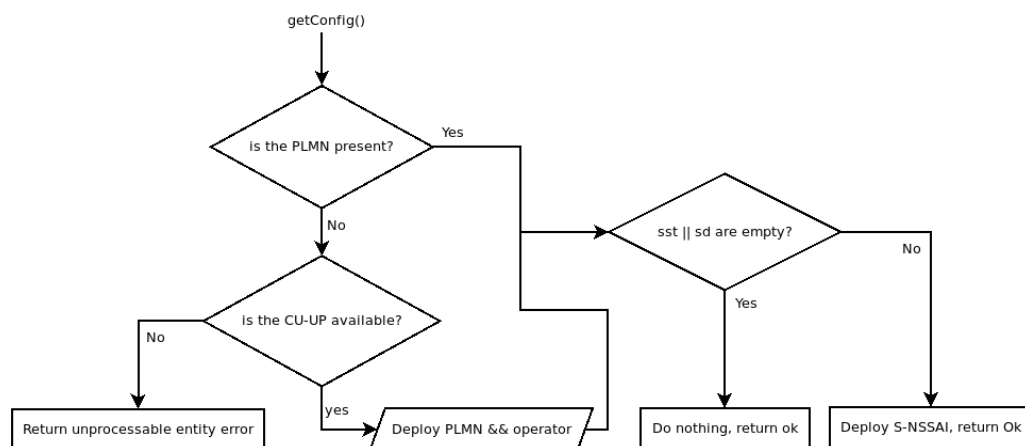


Figure 74: Workflow followed by CUOM to deploy a new SNSSAI

We describe the behaviour of CUOM through an example where we provision three different slices.

- First, we deploy a default slice where no specific SST/SD pair is provided. This is shown in Figure 75, where we can see that deploying a new operator with PLMNID 00109 requires adding the IP address of the corresponding AMF in the CU-CP (shown in the right of Figure 75), and adding PLMNID 00109 and a default SST/SD, since no specific SST/SD is provided.
- Second, we deploy two specific SNSSAIs within the exiting PLMNID 00109. In this case the CU-CP does not need to be reconfigured, and only the CU-UP is reconfigured to add the corresponding SST/SD pairs to the list. Depicts the NETCONF RPCs CUOM is calling to add the corresponding SST/SD pairs, and Figure 76 depicts the resulting configuration with of CU-UP with the default and two added SST/SD pairs.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <edit-config>
    <target>
      <running/>
    </target>
    <default-operation>none</default-operation>
    <config>
      <gnb-cu-up xmlns="http://accelleran.com/ns/yang/accelleran-gnb-cu-up">
        <supported-plmn-slices
          xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
          nc:operation="create">
            <plmn-id>00109</plmn-id>
            <s-nssai nc:operation="create">
              <sst>embb</sst>
              <sd>16777215</sd>
            </s-nssai>
          </supported-plmn-slices>
        </gnb-cu-up>
      </config>
    </edit-config>
  </rpc>]]>]]>
```

Adding PLMNID and default SST/SD to CU-UP lists

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <edit-config>
    <target>
      <running/>
    </target>
    <default-operation>none</default-operation>
    <config>
      <gnb-cu-cp
        xmlns="http://accelleran.com/ns/yang/accelleran-gnb-cu-cp">
          <operator xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
            nc:operation="delete">
            <operator-id>192.168.100.100</operator-id>
          </operator>
        </gnb-cu-cp>
      </config>
    </edit-config>
  </rpc>]]>]]>
```

Adding operator (AMF IP@) to CU-CP

Figure 75: Provisioning of a new operator in CU-UP and CU-CP

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <edit-config>
    <target>
      <running/>
    </target>
    <default-operation>none</default-operation>
    <config>
      <gnb-cu-up xmlns="http://accelleran.com/ns/yang/accelleran-gnb-cu-up">
        <supported-plmn-slices
          xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
            <plmn-id>00109</plmn-id>
            <s-nssai nc:operation="create">
              <sst>embb</sst>
              <sd>1234567</sd>
            </s-nssai>
          </supported-plmn-slices>
        </gnb-cu-up>
      </config>
    </edit-config>
  </rpc>]]>]]>
```

Adding second SST=embb, SD=1234567 to CU-UP

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <edit-config>
    <target>
      <running/>
    </target>
    <default-operation>none</default-operation>
    <config>
      <gnb-cu-up
        xmlns="http://accelleran.com/ns/yang/accelleran-gnb-cu-up">
          <supported-plmn-slices
            xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
              <plmn-id>00109</plmn-id>
              <s-nssai nc:operation="create">
                <sst>embb</sst>
                <sd>12345678</sd>
              </s-nssai>
            </supported-plmn-slices>
          </gnb-cu-up>
        </config>
      </edit-config>
    </rpc>]]>]]>
```

Adding third SST=embb, SD=12345678 to CU-UP

Figure 76: Provisioning of new slices under existing operator in CU-UP


```

<gnb-cu-up xmlns="http://accelleran.com/ns/yang/accelleran-gnb-cu-up">
  <gnb-cu-up-name>accelleran-cu-up-1</gnb-cu-up-name>
  <gnb-cu-up-id>1</gnb-cu-up-id>
  <admin-state>unlocked</admin-state>
  <up-integrity-protection-enabled>false</up-integrity-protection-enabled>
  <up-ciphering-enabled>true</up-ciphering-enabled>
  <e1-link-policy>
    <sctp-policy>
      <in-streams>1</in-streams>
      <out-streams>1</out-streams>
    </sctp-policy>
  </e1-link-policy>
  <e1-link>
    <dest-address>192.168.100.153</dest-address>
  </e1-link>
  <supported-plmn-slices>
    <plmn-id>00109</plmn-id>
    <s-nssai>
      <sst>emb</sst>
      <sd>16777215</sd>
    </s-nssai>
    <s-nssai>
      <sst>emb</sst>
      <sd>1234567</sd>
    </s-nssai>
    <s-nssai>
      <sst>emb</sst>
      <sd>12345678</sd>
    </s-nssai>
  </supported-plmn-slices>
</gnb-cu-up>

```

Final configuration in CU-UP with three SNSSAIs

Figure 77: Resulting configuration in CU-UP

Slice Manager integration with RAN Controller

Through the representation of a set of Radio Access Network devices controlled by a RAN Controller that exposes a northbound API, the Slice Manager can control different RAN infrastructures. RAN Infrastructure controller admits posting data regarding a controller including its name, location, and authentication schema. Each infrastructure has a defined topology that can be queried to list all the wireless hardware elements controlled by the system, type of interfaces and capabilities, WiFi, LTE, etc.

The slice manager API allows the orchestrator to request a Radio Chunk to select a set of interfaces and links belonging to the topology, grouping them logically with the aim of deploying some service in the future. Taking a radio chunk as a starting point, slice manager's radio service API allows the orchestrator to call a POST endpoint in order to deploy a wireless connectivity service for end user devices, creating the slice.

The Figure 78 shows the response from the Slice Manager after it has gathered all the information from the RAN Controller and the involved infrastructure.

```
{
  "name": "test-SNSSAI1",
  "user_id": "6397854a46ebfc19e96a7591",
  "id": "639838bb46ebfc466bfff5664",
  "radio_chunk": {
    "id": "639838b246ebfc466bfff5661",
    "ran_infrastructure_id": "6397905946ebfc2f95ddc3f9",
    "chunk_topology": {
      "selectedPhys": [
        {
          "id": "76344247-5896-4359-9b96-227e6cd3322f",
          "name": "primaryPLMN",
          "type": "ACCELLERAN_CELL",
          "config": {
            "earfcn": 41690,
            "phyCellId": 512,
            "refSignalPower": -7
          }
        }
      ]
    }
  },
  "radio_service": {
    "id": "639838b846ebfc466bfff5663",
    "coreAddress": "172.16.0.10",
    "corePort": 8188,
    "plmn_id": "00110",
  }
}
```

Figure 78: Slice Manager's response

The log in Figure 79 shows how the slice configuration is created after the RAN Controller confirms the two parameters included in the SNSSAI: the SST and the SD pair. The SST defines the Slice/Service Type, showing basically what the expected behaviour of the slice is in terms of special features or types. The standardized values of the SNSSAI includes the codes for defining the three main traffic-types of 5G: eMBB, URLLC, and mMTC. On the other hand, the SD acts as the Slice Differentiator in cases where there are more than one slices with the same main traffic type, therefore differentiating the slices from the same Service Type.

```
2022-12-13 09:32:49 INFO business.radio_chunk Chunk topology successfully validated
2022-12-13 09:32:49 INFO clients.ran_controller Trying to create radio chunk with topology {'selectedPhys': [{'id': 'd7cdf823-6a60-4cf7-b1ee-1f2062634d38', 'name': 'Cell-1', 'type': 'ACCELLERAN_CELL', 'config': {'earfcn': 41690, 'phyCellId': 512, 'refSignalPower': -7}}, {'id': 'e3fcc74a-73ea-4644-9b7f-65743a82e0bd', 'name': 'eth0', 'type': 'WIRED_ROOT', 'config': None}]}
2022-12-13 09:32:49 INFO clients.ran_controller RADIO_CHUNK POST ON RAN CONTROLLER ---
2022-12-13 09:32:49 INFO clients.ran_controller RADIO_CHUNK URL http://84.88.36.61:8008/chunkController/chunk
2022-12-13 09:32:49 INFO clients.ran_controller RADIO_CHUNK POST ON RAN CONTROLLER ONLY IDS ---
2022-12-13 09:32:49 INFO clients.ran_controller POST BODY {'name': 'test-SNSSAI1', 'selectedPhys': ['d7cdf823-6a60-4cf7-b1ee-1f2062634d38', 'e3fcc74a-73ea-4644-9b7f-65743a82e0bd'], 'selectedLinks': []}
2022-12-13 09:32:49 INFO clients.ran_controller {'id': '59318f56-d97d-47f0-b680-f87ee2ed432d'}
2022-12-13 09:32:50 INFO api.ran_slice_subnet radio_chunk data:
2022-12-13 09:32:50 INFO clients.ran_controller Radio Service URL http://84.88.36.61:8008/chunkController/chunk/59318f56-d97d-47f0-b680-f87ee2ed432d/service
2022-12-13 09:32:50 INFO clients.ran_controller RADIO_SERVICE POST ON RAN CONTROLLER ---
2022-12-13 09:32:50 INFO clients.ran_controller RADIO_SERVICE URL http://84.88.36.61:8008/chunkController/chunk/59318f56-d97d-47f0-b680-f87ee2ed432d/service
2022-12-13 09:32:54 INFO clients.ran_controller POST BODY {'selectedPhys': {'id': '76344247-5896-4359-9b96-227e6cd3322f', 'name': 'primaryPLMN', 'type': 'ACCELLERAN_CELL', 'config': {'earfcn': 41690, 'phyCellId': 512, 'refSignalPower': -7}}, 'vlanId': 1520, 'serviceConfig': {'cellularConfig': {'plmnId': '00110', 'acceleratedConfig': {'coreIpAddress': '172.16.0.10', 'corePort': 8188, 'vlanCore': 1588}, 'snssaiList': [{'sst': 1, 'sd': '12345679']}}}}
2022-12-13 09:32:54 INFO clients.ran_controller {'id': 'd2cc1d1b-c477-4f94-822c-206671740b7c'}
2022-12-13 09:32:54 INFO business.ran_slice_subnet RESPONSE:
2022-12-13 09:32:54 INFO business.ran_slice_subnet {'id': 'd2cc1d1b-c477-4f94-822c-206671740b7c'}
127.0.0.1 - - [13/Dec/2022 09:33:01] "POST /api/v1.0/ran_infrastructure/6397905946ebfc2f95ddc3f9/ran_slice_subnet HTTP/1.1" 200 -
```

Figure 79: Slice configuration log after the slice creation

6.4.3 Configurable parameters and KPIs

MCX System KPIs

The MCX system KPIs that are going to be evaluated within MCS pilot are mainly focused on MCX call performance and are the following:

Round Trip Time (RTT)

3GPP defines RTT as the time required from the moment the end-user transmits a piece of information until it reaches the application or service provider. In this case, the time required to transmit a packet of information between two previously defined nodes, to process that information at the receiving node, and to transfer a status acknowledgement back to the transmitting node, regardless of whether the message transmission and the acknowledge are successful or not is considered as shown in Figure 80.

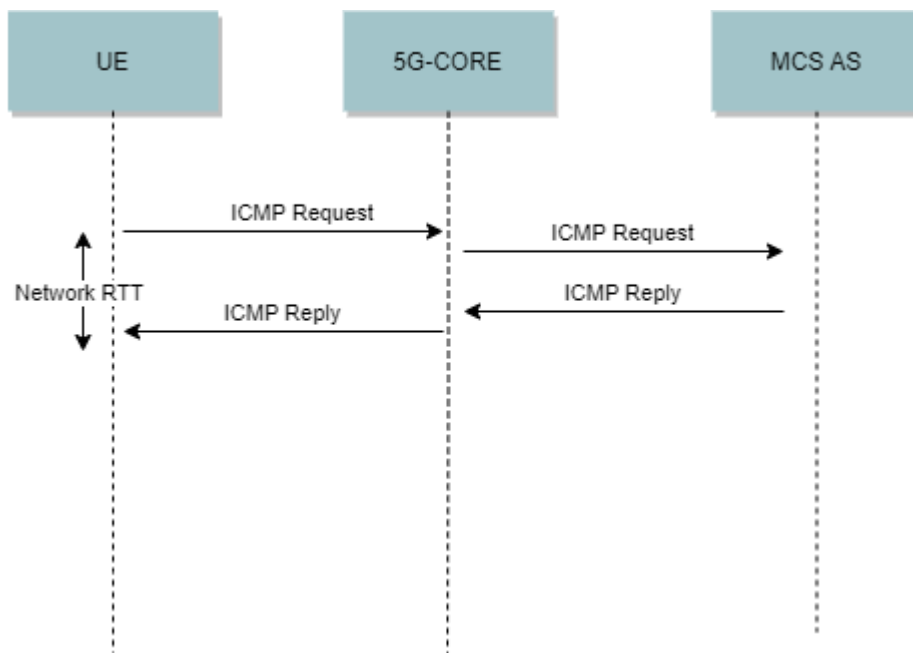


Figure 80: RTT MCX system KPI call-flow

MCPTT Access Time (KPI1)

The MCPTT access time is defined as the time between when an MCPTT User requests to speak and when this user gets a signal to start speaking. This time does not include confirmations from receiving users.

The standard definition leaves two different explanations or interpretations of KPI1. On the one hand, KPI1a refers to an access time in which the call setup takes place. On the other hand, for KPI1b, once the call is established, the access or token time as the time between a PTT press event, the token is requested, and the access is granted. In the figure below a diagram of the second interpretation of KPI1, which is the one to be measured, can be seen.

This definition is directly associated with the definition of KPI 1 provided by 3GPP TS 22.179 22 [9].

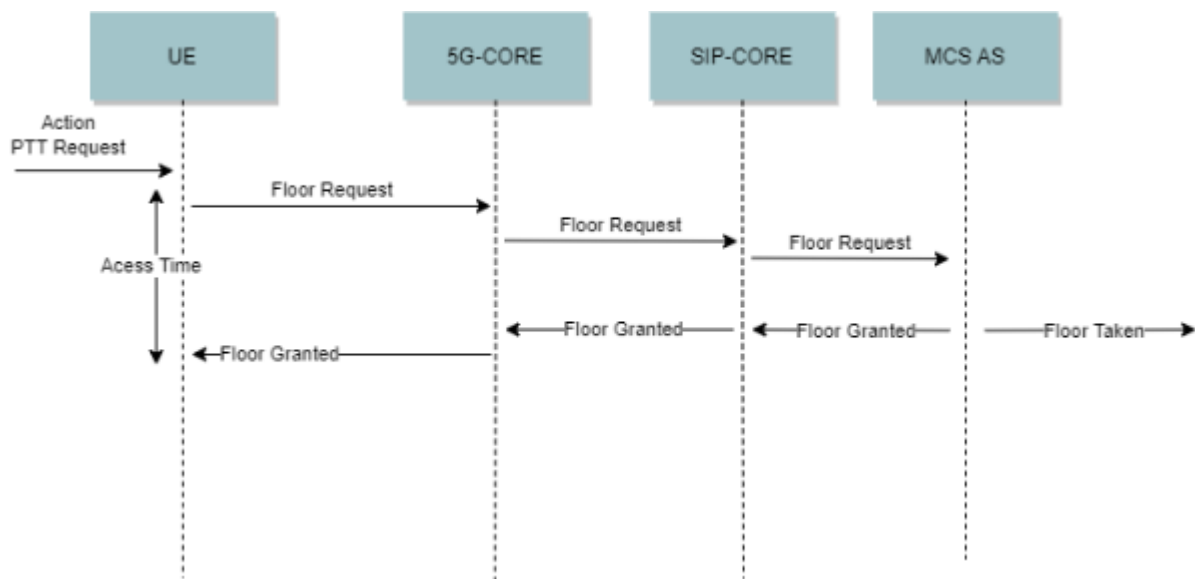


Figure 81: MCPTT Access Time (KPI1) call-flow

MCPTT E2E Access Time (KPI2)

End-to-end MCPTT Access time is defined as the time between when an MCPTT User requests to speak and when this user gets a signal to start speaking, including MCPTT call establishment and acknowledgment from receiving user(s) before voice can be transmitted. A typical case for the End-to-end MCPTT Access time including acknowledgement is an MCPTT Private Call request where the receiving user's client accepts and joins the call. This KPI is associated with the definition in 3GPP TS 22.179 as well [9].

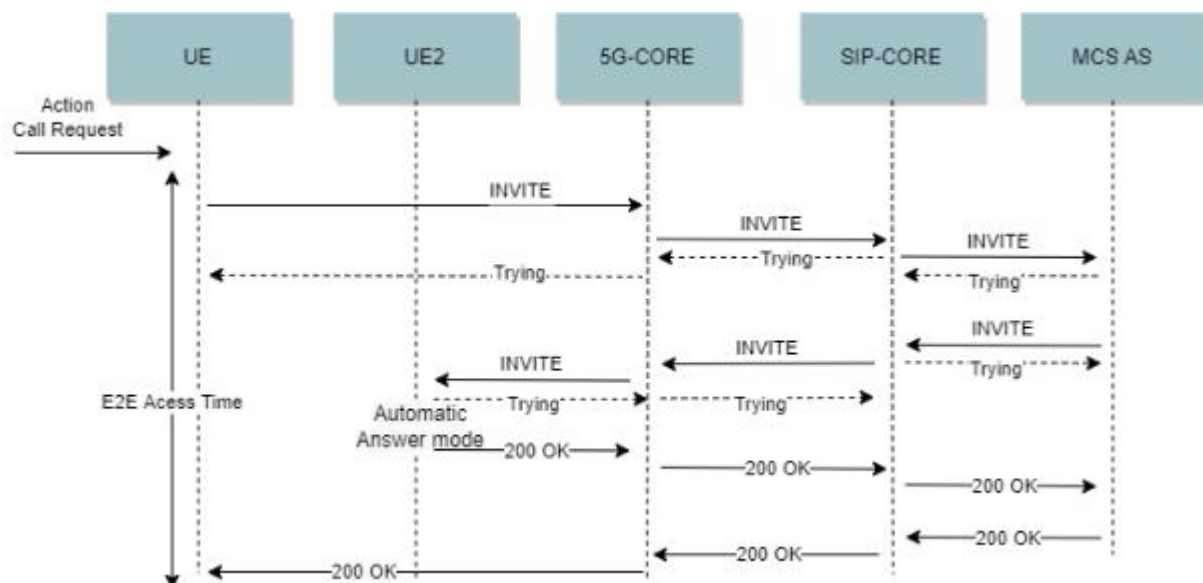


Figure 82: MCPTT E2E Access Time call-flow

Metrics, monitoring and predictions

MCX service-related monitoring metrics are going to determine the actions to be performed for MCS pilot scenario 1. This metrics are related with the number of users registered in the MCX

system and the activity they are performing. The NWDAF and associated AI/ML module predict changes in the MCS service behaviour and communicates the resulting alarms to the Orchestrator. The metrics that will be monitored for MCS pilot scenario 1 are the following:

- Number of registered users. The users registered in the MCX service, independently of their current activity (e.g. in a call, or in stand-by).
- Number of active private calls. Number of simultaneous private calls that are taking place concurrently in the MCX service.
- Number of active group calls. Number of simultaneous group calls that are taking place concurrently in the MCX service (note that while private calls always imply only 2 users, group calls can include any number of users higher or equal than 2).

The above metrics are provided to the NWDAF Prometheus continuously at regular time instances (sampling period is 30 sec). The NWDAF includes a pre-trained AI/ML model to predict the eventual load increase in the service (i.e. combination of registered users and simultaneous group calls within the registered users). For this purpose, the pre-trained ML model in tensorflow format takes 9 variables as inputs, namely the Number of registered users, the number of active private calls and the number of active group calls for the three previous time instance, i.e. $t-3$, $t-2$ and $t-1$. Based on these inputs, the model has been trained to returns 2 values for the t time instance, namely the "Probability of Overload (%)" and "Probability of No Overload (%)", the latter being complementary to the former. It should be noted that the ML model has been trained with realistic simulated datasets, based on the total number of registered users that will be active (4 UEs are foreseen to be available for the final demo).

Figure 83 depicts the dimensionality of the Deep Neural Network that is used for overload prediction in the MCS scenario. The ML model can be inferred to predict the probability of overload occurrence at time t , based on the 3 previous time samples of all 3 metrics. Therefore, for a given 9-valued input, the model outputs a single value 0 (if the probability of overload < 50%) or 1 (otherwise). In case that overload is predicted by the AI/ML model, the Orchestrator will trigger the required actions (service scalability) to face an eventual load increase in the service.

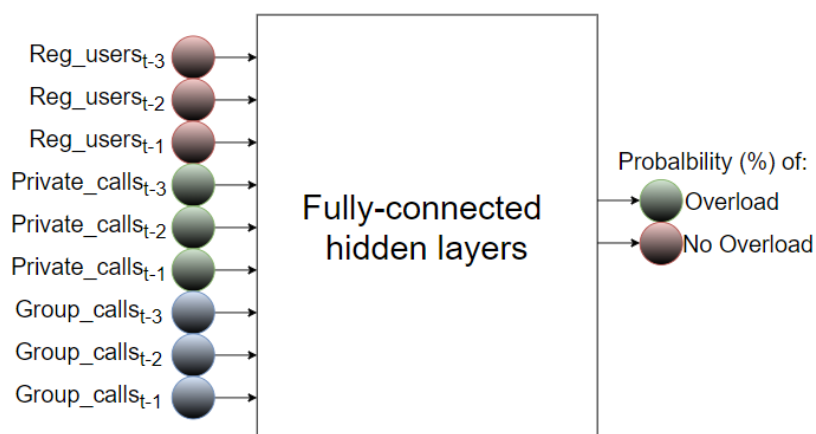


Figure 83 : Dimensionality of the Deep Neural Network trained to predict the MCS service overload

6.4.4 Test cases and results

Functional test cases

The functional test cases are the steps accomplished within each one of the scenarios described and depicted in 6.4.1.

Scenario 1

As described in Figure 2, these are the steps leading to the MCX service scaling test case. The accomplishment of each one of these steps will validate the MCS scaling scenario.

Table 3: Scenario 1 related functional test case sequence

Step	Origin component	Destination component	Item	By means of
1	MCX	Telemetry system	Metrics	Prometheus client/server
2	NWDAF	NWDAF	Prediction	ML/AI
3	NWDAF	Orchestrator	Alarms	Prometheus client/server
4	Orchestrator	MCX system	MCX pods reinstantiation	Kubernetes
5	MCX system	MCX system	Data integrity	Load balancer

For demonstration purposes of the MCS scenario, the demo was executed in the framework of Affordable5G project and involved the use of 4 UEs with NEM MCS application installed within.

In the figures below the deployment of each service is shown in the Kubernetes (k8s) cluster made available by Nearby Computing. Figure 85 and Figure 87 below show the deployed NWDAF and MCX system pods needed to carry out the experimentation.

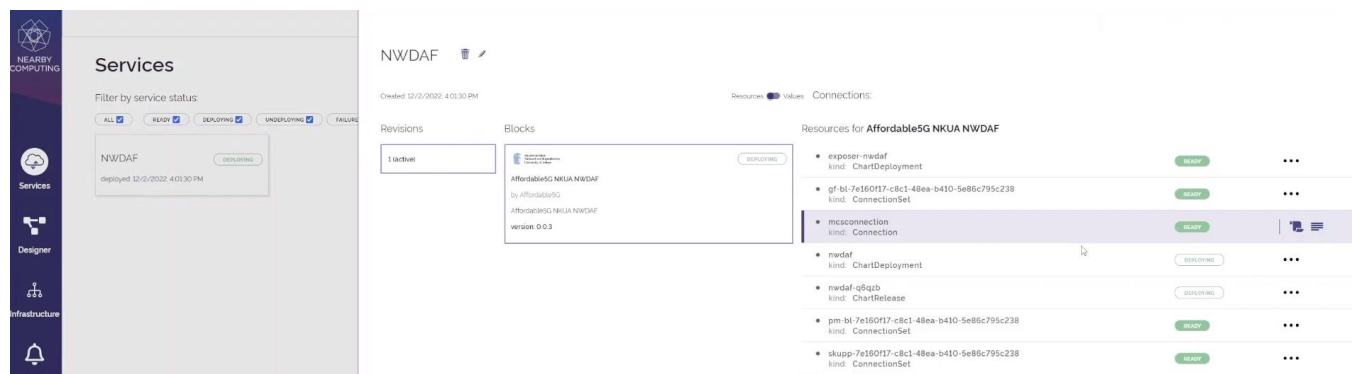
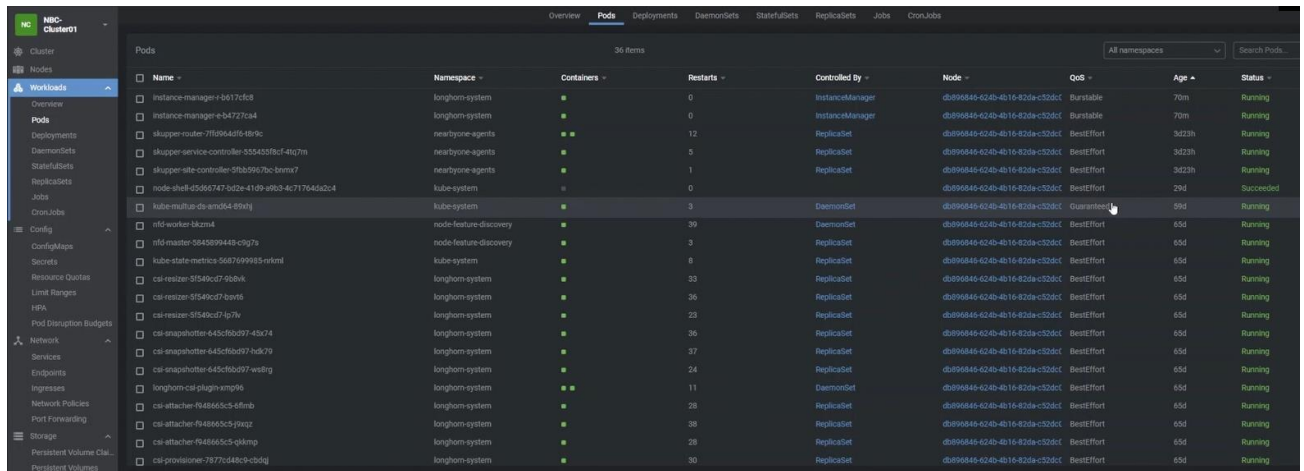
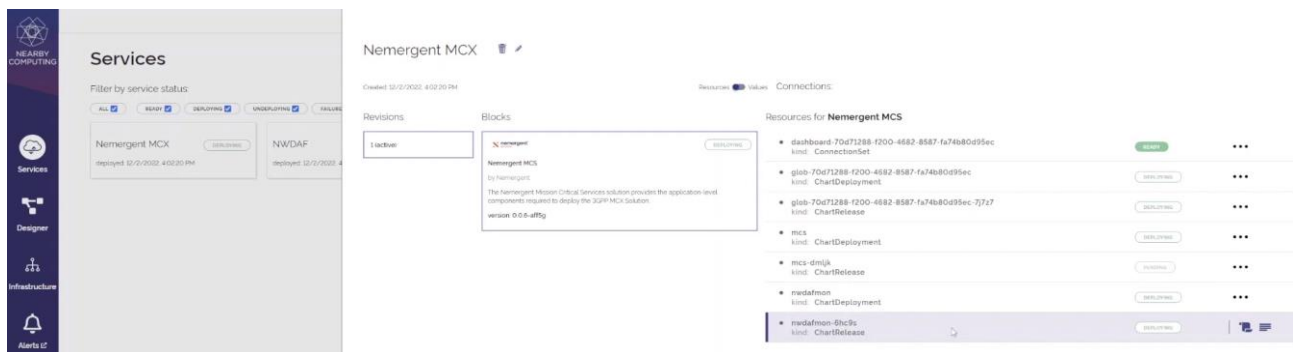


Figure 84: NWDAF deployment through the designer tab in Nearby's k8s cluster



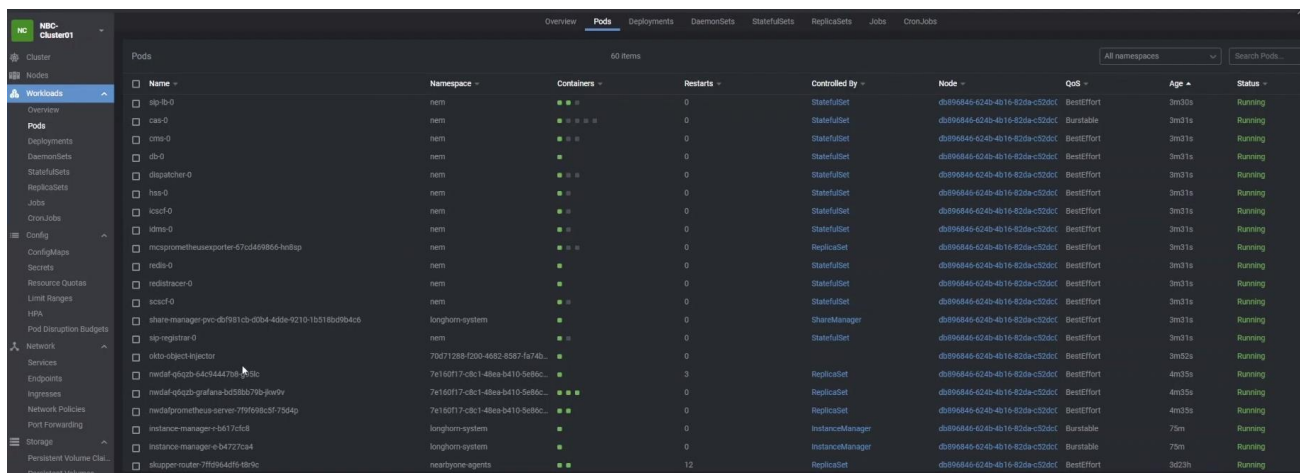
Name	Namespace	Containers	Restarts	Controlled By	Node	QoS	Age	Status
instance-manager-e-b617cd8	longhorn-system		0	InstanceManager	db96846-624b-4b16-82da-c52dc1	Burstable	70m	Running
instance-manager-e-b4727c4d	longhorn-system		0	InstanceManager	db96846-624b-4b16-82da-c52dc1	Burstable	70m	Running
skupper-router-7f5964d9-d8fc	nearbyone-agents		12	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3d23h	Running
skupper-service-controller-555435bfc4-4tq7m	nearbyone-agents		5	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3d23h	Running
skupper-site-controller-9fb5967bc-bvms7	nearbyone-agents		1	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3d23h	Running
node-shell-d566747-bt2e-41d9-v9b3-4c71764d2c4	kube-system		0		db96846-624b-4b16-82da-c52dc1	BestEffort	29d	Succeeded
kube-multus-daemonset-69nqj	kube-system		3	DaemonSet	db96846-624b-4b16-82da-c52dc1	Guaranteed	59d	Running
nfd-worker-bkzm4	node-feature-discovery		39	DaemonSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
nfd-master-584599449-c9g7e	node-feature-discovery		3	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
kube-state-metrics-568769993-vkml	kube-system		8	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-resizer-9f549cd7-bvms	longhorn-system		33	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-resizer-9f549cd7-bvms	longhorn-system		26	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-resizer-9f549cd7-bvms	longhorn-system		33	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-snapshotter-645cb0d97-43v74	longhorn-system		36	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-snapshotter-645cb0d97-43v74	longhorn-system		37	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-snapshotter-645cb0d97-43v74	longhorn-system		24	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
longhorn-csi-plugin-xmp96	longhorn-system		11	DaemonSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-attacher-94d665c5-4fmb	longhorn-system		28	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-attacher-94d665c5-4fmb	longhorn-system		28	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-attacher-94d665c5-4fmb	longhorn-system		28	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running
csi-provisioner-7877c48c9-cbdg	longhorn-system		30	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	65d	Running

Figure 85: NWDAF component's list of deployed pods



The image shows the 'Services' tab in the Nearby Computing Designer. It displays the deployment of 'Nemergent MCX' and 'NWDAF'. The 'Nemergent MCX' deployment is highlighted, showing its details and the resources it uses. The 'Resources for Nemergent MCS' section lists various components and their configurations.

Figure 86: MCX deployment through the designer tab in Nearby's k8s cluster



Name	Namespace	Containers	Restarts	Controlled By	Node	QoS	Age	Status
slp-lb-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m30s	Running
cas-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	Burstable	3m31s	Running
oms-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
db-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
dispatcher-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
has-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
kscf-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
kids-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
mcsprometheus-exporter-67d459865-hs3ap	nem		0	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
redis-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
redistracer-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
sccf-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
share-manager-prc-df931cb-d3d4-4d5e-921b-1b318d99Ac0	longhorn-system		0	ShareManager	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
slp-registrar-0	nem		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m31s	Running
okto-object-injector	70d71288-f200-4682-8587-fa74b0d95sec		0	StatefulSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3m32s	Running
nwdaf-q5qzb-64c9444788-33c	7e160f17-c8c1-48ea-b410-568bc...		3	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	4m35s	Running
nwdaf-q5qzb-grafana-b53b679b-jwlvv	7e160f17-c8c1-48ea-b410-568bc...		0	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	4m35s	Running
medafprometheus-server-7f9f99c5f-754p	7e160f17-c8c1-48ea-b410-568bc...		0	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	4m35s	Running
instance-manager-e-b617cd8	longhorn-system		0	InstanceManager	db96846-624b-4b16-82da-c52dc1	Burstable	75m	Running
instance-manager-e-b4727c4d	longhorn-system		0	InstanceManager	db96846-624b-4b16-82da-c52dc1	Burstable	75m	Running
skupper-router-7f5964d9-d8fc	nearbyone-agents		12	ReplicaSet	db96846-624b-4b16-82da-c52dc1	BestEffort	3d23h	Running

Figure 87: MCX system's list of deployed pods

After having deployed the required components, Mission Critical users' registration, emergency private calls and groups calls have been triggered as shown in the pictures below.

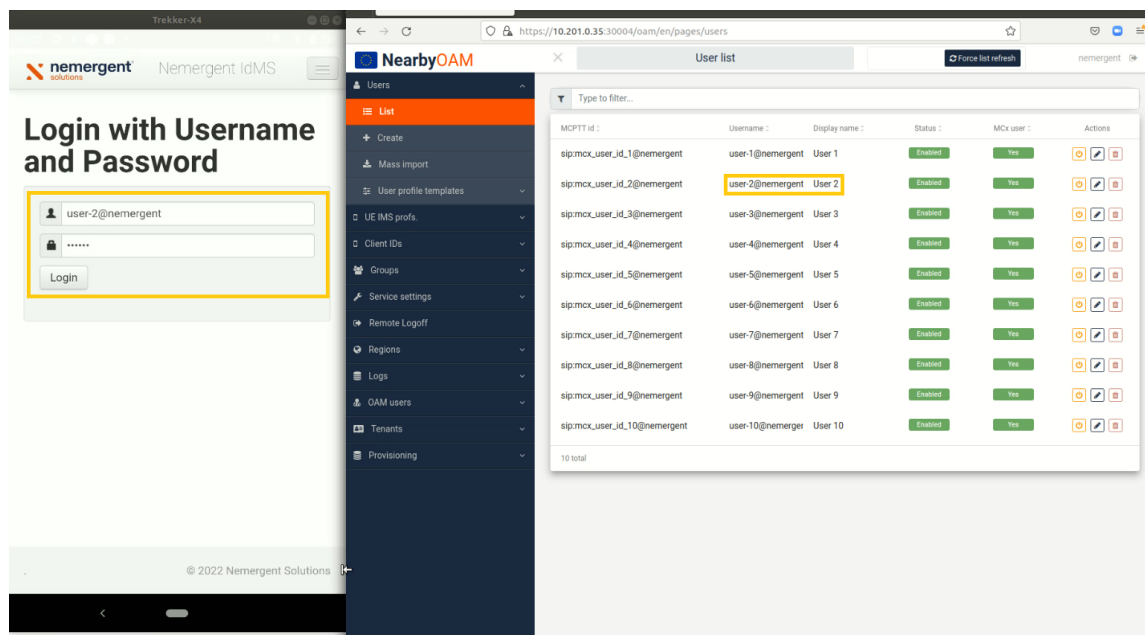


Figure 88: MCX user registration procedure

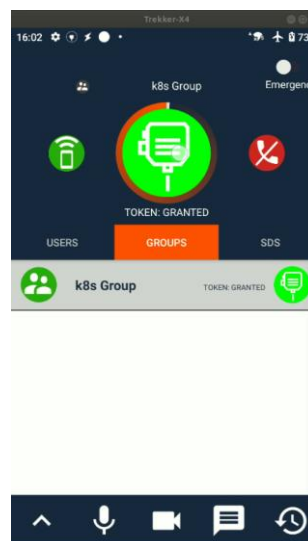


Figure 89: Ongoing MCX group call

Depending on the number of registered users, the number of active private calls and the number of active group calls, the NWDAF module has elaborated predictions in order to determine whether the MCX system required to be capacitatively resized, by means of the orchestrator, instantiating new MCX pods in the K8s cluster to give response to the load increase.

In this case, the overload condition is defined as the moment when the 4th UE tries to join an active group call. The ML model has been pre-trained with simulated time-series measurements of registered users, active private calls and active group calls, sampled every 30 sec, while the epoch duration was 10 min, and the complete simulated dataset duration was 1 month (4320 epochs). The training accuracy of the ML model was additionally tested using validation samples for 1 week (i.e. samples that were not seen during the training). The resulting confusion matrix is depicted in Figure 90, where the false positive predictions

(overload prediction when an overload condition does not exist) are 3% and the false negative predictions (no overload prediction when an overload condition exists) are 4% of the validation samples.

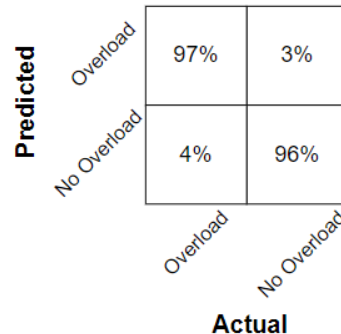


Figure 90 : Confusion matrix regarding the validation accuracy of the ML model training process

Once the NWDAF service is ready, the Prometheus can be accessed to plot the variables that are sent through the MCX NEM application (number of registered users, number of active group call and number of active private calls), as shown in Figure 91. In addition, the aforementioned metrics from the last 3-time instances are processed by the NWDAF, following the logic of the AI/ML module and are used to predict the overload condition (categorical variable that takes the value of 1 if overload is predicted or 0 otherwise). As seen from Figure 91, the number of registered users is initially zero and switched to 1 at a selected time instance, while the prediction of the ML model is zero.

In order to verify the prediction accuracy of the NWDAF ML model and validate the scaling capabilities of the Orchestrator, the number of users is increased to 3, registering 2 more users in the MCX application, while also initiating a group call between them. These developments are depicted in Figure 92 and designate that the overload condition is impending, i.e. will happen when the 4th user is registered in the MCX application.

Moreover, Figure 91 illustrates the model prediction outputs for each time instance (i.e. every 30 sec), using the 9-valued input data of the previous 3 time instances.

The time variation of these variables can be shown in Figure 92. Evidently, the prediction result of the ML model that is associated with the NWDAF switches to one, indicating that an overload condition will occur in an upcoming time instance. The overload condition is shortly verified, since the Number of registered users changes to 4, while the number of active group calls remains one (specifications of the overload condition in this demo).

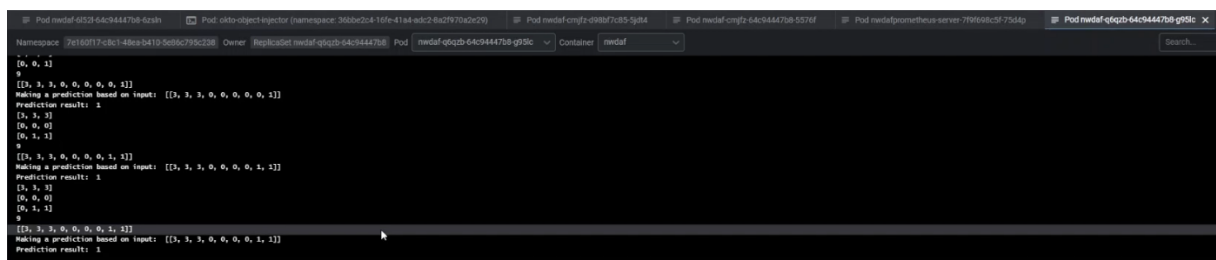


Figure 91 : Log showing the model prediction outputs for given 9-valued inputs

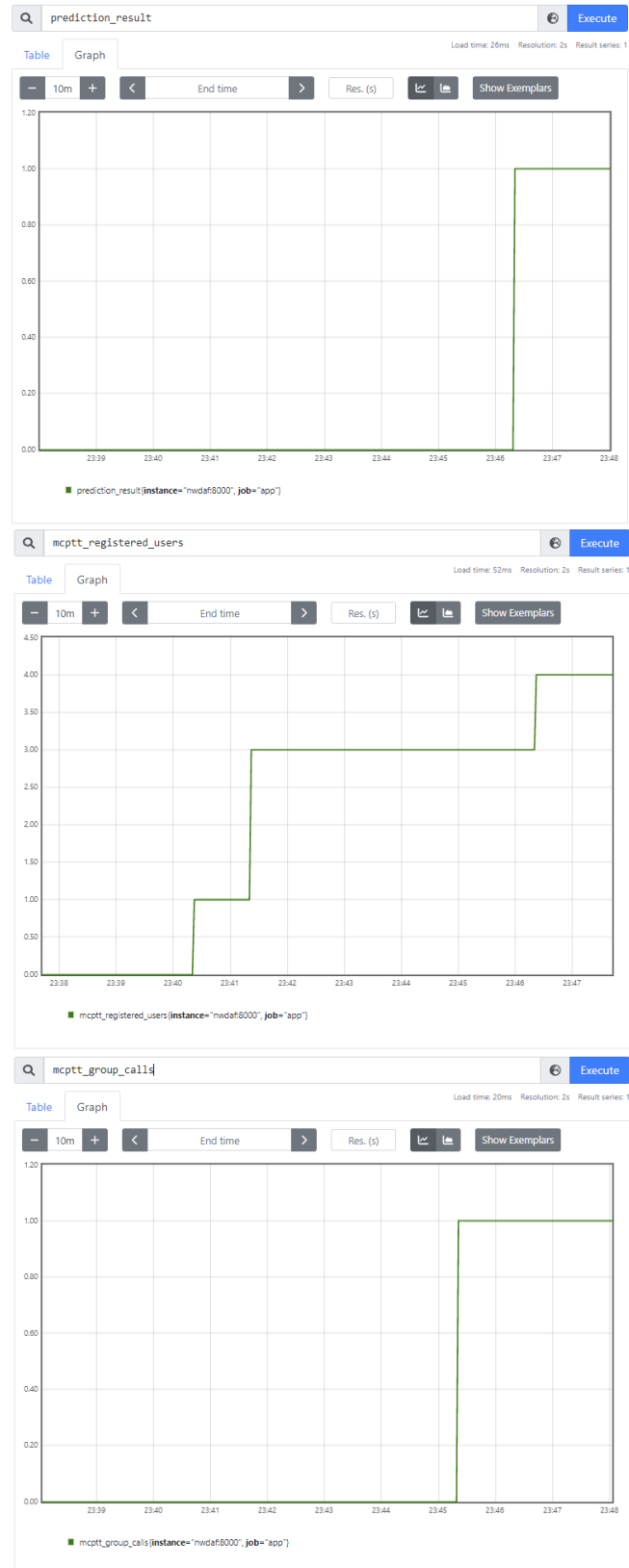


Figure 92: Prometheus illustrating the prediction results in the upper plot (switching from 0 to 1), the number of registered users in the middle plot (increasing from 0 to 4) and the number of active group calls in the lower plot (from 0 to 1)

As the prediction result value is equal to one, an overload situation is being predicted and in consequence, the orchestrator will escalate the required MCX service pods accordingly to face the incoming overload situation.

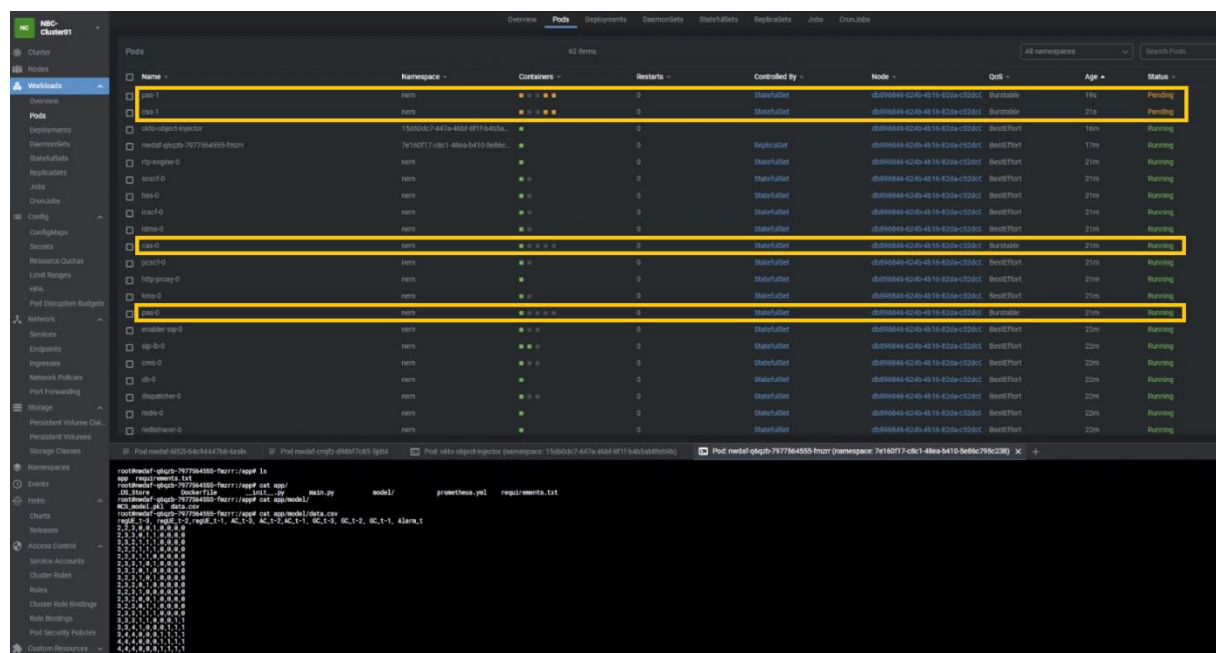
Manifest: mcs-bxww7

```

83 image: nenergent-db:nbc-no-huge-0.0.1
84 exporter:
85   enabled: true
86   prometheusExporter:
87     enabled: true
88   prometheusIntegration:
89     enabled: false
90 imagePullSecrets: aws1mf10GhZ1JogewoJCS3yD4Dpc3RyeS5u2Mfym1j3z1wdXKpbecu29t1JogewoJCKG
91 imageRegistry: registry.nearbycomputing.com/nenergent/
92 loadBalancer:
93   enabled: false
94 mcsDispatch:
95   oamDisplayName: Nearby
96   oamFlagCode: EU
97   provisioningTransport: tcp
98 nodePort:
99   mcsPort: 30001
100   dispatcherPort: 30002
101   enabled: true
102   httpProxyPort: 30003
103   httpProxySslPort: 30004
104   lmsPort: 30005
105   pcsPort: 5000
106   sipRegistrar: 30006
107   sipRegistrarSsl: 30007
108
109 cas:
110   replicas: 2
111 pas:
112   replicas: 2
113   #connectionPrediction: "1"
114   version: 2.0.1
115 status:
116   installedChartVersion: 2.0.1
117   phase: Deploying

```

Figure 93: MCX "cas" and "pas" pods are being reinstantiated due to the prediction received by the orchestrator



Name	Namespace	Containers	Restarts	Controlled by	Node	QoS	Age	Status
cas-1	mcs	cas	0	StatefulSet	d099999-42-40-40-19-42da-c520c	BestEffort	11m	Pending
cas-0	mcs	cas	0	StatefulSet	d099999-42-40-40-19-42da-c520c	BestEffort	21m	Pending
pas-1	mcs	pas	0	StatefulSet	d099999-42-40-40-19-42da-c520c	BestEffort	11m	Running
pas-0	mcs	pas	0	StatefulSet	d099999-42-40-40-19-42da-c520c	BestEffort	21m	Running

Figure 94: MCX system's running pods cas-0, pas-0 and pending cas-1, pas-1

The time elapsed between the orchestrator receives the prediction, it processes it and the required reinstantiation action takes place has been measured and it is shown in Table 3. They refer to the step 3 to 5 mentioned in Table 2.

*Table 4: KPI results from Scenario 1 -
Time elapsed between the prediction is received and the MCX service is reinstated*

KPI Definition:	Time between external component (Prometheus) receives prediction, orchestrator reacts and k8s reinstatement is completed
Samples	16,00
Avg	87s
Min	75s
Max	99s
Median	87s
Stdev	6s

Several iterations have been made to obtain the data above and face to an emergency scenario, the values satisfactorily meet the expectations face to an imminent overload situation in which the service needs to readjust to cope with the increasing number of users registered as well as the number of calls.

Scenario 2

As described in Figure 3, these are the required steps to accomplish in order to satisfactorily create a dedicated slicing for emergency service bodies responding to the events described in Scenario 2.

Table 5: Scenario 2 related functional test case sequence

Step	Origin component	Destination component	Item	By means of
1	Emergency witnesses	Emergency control Center (CC)	Emergency notification	Call
2	Emergency CC	Orchestrator	Emergency network request	Call
3	Orchestrator	Edge 5GC	2 nd slice activation request	API
4	Orchestrator	MCX Edge System	MCX pods instantiation	Kubernetes

Step 3 is the functional test regarding the interoperability between Orchestrator and Athonet 5GC, and considers the process of activating the second slice, triggered by the orchestrator. This is described in detail in section 6.3.2 and, as shown in Figure 66 and Figure 67, this step was successfully achieved.

Extended scenario 2

As mentioned above and due to the unavailability of the Castellolí premises to perform the test, the multi Point of Presence (PoP) feature on MCX service level has been validated at Nemergent Solution premises.

The ability to completely move a working and serving MCX service from one PoP to another one, switching the traffic of the users from one PoP to another one in the most seamless manner is fundamental in an emergency situation. This way, we could tackle or be responsive in scenarios where infrastructures are buggy and/or sudden errors could pose a threat to the system.

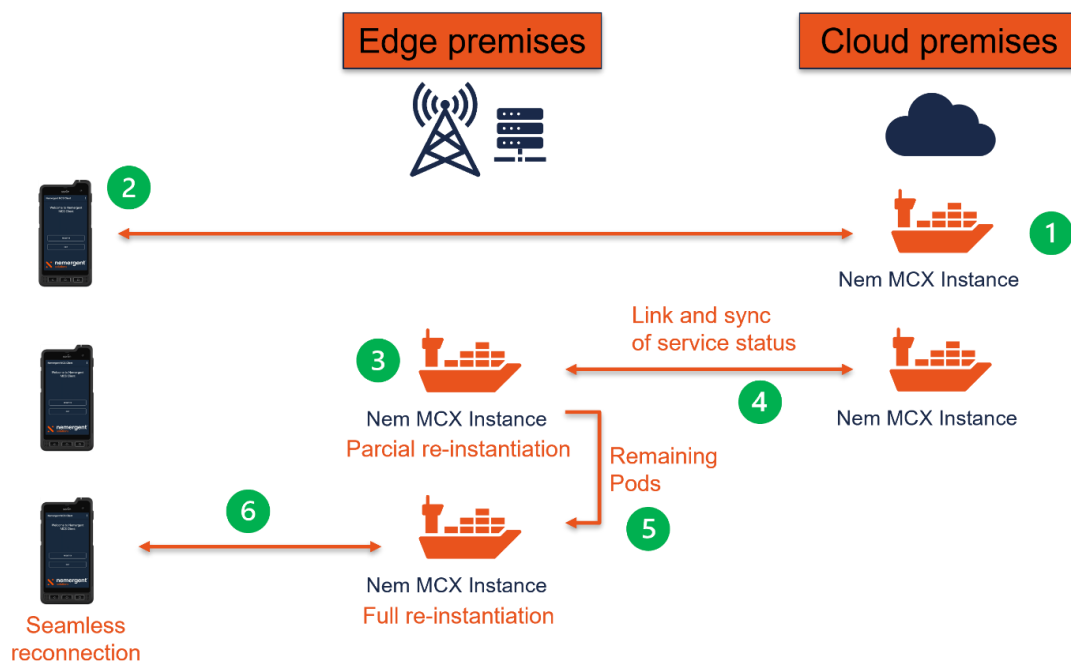


Figure 95: Steps for multi-PoP feature accomplishment

The normal course of action of the multi-PoP feature is summarized in Figure 95. The steps shown in the figure correspond to the following actions:

1. Initial MCX service status: KNF deployment of MCX service working in main (cloud) instance.
2. Initial MCX clients status: clients provisioned in main instance's MCX service are using it.
3. Partial re-instantiation of MCX service in edge instance.
4. Stateful components synchronization with main instance's MCX service for correct service replication.
5. Finish full deployment of MCX service remaining pods in edge instance
6. Clients instructed to start using new MCX service replica deployed in edge instance after seamless reconnection.

Figure 96 shows a detailed diagram of the process followed for validation of the multi-PoP feature, including the interactions that take place between the different elements involved.

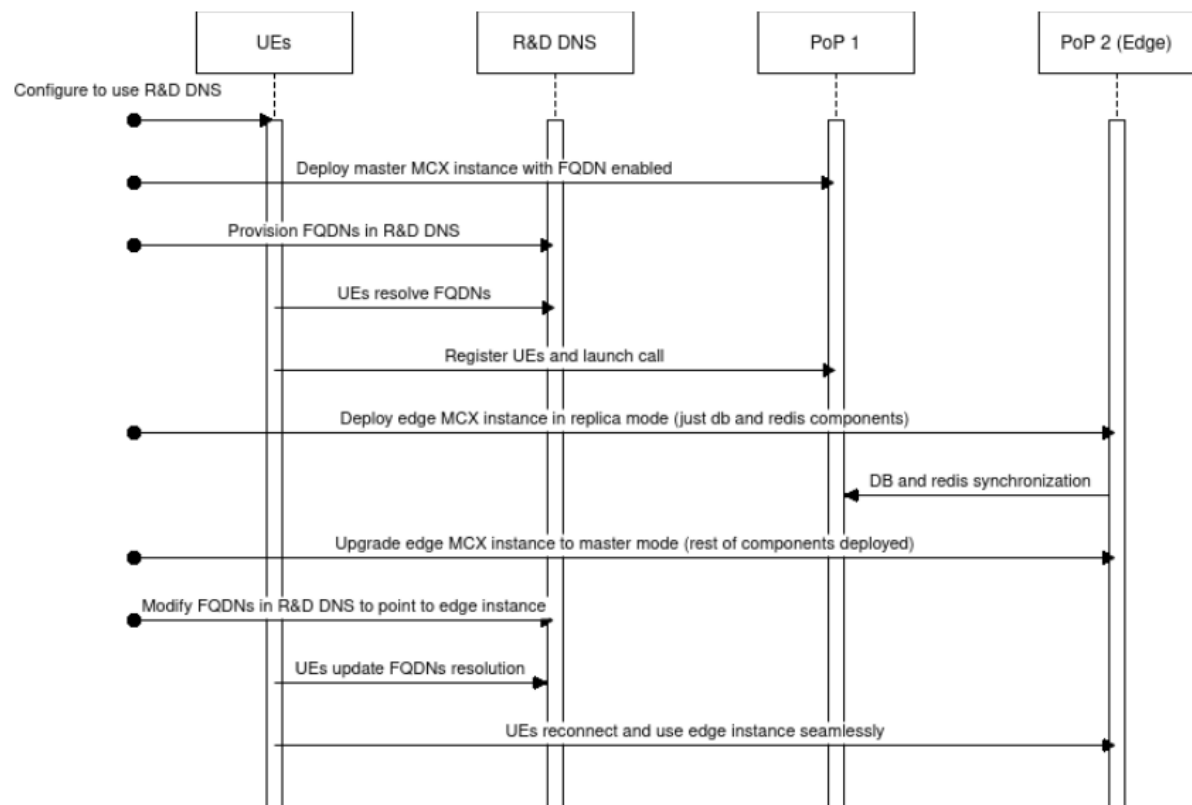


Figure 96: Detailed diagram describing the interaction in multi-PoP validation

Technical test cases

The technical test cases that are listed below correspond to the ones which will allow us to evaluate the MCX system related KPIs.

The necessary pre-requisites and specific setup for the test cases are listed below.

Pre-requisite identifier	Description
PR1	Nemergent MCS Android Client application, installed, provisioned, and configured.
PR2	Nemergent MCS Application Server deployed, configured and running.
PR3	UE access through ADB activated
PR4	Android UEs with 5G connectivity and access to NEM's MCS Application Server.
PR5	5GC network up and running

Setup Identifier	Description
SETUP 1	One Mission Critical Agency-A deployed. At least two Mission Critical clients provisioned: Client-01 and Client-02 .

	At least two different MCS users correctly logged in: Client-01 and Client-02 . A pre-arranged group (Group1) with at least two members affiliated, users: Client-01 and Client-02 .
--	--

Technical test #1: Network RTT

Test case name	Network RTT	Test Case id	Test-04-01
Test purpose	The objective of this test is to measure the average, minimum and maximum network RTT between a mission critical UE and the Application Server.		
Pre-requisites	PR2, PR3, PR4, PR5	Setup	SETUP 1
Test tools			
Components Involvement	All AFFORDABLE 5G Components ³ + Mission Critical Services		
Test sequence	Step 1	Access UE through ADB	
	Step 2	Run network RTT experimental script, which will send multiple ICMP echo request messages, at least 100.	
	Step 3	Obtain the average, minimum, maximum and the confidence interval of the experimental results	
	Step 4	Repeat the experiment several times, at least 50 times.	
Validation Criteria	Network RTT will characterize the delay mainly due to the underlying network and it will set the baseline of the service latency that must be offered by any application services, like MC Communication. The chosen target values are representative of the MC service under test, since the lower target envisages the requisites of this kind of services, where the latency needed is related to the user perception.		
Target Values	Upgradable ≥ 100 ms > Acceptable ≥ 40 ms > Optimal		
Experiment Variations	A: Service instantiated in the CORE B: Service instantiated in the EDGE		

Technical test #2: MCPTT Access Time

Test case name	MCPTT Access Time	Test Case id	Test-04-02
Test purpose	MCPTT access time is defined as the time between when an MCPTT User requests to speak and when this user gets a signal to start speaking and it does not include confirmations from receiving users, as defined by the 3GPP Technical Specification (TS 122 179. Section 6.14).		
Pre-requisites	PR1, PR2, PR3, PR4, PR5	Setup	SETUP-1

³ The current tests have been performed via VPN, across Wifi. They might lead to KPI values representing delays that have not been taking into account when describing the reference target values that are based on 5G networks.

Test tools		
Components Involvement	All AFFORDABLE 5G Components ⁴ + Mission Critical Services	
Test sequence	Step 1	Select from Client-01 the common MCPTT group with Client-02
	Step 2	Set up a MCPTT pre-arrange group call
	Step 3	Accept the call at the Client-02 (this step can be automatic)
	Step 4	Request the token to talk from Client-01
	Step 5	Release the token
	Step 6	Repeat the process, step 5 and 6, several times
	Step 7	Hang up the call and obtain the measurements from device logs or database.
	** This process might be automated by a script launching several calls.	
Validation Criteria	This measure will characterize the time between a request is sent to the server and this request is processed and answered. According to 3GPP TS 22.179, MCPTT Access Time shall be less than 300 ms for 99% of all MCPTT requests. Also, the Access Time cannot be less than the network RTT, for that, we consider the same optimal value as network RTT.	
Target Values	Upgradable ≥ 100 ms > Acceptable ≥ 40 ms > Optimal	
Experiment Variations	A: Service instantiated in the CORE B: Service instantiated in the EDGE	

Technical test #3: MCPTT E2E Access Time

Test case name	MCPTT E2E Access Time	Test Case id	Test-04-03
Test purpose	MCPTT access time is defined as the time between when an MCPTT User requests to speak and when this user gets a signal to start speaking and it does not include confirmations from receiving users, as defined by the 3GPP Technical Specification (TS 122 179. Section 6.14).		
Pre-requisites	PR1, PR2, PR3, PR4, PR5	Setup	SETUP-1
Test tools			
Components Involvement	All 5G Components ⁵ + Mission Critical Services		
Test sequence	Step 1	Select from Client-01 the common MCPTT group with Client-02	
	Step 2	Set up a MCPTT pre-arrange group call, automatic answer mode must be configured at the receiver (Client-02)	

⁴ The current tests have been performed via VPN, across Wifi. They might lead to KPI values representing delays that have not been taking into account when describing the reference target values that are based on 5G networks.

⁵ The current tests have been performed via VPN, across Wifi. They might lead to KPI values representing delays that have not been taking into account when describing the reference target values that are based on 5G networks.

	Step 3	Hang up the call
	Step 4	Repeat this process, from Step 1 to 3, several times.
	Step 5	Obtain the measurements from device logs or database.
	** This process might be automated by a script launching several calls.	
Validation Criteria	This measure will characterize the time between a request is sent to the server and it is processed and answered. According to TS 122.179, MCPTT Access Time shall be less than 1000 ms, when both users are under the coverage of the same network. Since the procedure requires at least two RTT and allocate resources at the server, we consider as an optimal value an MCPTT E2E less than 250 ms.	
Target Values	Upgradable ≥ 1000 ms > Acceptable ≥ 250 ms > Optimal	
Experiment Variations	A: Service instantiated in the CORE B: Service instantiated in the EDGE	

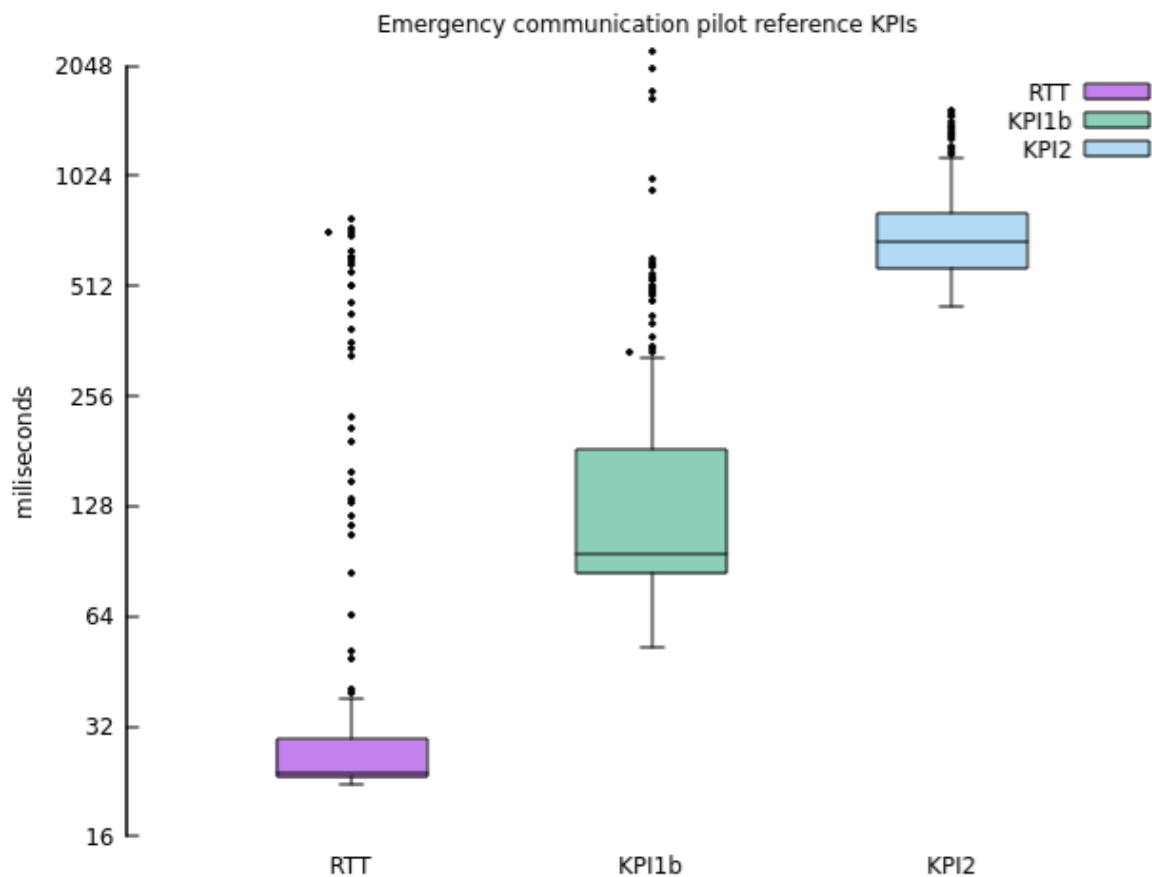


Figure 97: Boxplot representing the KPI obtained

Table 6: Detailed KPI values for the iterations carried out

KPI	RTT (ms)	KPI1b (ms)	KPI2 (ms)
Samples	250	250	250
Avg	71.69	168.25	734.16

Min	22.3	53	451
Max	780	940	1542
Median	23.8	95	678.5
StDev	147.89	162.43	234.49
Mode	23.4	86	639
P25	23.3	83.75	572.25
P75	29.65	164.5	808.25

As mentioned in the notes, the test conditions differ from the ones considered in the first approach, tests have been carried out on top of a VPN through Wifi instead of 5G network due to the unavailability of the Castellolí 5G platform. Nevertheless, RTT and MCPTT E2E Access Time (KPI2) remain in the acceptable defined range for 5G network, whereas MCPTT Access Time is beyond the acceptable established limit but differs not much from the 100ms established range.

KPI	RTT (ms)	KPI1b (ms)	KPI2 (ms)
Avg	71.69	168,25	734.16

6.4.5 Innovations and conclusions

Intelligence loop between MCX app, NWDAF and Orchestrator

The continuous communication loop and coordination between the three components that is demonstrated in scenario 1 outlines the concept of zero-touch automation networks and self-configuration services. Specifically, the metrics provided by the MCX application to the NWDAF are used not only to monitor the requirements of the service and, ultimately the QoS of the PPDR solution, but also to predict future needs and alert the Orchestrator to perform pro-active actions regarding the service scalability. Towards this direction, MCS scenario 1 illustrates a closed-loop solution for the optimization of 5G network resources and services in an automated and self-configured manner. Furthermore, the aforementioned scenario describes the practical implementation of an intelligence loop that can be used in disaggregated O-RAN and details how the training/testing of ML models can be realized using realistic data within a PPDR service architecture. In this context, the NWDAF (which is also a 3GPP-compliant component) consumes the time-series data originating from the MCX application and provides prediction alarms to the Orchestrator concerning service scalability. The Orchestrator, incorporating the functionalities of a near-real time intelligent controller can then perform the required actions of scaling up the network resources, targeting to the reliability of the MCX service under enhanced traffic load.

Metrics

An important innovation that has taken place in the context of the mentioned use cases is the gathering and exposure of MCX service-related metrics, such as the number of registered users, number of active private calls and number of active group calls. As proven in scenario 1 this is a key enabler for service behaviour prediction, which in turn allows for pre-emptive actions against future issues like system overload. This adds robustness and reliability to the service, important assets for PPDR scenarios. At the same time, the gathering of service-related metrics also allows for a more classical approach of service and performance monitoring which also adds value and enhance the usability of the MCX service.

Service scalability (load balancing)

In the case of the pre-emptive actions performed after the corresponding predictions in scenario 1, one of its key enablers is the service scalability. The possibility of enhancing the

deployment in case of need with new pods and also the correct balancing of the load among them in a correct and seamless way for the users constitutes an important innovation with a great value for PPDR scenarios, improving again the reliability of the service securing its endurance against unexpected issues or critical situations.

Multi-PoP

The multi-PoP feature, developed in the framework of Affordable 5G project to give response to one of the use cases described initially in the proposal, is a very interesting capability for PPDR (Public Protection and Disaster Relief) scenarios, since it allows a full deployment to be re-deployed in a new PoP on demand, also featuring a seamless reconnection of the impacted users, avoiding an outage of the service in the process. This capability can be used in situations where the service suffers a degradation in its performance or a shortage of resources, moving and re-deploying the full service in a PoP located closer to the user or with more available resources, hence solving the service issues that may have appeared in the initial PoP.

7 CONCLUSIONS

In this journey to build an affordable 5G SA network, many technical difficulties have been encountered that have been addressed throughout the development and integration of the different components. Some of these challenges have been successfully overcome and the integrations were achieved, but others had to be managed in a different way to surpass the difficulties. For this reason, this deliverable has not only focused on pilot's testing and validation, but also in other partial integrations, either inside Malaga and Castellolí platforms or outside at partners lab facilities.

In any case, the focus remains on pilots validation and an updated vision of all 3 deployed pilots is presented together with work carried out to roll-out the final version in each of the 5G solutions in both testbeds.

One of the main issues that have conditioned the deployment of the pilots is the lack of a full O-RAN solution (RU-DU-CU) fully operationally integrated and provided by the consortium. This does not mean that a lot of work have not been done to this end, but operational constraints led the consortium to consider other options to build the solution. Thus, alternative O-RAN solutions have been proposed to be utilized in both platforms. These alternatives also came with new integrations issues that were solved in different ways in each of the platforms, as it has been explained. A completed explanation addressing objectives achieved, faced issues and partial integrations such as CU element and 5GCore, O-RAN fronthaul and S-Plane implementation, introduction of AI/ML framework for smart control loops, slicing and orchestration capabilities as well as usage of edge computing and others has been well reported in the document.

All setbacks have conditioned the smooth execution of test cases and its KPIs validation. Even so, necessary efforts were done to provide a complete end to end operational 5G SA solution on testbeds for pilots. This allows us to present a wide set of test cases including real results, over project platforms or other lab facilities.

As final deliverable concerning WP4, the general conclusion is that integration and validation tasks are critical processes generally underestimated in which is not easy to put together all developments to converge in a full operational system level solution. Even so, Affordable5G has allowed partners to improve their technologies, to enhanced products and to evolve and test innovative features within project's framework as shown incrementally across technical deliverables, starting from simple and individual building blocks and ending as multi-vendor collaborative 5G network integrations.

REFERENCES

- [1] Affordable5G, D4.2. 5G roll-out and system testing report [Online], <https://www.affordable5g.eu/deliverables/>, Accessed: December 2022.
- [2] Affordable5G, D4.1. Affordable5G, D4.2. 5G roll-out and system testing report [Online], <https://www.affordable5g.eu/deliverables/>, Accessed: December 2022.
- [3] Netropy, «Netropy Network Emulator, v4.0,» 10 2019. [Online]. Available: <https://www.apposite-tech.com/wp-content/uploads/2019/10/Netropy-4.0.-usersguide-2019.pdf>
- [4] Intel, «Intel FlexRAN Github 'Real-Time Host Installation',» [Online]. Available: <https://github.com/intel/FlexRAN>
- [5] P4 Open Source Programming Language [Online] - <https://p4.org/>, Accessed: June 2022.
- [6] Affordable5G, D3.1. Open platform usage developments [Online], <https://www.affordable5g.eu/deliverables/>, Accessed: December 2022.
- [7] Affordable5G, D3.2. Open platform usage developments [Online], <https://www.affordable5g.eu/deliverables/>, Accessed: December 2022.
- [8] 5G-CLARITY D4.1. Initial Design of the SDN/NFV Platform and Identification of Target 5G-CLARITY ML Algorithms [Online], <https://www.5gclarity.com/index.php/deliverables/>, Accessed: December 2022
- [9] 3GPP, TS 22.179 22. Mission Critical Push to Talk (MCPTT) [Online], https://www.3gpp.org/ftp/Specs/archive/22_series/22.179/, Accessed: December 2022.
- [10] TheAIGuysCode, Object tracking implemented with YOLOv4, DeepSort, and TensorFlow [Online]. Available: <https://github.com/theAIGuysCode/yolov4-deepsort>, Accessed: December 2022.