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

High-tech and affordable 5G network roll-out to every corner

D3.1: Open platform usage developments

Revision: v1.0

Work package	WP3
Task	Task 3.1, 3.2, 3.3
Due date	31/07/2021
Submission date	29/07/2021
Deliverable lead	I2CAT
Version	1.0

Abstract

This first deliverable of WP3 reports on the initial status of the software open platforms and focus on analysing the specific components selected to form the Affordable5G architecture. The document details for each layer of the Affordable5G architecture (as defined in D1.2), the software elements considered, and evaluates the description and motivation for their selection, the main interfaces defined, and the most relevant technical challenges envisioned. Finally, each component dissects the requirements identified in D1.2 used to steer the project developments and the successful completion of the pilots, analysing their level of fulfilment and roadmap expected for the second half of the project.

Keywords:

List of Contributors

Partner	Short name	Contributor(s)
ATOS SPAIN SA	ATOS	Sergio González Díaz Borja Otura García Josep Martrat
ADVA Optical Networking Israel Ltd	ADVA	Andrew Sergeev
ACCELLERAN	ACC	Simon Pryor
ATHONET SRL	ATH	Marco Centenaro, Nicola di Pietro, Arif Ishaq, Daniele Munaretto
RUNEL NGMT LTD	REL	Israel Koffman
MARTEL GMBH	MAR	Gabriele Cerfoglio
EIGHT BELLS LTD	8BELLS	George Kontopoulos
NEARBY COMPUTING SL	NBC	Angelos Antonopoulos, Oscar Trullols
UNIVERSIDAD DE MALAGA	UMA	F. Luque-Schempp, F. J. Rivas, P. Merino
ETHNIKO KAI KAPODISTRIAKO PANEPISTIMIO ATHINON	NKUA	Panagiotis Trakadas, Anastasios Giannopoulos, Sotirios Spantideas, Lambros Sarakis, Panagiotis Gkonis
FUNDACIO PRIVADA I2CAT, INTERNET I INNOVACIO DIGITAL A CATALUNYA	I2CAT	Giovanni Rigazzi, Estefania Coronado
EURECOM	EUR	Navid Nikaein Sofia Pison

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	04/05/2021	ToC	I2CAT
V0.2	18/05/2021	ToC revised	I2CAT, ACC, NBC, NKUA
V0.3	10/06/2021	Integrated first input	I2CAT, NBC, NKUA, ATOS, MAR
V0.4	17/06/2021	Integrated new input	I2CAT, UMA, ATH
V0.5	24/06/2021	Integrated input from ACC	ACC
V0.6	26/06/2021	Integrated input from REL	REL
V0.7	01/07/2021	First full version of the document	ALL
V0.8	12/07/2021	Integrated first round of reviews	ALL
V0.9	27/07/2021	Integrated second round of reviews	NKUA, ATOS, I2CAT
V1.0	29/07/2021	Final check	ATOS, I2CAT

Disclaimer

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	O	
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 deliverable "D3.1. Open platform usage developments" reports on the progress and achievements of the tasks T3.1, T3.2 and T3.3, versing on split of target functions and open platforms, RAN, core and edge open solutions, and network management solutions, respectively.

The main objective of this document is to identify and assess the software components that will take part of the Affordable5G system architecture. These software components must be aligned with the principles of the project and prove to fulfil the system requirements identified in D1.2 to optimize open software platforms for 5G network elements.

The Affordable5G architecture, divided into i) network function layer, ii) management orchestration and automation layer, and iii) infrastructure layer, as defined in deliverable D1.2, is taken together with the requirements identified in the same document for the Affordable5G pilots in private and enterprise networks, in order to serve as a reference for the selection of the specific software tools.

In this context, this deliverable contributes to the Affordable5G project by:

- Motivating the selection and technical challenges faced by the different software components at each architectural layer.
- Identifying the interfaces related the different components and the fulfilment of the required functionalities.
- Analysing the requirements related to each of the software artefacts and their level of fulfilment at this stage of the project.

The analysis carried out in this document will be used as input for the integration plan performed in WP4 as well as for the roadmap and evolution of the software components envisioned in the second half of the project lifetime.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

3GPP	3rd Generation Partnership Project
5G	5th Generation of Mobile Networks
5GC	5G Core
AF	Application Function
AI	Artificial Intelligence
AKA	Authentication and Key Agreement
AMF	Access and Mobility Management Function
ARPF	Authentication credential Repository and Processing Function
AUSF	Authentication Server Function
CFRA	Contention Free Random Access
CM	Configuration Management
CN	Core Network
CP	Control Plane (also C-plane)
CP	Cyclic Prefix
CU	Central Unit
CU-CP	Centralized Unit Control Plane
CU-UP	Centralized Unit User Plane
DL	Downlink
DN	Data Network
DNS	Domain Name System
DS-TT	Device-side TSN translator
DU	Distributed Unit
E2AP	Application Protocol
E2E	End-to-end
E2SM	Service Model
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
FHI	Fronthaul Interface
GBR	Guaranteed Bit Rate
gNB	gNodeB, 5G NR, next generation NR eq. of base station
GNSS	Global Navigation Satellite System
GTP-U	GPRS Tunnelling Protocol for User data
GUTI	Global Unique Temporary Identifier
HW	Hardware
iFFT	Inverse Fast Fourier Transform

IoT	Internet of Things
IP	Internet Protocol
ISG	Industry Specification Group
KNF	Kubernetes Network Function
LLS	Low Layer Split
MANO	Management and Orchestration
MC-PTT	Mission-Critical Push-To-Talk
MDAF	Management Data Analytics Function
MEC	Multi-access Edge Computing
MEO	MEC Orchestrator
MEP	MEC Platform
MEPM	MEC Platform Manager
ML	Machine Learning
MNO	Mobile Network Operator
MnS	Management Service
MOCN	Multi-Operator Core Networks
MORAN	Multi Operator Radio Access Network
MPSOC	Microprocessor System-On-Chip
NAS	Non-Access Stratum
Near-RT RIC	Near Real Time RAN Intelligent Controller
NETCONF	Network Configuration Protocol
Non-RT RIC	Non-Real Time RAN Intelligent Controller
NF	Network Function
NFVI	Network Function Virtualization Infrastructure
NG	Next-Generation
NG AP	NG Application Protocol
NPN	Non-Private Network
NR	New Radio
NRF	Network Repository Function
NSA	Non-Standalone
NSSAI	Network Slice Selection Assistance Information
NSSF	Network Slice Selection Function
NWDAF	Network Data Analytics Function
NW-TT	Network-side TSN translator
OAM	Orchestration And Management
OFDMA	Orthogonal Frequency-Division Multiple Access

OSM	Open Source MANO
O-CU	Control Unit: a logical node hosting PDCP, RRC, SDAP and other control functions
O-DU	Distributed Unit: a logical node hosting RLC/MAC/High-PHY layers based on a lower layer functional 7 split.
OSS	Operation Support Systems
PCC	Policy and Charging Control
PCF	Policy Control Function
P-CSCF	Proxy - Call Session Control Function
PDCP	Packet Data Convergence Protocol
PDU	Protocol Data Unit
PFCP	Packet Forwarding Control Protocol
PLMN	Public Land Mobile Network
PM	Performance Measurement
QoS	Quality of Service
RAN	Radio Access Network
RIC	RAN Intelligent Controller
RLC	Radio Link Control
RRC	Radio Resource Control
RU	Radio Unit
SCTP	Stream Control Transmission Protocol
SIDF	Subscriber Identity De-concealing Function
SLO	Service Level Objective
SMO	Service Management and Orchestration
S-NSSAI	Single Network Slice Selection Assistance Information
SoC	System on Chip
SUCI	Subscription Concealed Identifier
SUPI	Subscription Permanent Identifier
UDM	Unified Data Management
UDR	Unified Data Repository
UE	User Equipment
UL	Uplink
UP	User Plane (also U-plane)
SA	Standalone
SMF	Session Management Function
TFS	TensorFlow Serving
TFX	TensorFlow Extended

TSN	Time Sensitive Network
VES	Virtual Event Streaming
VIM	Virtual Infrastructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
YANG	Yet Another Next Generation

1 INTRODUCTION

1.1 Objectives of the deliverable

The main objective of this deliverable, *D3.1: Open platform usage developments*, is to provide a holistic view on the software components that form the Affordable5G system architecture as well as to report on the output of the activities carried out in T3.1, T3.2 and T3.3.

In this manner, this document aims to provide a full analysis of the software components selected, the motivation behind such a selection and the software design choices that led to it. Such software components regard the RAN, 5Gcore, edge and management systems, per each of the architectural layers defined in D1.2 [1] as reported in the next subsection.

The deliverable covers for each software component their context and motivation, the interfaces defined and their integration, and the technical challenges. In addition to it, this document aims to provide a review of the requirements defined in D1.2 and analyze their status for each related software component to assess the technical roadmap pending in terms of software design and development for the second half of the project.

1.2 Overview of the Affordable5G system architecture

The system architecture of Affordable5G was introduced in the deliverable D1.2 [1], differentiating essentially three main layers, namely Management, Orchestration and Automation layer, Network Function layer and Infrastructure layer, as depicted in Figure 1.

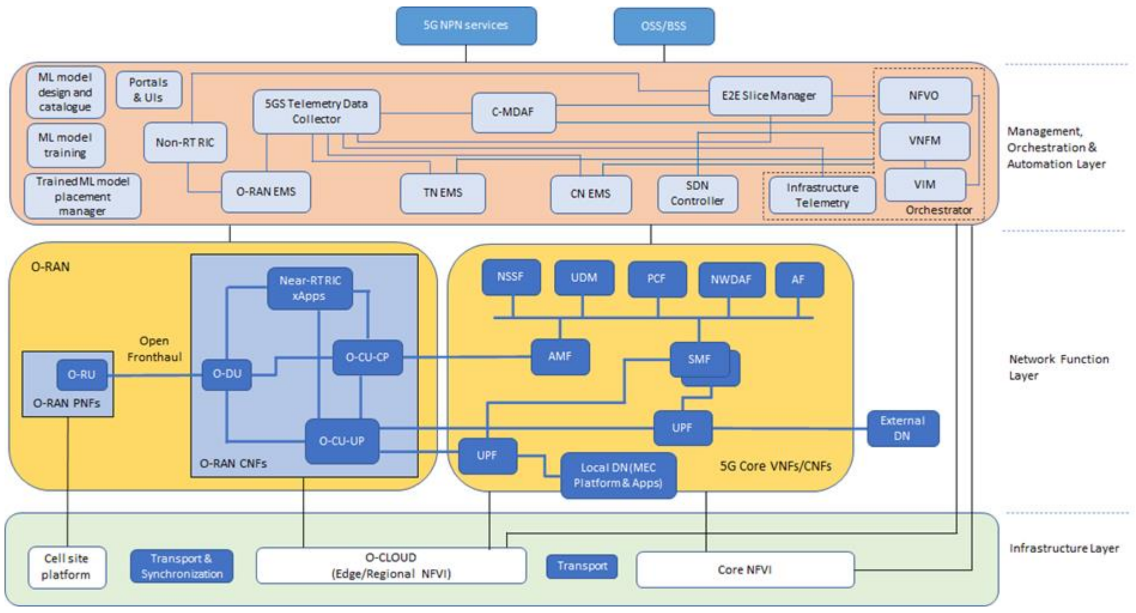


Figure 1: Affordable5G system architecture

The **Infrastructure Layer**, consisting of the Cell Site platform ensuring the OpenRAN approach to the overall solution, the Edge/Regional Network Functions Virtualisation Infrastructure (NFVI) and the Transport and synchronization network, are described in detail in *D2.1: Hardware elements and usage in Affordable5G solutions*. Transport and synchronization components will be provided by ADVA. Global Navigation Satellite System (GNSS) will be used as the synchronization source used by the Radio Unit (RU) and the Distributed Unit (DU). Time Sensitive Network (TSN) components and experimentation setup will be handled by UMA and ADVA.

At the **Network Function Layer**, the Affordable5G RAN adopts a double split option (RU-DU-Centralized Unit (CU)), making use of the 3GPP option 2 high-layer split and the O-RAN Alliance option 7.2 low-layer split.

An RU, provided by RunEL, is interfacing to a DU, provided by Eurecom, implementing the option 7.2 physical layer split. The DU is connected to a CU, provided by ACC, through the 3GPP F1 mid-haul standardized interface. The CU is split in the Centralized Unit User Plane (CU-UP) and Centralized Unit Control Plane (CU-CP). New CU-UP instances are created per Public Land Mobile Network (PLMN) / Simple Network Management Protocol (SNMP) ID (for neutral hosting) and per network slice.

An O-RAN Near-Real Time (RT) Radio Intelligent Controller (RIC), provided by ACC, is connected to the underlying nodes through the E2 O-RAN alliance interface. The CU connects to mobile core (5GC) via the N2 and N3 backhaul interfaces (NG-C/NG-U). Athonet is providing Affordable5G with a SA and fully virtualized 5GC, compliant to 3GPP technical specifications. Core network functions, particularly the User Plane Function (UPF), can be distributed to the edge of the network, bringing them closer to the end users. This way, edge computing can be enabled, steering traffic directly towards the servers where edge applications run. Both the RAN and Core Network support slicing, allowing deployment of independent services with performance guarantees.

At the **Management, Orchestration & Automation Layer**, the architecture differentiates various functionalities, including orchestration, management at the O-RAN non-RT RIC level, network slicing, telemetry tasks and Artificial Intelligence (AI)-based functions.

The Affordable5G architecture aims to leverage to the possible extend the advantage offered when standardized interfaces are used in the system. For that reason, the Management and Orchestration (MANO) stack will include and test two different solutions, namely NearbyOne Orchestrator and Open Source MANO (OSM). The NearbyOne Orchestrator, provided by Nearby Computing, will be used as an end-to-end orchestration engine, including the service placement function, the multi-access edge orchestrator, the Multi-access Edge Computing (MEC) platform manager and the MEC platform. OSM will be enhanced in the area of Kubernetes Network Functions (KNFs) placement.

The i2CAT solution will be used as a slice manager engine, interconnected to the orchestrator and the non-RT RIC. The non-RT RIC will be provided by i2CAT, as an orchestration and automation component as described by the O-RAN Alliance for non-real-time intelligent management of RAN functions. The non-RT RIC will communicate with near-RT RIC elements in the RAN via the A1 interface. An O-RAN compliant EMS will be provided by i2CAT, performing Configuration Management (CM) operations through the Network Configuration Protocol (NETCONF)/Yet Another Next Generation (YANG) protocol to the RAN.

A Telemetry data collector framework will be employed, developed by NKUA, supporting collection of network data analytics to be used by the Machine Learning (ML) analytics module. An AI/ML framework will be developed by ATOS, employing the TensorFlow ecosystem. AI/ML models will be executed in the O-RAN near-Real-Time RIC, employing the xAPP framework supported by dRAX. The Affordable5G AI/ML framework aims to making the orchestration and management processes intelligent and adaptative, incorporating the monitoring and telemetry information coming from the network analytics functions.

Two different implementations will be employed at the two Affordable5G sites (Malaga and Castelloli). Both setups will employ an end-to-end 5G network. The Malaga implementation will mostly focus on the industrial use cases, leveraging the TSN experimentation setup. The Castelloli implementation will focus on MANO, RAN optimization through AI/ML automation and Smart City use cases. More details on the differences between the two implementations can be found on the WP4 deliverables.

2 NETWORK FUNCTION LAYER

2.1 O-RAN

The Affordable5G RAN is conformant to a 3GPP NG-RAN as described by [2]. Additionally, the Affordable5G RAN is also aligned to the O-RAN Alliance architecture [3], adding the Near-RT RIC (RAN Intelligent Controller) and associated interfaces (like O1/O2/A1/E2) as highlighted below in Figure 2:

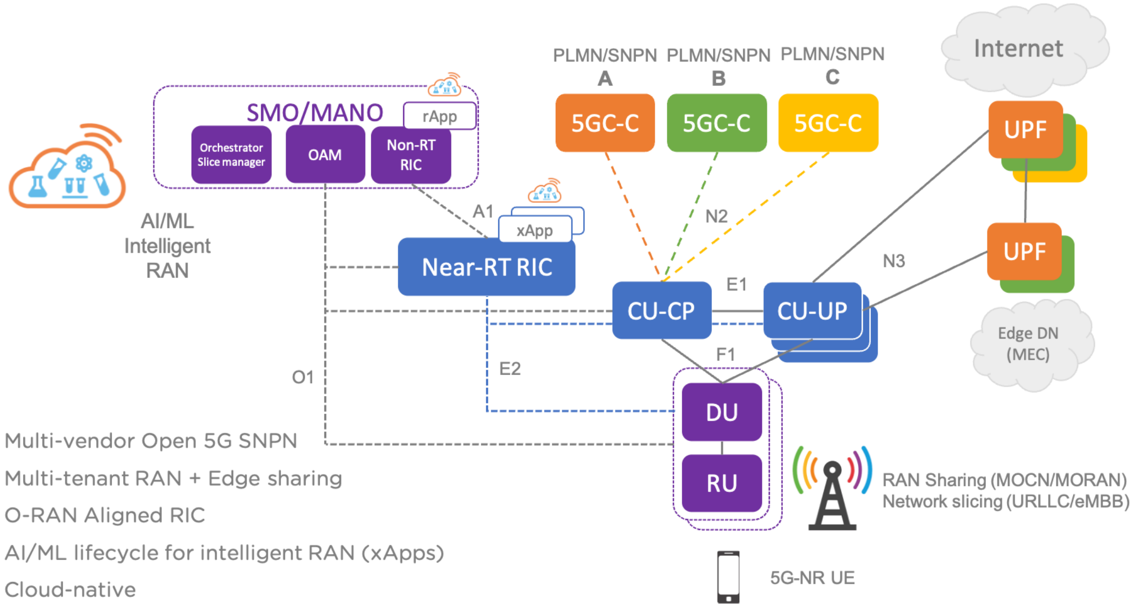


Figure 2: Affordable5G 3GPP NG-RAN and O-RAN Alliance aligned RAN

2.1.1 Radio Unit

2.1.1.1 Description and Motivation

The RU or O-RAN Radio Unit O-RU is the unit at the edge of the access network. It interfaces with the O-DU northbound via standardized O-RAN Interfaces and with the 5G User Terminals and Internet of Things (IoT) sensors via air Interface as depicted in Figure 3 below.

RunEL selected the Sparq-2020-ORU [4] as the candidate to perform the RU functionalities needed in the Affordable5G due to the following main reasons:

- Includes O-RAN Interface between the O-RU and the O-DU, therefore the integration process is expected to be simple and fast.
- Has an embedded 3.3 to 3.8 GHz beam forming antenna that has been auctioned as 5G spectrum in Europe.
- It is programmable, based on FPGA, therefore it will be relatively simple to make changes if necessary.
- It can be installed in indoor (UMA) or outdoor (Castelloli) environments, as required by the project.

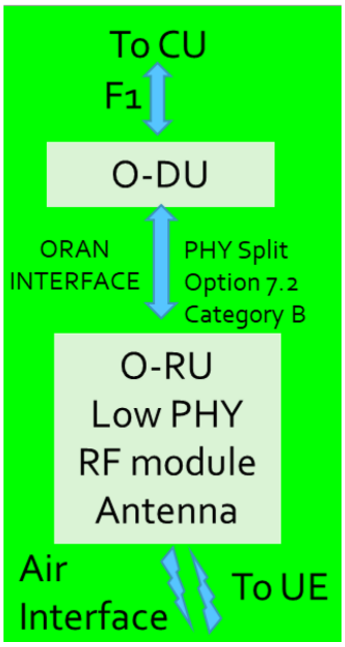


Figure 3: RU Interfaces

The O-RU implements the Physical Layer (layer-1) split defined by the O-RAN association as option 7.2 [3], therefore the O-RU device implements *Low-PHY* function that is connected to the *Hi-PHY* implemented in O-DU.

As mentioned earlier, the O-RU was designed for both indoor and outdoor operation and will be installed in the Affordable5G project in the UMA Testbed (Indoor) for Time Sensitive Network test and measurements and in the Outdoor Castelloli deployment in order to test the end-to-end performance (latency, throughput, coverage, etc.) of the 5G Network in a real case scenario.

2.1.1.2 Interfaces

All the O-RU interfaces targeted in Affordable5G are following standards based on:

- O-RAN Interface between the O-RU and O-DU (Option 7.2) over a 10Gbps Ethernet link.
- Orthogonal Frequency-Division Multiple Access (OFDMA) Air Interface between the O-RU and the User Terminals according to the 3GPP standard (Sub 6GHz band).

The FPGA software/firmware is written using RTL language and is all RunEL own IP and not based on any open-source software. A specialized HW must be used for O-RU. The RunEL Sparq-2020-ORU is shown in Figure 4.

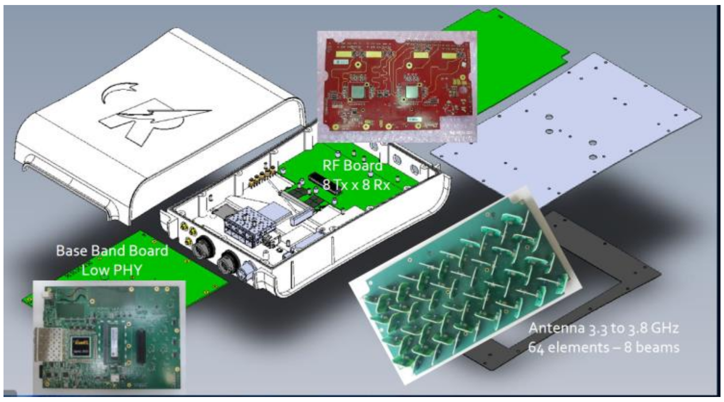


Figure 4: RunEL- RRH (O-RU)

The O-RU includes the following main components:

- Baseband Board.
- RF Board.
- Beam Forming Antenna.

The Baseband Board is based on the Sparq-2020 System on Chip (SoC) that executes the Low Physical Layer Modules and the Beamforming Precoding modules (according to O-RAN Specifications Category B) and O-RAN Interface functionalities. The SoC is based on a powerful FPGA accelerator device. The Baseband Board also includes GPS, 1PPM and 10 MHz interfaces for time Synchronization and 8 x 10Gbps Ethernet links for backhaul and control purposes.

The RF Board covers the Sub 6GHz bands and is based on an 8 channels Analogue to Digital and Digital to Analogue RFIC that drives the beam forming antenna.

The Beam Forming Antenna is a 64 elements antenna that covers the 3.3-3.8 GHz band with four dual polarized beams.

2.1.1.3 Requirements' Mapping

The O-RU satisfies the following Affordable 5G requirements (as per D1.1 [5], Section 3.6):

No.	ID	Requirement	Note
1	REQ-NET-01	The network shall conform to 3GPP 5G network requirements, architectures and interfaces in both the 5GC and NG-RAN	Ongoing
2	REQ-NET-02	The system shall support operation as a 5G SNPN	Ongoing
3	REQ-NET-04	The system shall support 3GPP release Rel. 15 SA specifications	Done
4	REQ-NET-05	The system may support 3GPP release Rel. 16 SA specifications	Ongoing
5	REQ-NET-07	The system shall support control and user plane separation in the NG-RAN	Done

No.	ID	Requirement	Note
6	REQ-NET-10	The system shall support several synchronization means: GNSS, PTP and SyncE	Done
7	REQ-NET-11	The system shall support MIMO functionality	Done
8	REQ-NET-14	The system shall support Multi Band and Multi Operator RRU (MORAN & MOCN)	Done
9	REQ-NET-17	The system shall support inter 5G cells mobility	Done
10	REQ-NET-19	The system shall support Ethernet based fronthaul with TSN functionality	Ongoing
11	REQ-NET-20	The system shall support operation in the TS 38.101-1 Frequency Range 1 FR1 (410-7125 MHz) bands including support for bands n77 and n78.	Done
12	REQ-NET-22	In the NG-RAN, the RU/DU/CU network functions shall support HLS/LLS split architecture to support multi-vendor RAN using open interfaces.	Done
13	REQ-NET-24	The system shall support authentication of the UEs and authentication of the network towards the user	Done
14	REQ-NET-39	The system shall support small outdoor cells capabilities	Done
15	REQ-NET-40	The system shall support remote management and separate management domain per operator.	Done
16	REQ-NET-44	The system should support data collection from NFs residing on the core and access part of the 5GS	Done
17	REQ-NET-58	The 5G system shall support high-bandwidth streams from a massive set of devices with a user experienced data rate of up to 100 Mbit/s	Done
18	REQ-NET-59	RUs and DUs should be deployed according to the scheduled task that may require communication either with distant AMRs or link establishment in harsh conditions	Ongoing
19	REQ-NET-60	RU and DUs should be deployable in GNSS-less environment	Ongoing
20	REQ-MAN-12	The system shall be able to configure the HW for specific workloads	Done

Table 1: Requirements' mapping of the O-RU

2.1.1.4 Technical Challenges

The main O-RU enhancements that are achieved in the Affordable5G project are:

- Completion of the O-RAN Interface implementation.
- Integration with Commercial 5G User Terminals.
- Outdoor Installation at Cellnex cell site.

2.1.1.5 Plan for the Integration

The RU will be integrated to the DU northbound and the UE commercial terminal southbound.

The Integration with the DU from Eurecom will be performed over the ORAN 7.2 Interface Category B in two steps:

- Remote Integration by exchanging of Vectors and signal recordings between RunEL and Eurecom.
- On site Integration at Eurecom laboratories under a controller environment, comprising a full 5G SA deployment with all RAN and CN components.

Following the successful integration between the DU and the RU, a commercial UE will be connected to the 5G link at 3.5GHz frequency band.

2.1.2 Distributed Unit

2.1.2.1 Description and Motivation

The 5G NR architecture adopts a split into CU and DU by means of the F1 functional split, as defined in TS 38.401 [6]. The DU is usually located in a range of several kilometres from the RU and runs parts of the 3GPP PHY layer, MAC and Radio Link Control (RLC) layers (see Figure 5). This logical node includes a subset of the gNB functions, depending on the functional split option, and its operation is controlled by the CU, and possibly by the RIC via the O-RAN E2 interface. A single DU controls one or more RUs via a Fronthaul Interface (FHI) such as the O-RAN 7.2x functional split.

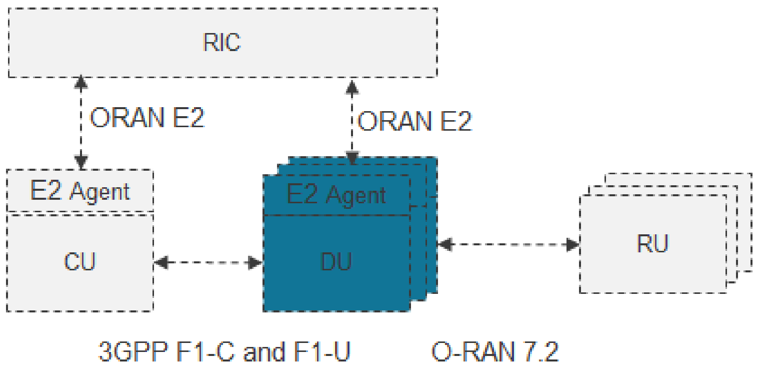


Figure 5: 5G NR functional split CU, DU, RU

The software tool adopted for this component is OpenAirInterface (OAI) due to its competitive advantages over other alternatives. In the first place, OAI's open-source software/hardware development model enables the use of the software and the implementation of the proposed DU in a testbed using commodity hardware. However, this may affect the outcome, as it does not economically restrain the reproducibility of the obtained results for future users.

Additionally, and since OAI's software is open source, it is expected that some of the results from this project will be integrated into the main baseline project. This will help the research community to better understand the potentials and limits of future 5G RANs and enhance an already vibrant open-source project. Finally, OAI is the most feature rich open-source project for 3GPP cellular networks, being able to implement both NSA and SA deployments.

OAI software includes the following features which are currently available and under continuous development:

- NSA.
- SA.
- FR1.
- SISO.
- Scheduling.
- F1AP (F1-C and F1-U) interfaces.
- E2 interface at CU and DU.

The supported NSA architecture is compliant to option 3a as per 3GPP Rel.15.

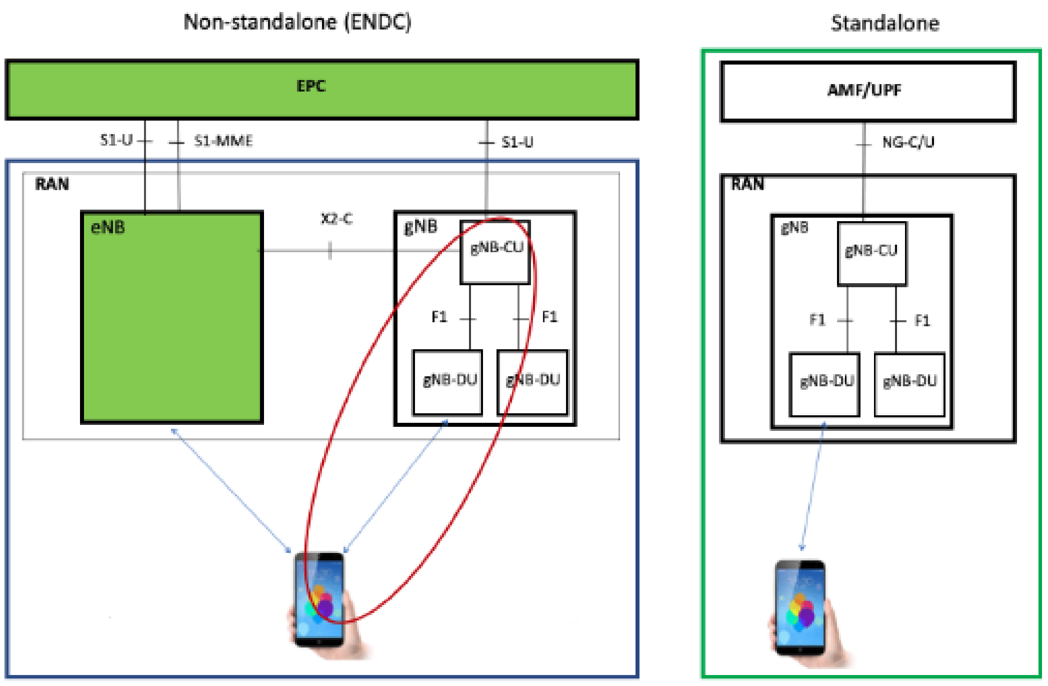


Figure 6: 5G NSA and SA. Source [7]

The developments in the OAI eNB and gNB can be grouped per layer as follows:

- MAC/PHY:
    - o Integration of 5G NR Contention Free Random Access (CFRA) procedures to enable successful 5G connection of the UE to the 5G cell according to specifications [8] and [9].

  - o Complete integration of SCF 5G FAPI interface between MAC and PHY layer according to [10].
  - o Integration of dynamic scheduling and capability to support multiple users.
  - o Integration of HARQ procedures on top of downlink and uplink physical channels to provide support for data acknowledging and retransmissions according to [11].

- PDCP/RLC: Complete implementation of NR PDCP and RLC AM (Acknowledged Mode) and UM (Unacknowledged Mode) according to [12] and [13] in order to support data plane traffic over 5G established DRBs.
- RRC extensions according to [14] and [9]:
  - o Integration of the LTE eNB procedures triggering the UE addition/release request to the 5G cell and the data path switch procedures towards the 5G cell.
  - o Extensions of LTE RRC messages with NR message containers originating from the gNB.
  - o Integration of all the gNB NSA configuration procedures for a UE added to the NR cell and construction of the corresponding NR message containers conveyed to the UE through the eNB.
- X2AP extensions according to [15]: Integration of the required X2AP messages and interfacing with RRC to support the establishment, maintenance, and release of an ENDC X2 connection between the eNB and gNB and the addition/release of a UE to the 5G cell.
- S1AP extensions according to [16]: Integration of the E-RAB Modification procedures initiated from the LTE cell (eNB) to trigger the data path switch towards the NR cell (gNB) at the core network.

2.1.2.2 Interfaces

3GPP F1

The CU-DU functional split, aggregate the PDCP layer and above in a CU, and the RLC layer and below in a DU. The standard interface between them is specified as F1. The F1 interface, connecting a gNB CU to a gNB DU, is a standardized interface that is specified in 3GPP TS 38.425 [17], TS 38.470 [18] and TS 38.473 [19].

The F1 interface is composed of two parts, the F1-C (control plane) that allows the signalling between CU-CP and DU and the F1-U (user plane) that allows the transfer of application data between CU-UP and DU. F1-C uses SCTP over IP while F1-U uses GTP over UDP over IP for the transport layer.

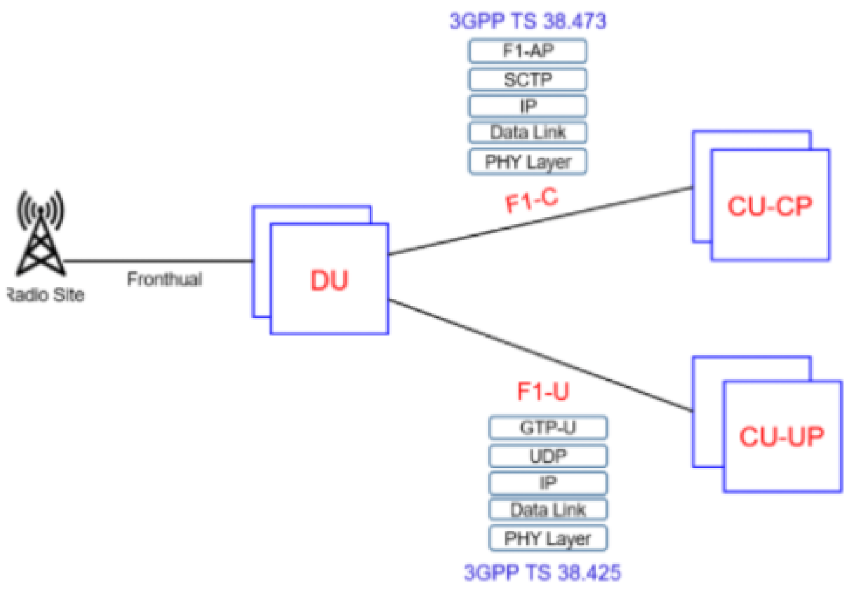


Figure 7: F1-U and F1-C in CU-DU split architecture. Source [20]

The functions related to F1 C-plane are mainly related to:

- Interface management (e.g., setup, reset, configuration).
- System information management.
- UE context management.
- Radio Resource Control (RRC) message transfer.
- UE Paging and warning message information transfer.
- On the other hand, F1 U-plane functions are related to UE data transfer and flow control.

CU/DU split mode integration in OAI (OpenAir Interface) gNB

The CU/DU split mode has recently been integrated in OAI gNB component according to 3GPP high level CU/DU functional split option 2. The OAI gNB portion is divided into two blocks: the CU containing the implementation of RRC and PDCP layers and the CU containing the implementation of RLC, MAC and PHY layers. The CU and DU components can thus run as separate programs in different hosts, offering flexibility for the deployment of the OAI 5G setup and the interoperability of OAI blocks with other commercial CUs or DUs.

Figure 8 depicts the 5G RAN protocol architecture of the OAI gNB according to the CU/DU functional split deployment. The extensions required for the control plane over F1-C (highlighted in green) have already been integrated and validated with the OAI Standalone (SA) end-to-end setup. The extensions for the integration of user-plane over F1-U are ongoing (highlighted in yellow).

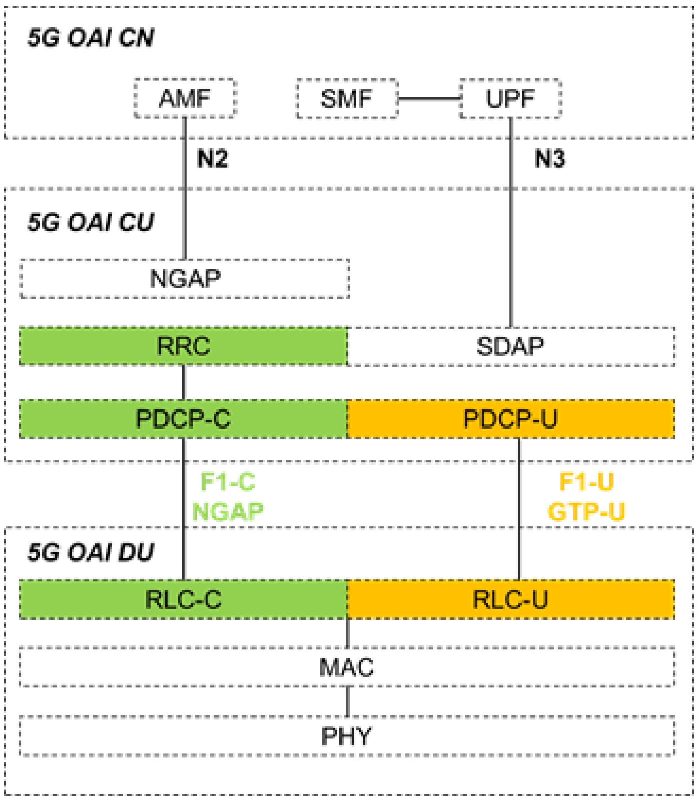
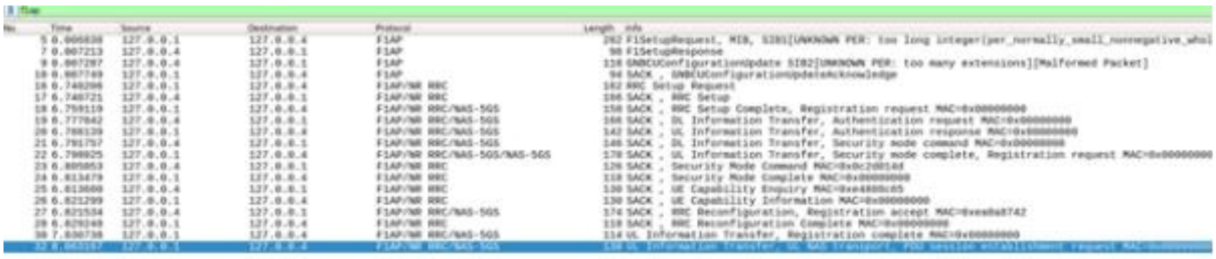


Figure 8: CU/DU split architecture option 2 and integration in OAI gNB

Figure 9 shows a traffic trace with all the F1AP exchanges taking place over F1-C interface at OAI gNB to support an RRC connection of a UE to the 5g cell, as well as its registration and Protocol Data Unit (PDU) session establishment with the Core Network.



```
f1ap
No.  Time       Source     Destination  Protocol                      Length  Info
  5 0.006838   127.0.0.1  127.0.0.4    F1AP                           262 F1SetupRequest, MIB, SIB1[UNKNOWN PER: too long integer(per_normally_small_nonnegative_whol
  7 0.007213   127.0.0.4  127.0.0.1    F1AP                            98 F1SetupResponse
  9 0.007287   127.0.0.4  127.0.0.1    F1AP                           118 GNBCUConfigurationUpdate SIB2[UNKNOWN PER: too many extensions][Malformed Packet]
 10 0.007749   127.0.0.1  127.0.0.4    F1AP                            94 SACK , GNBCUConfigurationUpdateAcknowledge
 16 6.740206   127.0.0.1  127.0.0.4    F1AP/NR RRC                    182 RRC Setup Request
 17 6.740721   127.0.0.4  127.0.0.1    F1AP/NR RRC                    186 SACK , RRC Setup
 18 6.759119   127.0.0.1  127.0.0.4    F1AP/NR RRC/NAS-5GS            158 SACK , RRC Setup Complete, Registration request MAC=0x00000000
 19 6.777042   127.0.0.4  127.0.0.1    F1AP/NR RRC/NAS-5GS            168 SACK , DL Information Transfer, Authentication request MAC=0x00000000
 20 6.788139   127.0.0.1  127.0.0.4    F1AP/NR RRC/NAS-5GS            142 SACK , UL Information Transfer, Authentication response MAC=0x00000000
 21 6.791757   127.0.0.4  127.0.0.1    F1AP/NR RRC/NAS-5GS            146 SACK , DL Information Transfer, Security mode command MAC=0x00000000
 22 6.798025   127.0.0.1  127.0.0.4    F1AP/NR RRC/NAS-5GS/NAS-5GS    178 SACK , UL Information Transfer, Security mode complete, Registration request MAC=0x00000000
 23 6.805053   127.0.0.4  127.0.0.1    F1AP/NR RRC                    126 SACK , Security Mode Command MAC=0x0c2d014d
 24 6.813479   127.0.0.1  127.0.0.4    F1AP/NR RRC                    118 SACK , Security Mode Complete MAC=0x00000000
 25 6.813608   127.0.0.4  127.0.0.1    F1AP/NR RRC                    130 SACK , UE Capability Enquiry MAC=0xe4860c65
 26 6.821299   127.0.0.1  127.0.0.4    F1AP/NR RRC                    130 SACK , UE Capability Information MAC=0x00000000
 27 6.821534   127.0.0.4  127.0.0.1    F1AP/NR RRC/NAS-5GS            174 SACK , RRC Reconfiguration, Registration accept MAC=0xea0a8742
 28 6.829240   127.0.0.1  127.0.0.4    F1AP/NR RRC                    118 SACK , RRC Reconfiguration Complete MAC=0x00000000
 30 7.830738   127.0.0.1  127.0.0.4    F1AP/NR RRC/NAS-5GS            114 UL Information Transfer, Registration complete MAC=0x00000000
 32 8.865187   127.0.0.1  127.0.0.4    F1AP/NR RRC/NAS-5GS            138 UL Information Transfer, UL NAS Transport, PDU session establishment request MAC=0x0000000
```

Figure 9: F1-C exchanges supporting configuration and control plane exchanges supporting UE 5G attachment

The implemented F1 container messages and the supported procedures can be grouped as follows:

- Setup (F1 Setup Request/Response): Covering the initial exchange of configuration data, to achieve interoperability between the CU and DU (also including System Information sent from the DU).
- Configuration updates (gNB CU configuration update/ACK): Covering updates of the configuration sent from the CU to the DU (which can include System Information).
- Initial UL/UL/DL RRC message transfers: Containers of UE-associated UL/DL RRC messages transferred transparently between the CU and DU.

ORAN E2

The E2 interface is proposed by the O-RAN alliance [3], which is composed by some major telecommunication industry companies. Its specifications can be downloaded from [21] after agreeing to the O-RAN adopter license.

As depicted in Figure 10, the E2 interface connects the O-RAN Near-RT RIC to the underlying E2 nodes, enabling the xApps loaded in the RIC to intelligently control the RAN Functions.

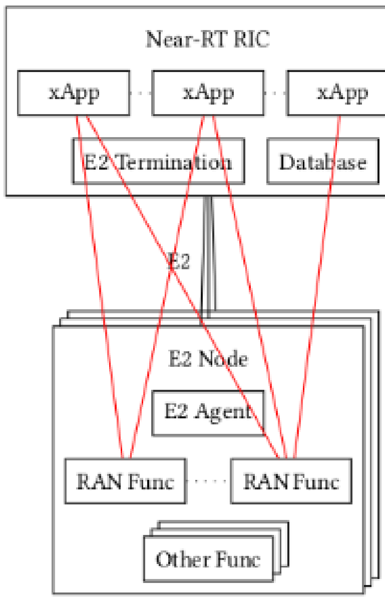


Figure 10: Near-RT RIC loaded with xApps connected through the E2 interface to the E2 Node

The E2 is divided into the Application Protocol (E2AP) and the Service Model (E2SM) [22]. The E2AP is composed of 26 messages that are encoded with PER ASN.1 and transported through SCTP. It defines the protocol and the message types that the E2 node and the RIC interchange. E2AP can be divided into the classes:

- E2 Global Procedures. These messages are used for connection management between a controller and the agent. Following a setup response, this includes reset messages, or connection update procedures.
- E2 Functional Procedures. These messages are specifically destined to RAN functions within the E2 agent for functional message exchange. Additionally, the functional procedures can be further divided into:
    - Subscription messages**:** That allow xApps to subscribe to event triggers in RAN functions to have them execute a specific action.
    - Indication messages**:** That allow RAN functions to send information to the RIC.
    - Control messages**:** That allow xApps to execute a procedure within a RAN function.

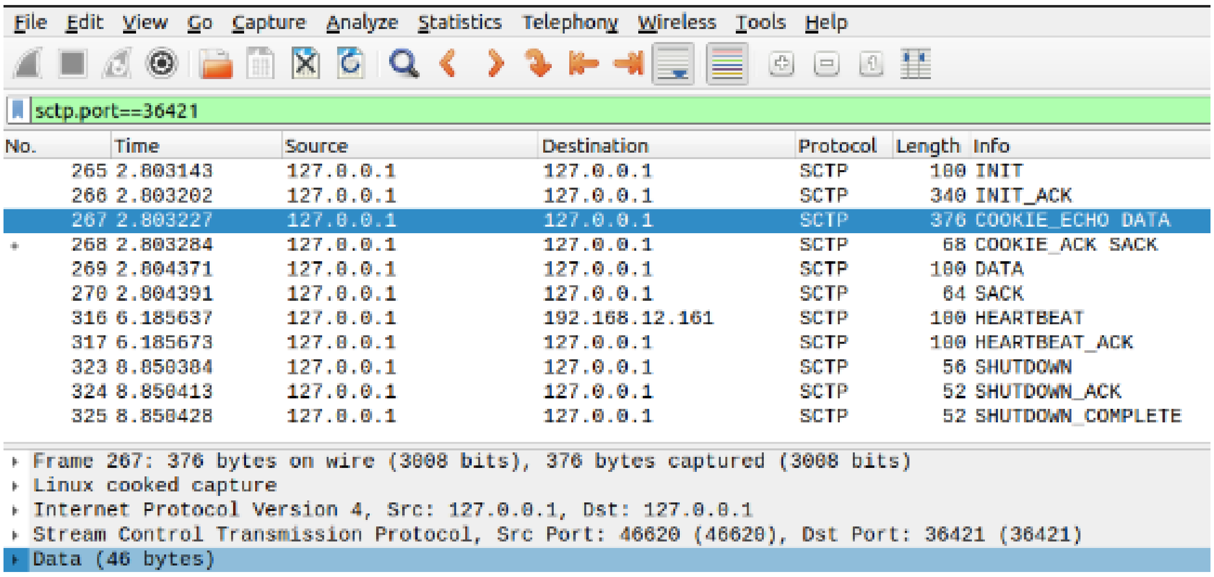
Every RAN function maps into an E2SM. In this manner, O-RAN decouples application interface (i.e., E2AP) from the service model (i.e., E2SM) that is contained and transported by E2AP. The E2SM is responsible for interpreting the bytes that arrive from 9 different

Information Elements, in 5 procedures that are specific to that SM. O-RAN also specifies the use of PER ASN.1 for transporting the E2SM.

E2AP and E2SM integration status

The E2AP and E2SM are developed from scratch as a C library in OAI with two encoding/decoding flavour, namely ASN.1 PER and Flatbuffers. The latter is specifically important for scenarios where the bandwidth is not the bottleneck, but processing shall be minimised. In these scenarios, Flatbuffers present better results as they avoid the encoding and minimise the copying of data. Similarly, the same technique has been followed when implementing the E2SMs.

In addition, a generic E2 agent and a RIC have been also developed. The E2 implementation is validated against the official O-RAN RIC with HelloWorld and KPM xApps to verify its correct behaviour. Besides, the integration of the E2 agent into OAI's RAN is also completed, so that it is possible to fetch data directly from a 4G/5G testbed through the E2 interface using the RIC. Figure 11 shows the E2 setup procedures in OAI.



```
sctp.port==36421

No.   Time      Source      Destination     Protocol Length Info
  265 2.803143  127.0.0.1   127.0.0.1       SCTP        100 INIT
  266 2.803202  127.0.0.1   127.0.0.1       SCTP        340 INIT_ACK
  267 2.803227  127.0.0.1   127.0.0.1       SCTP        376 COOKIE_ECHO DATA
  268 2.803284  127.0.0.1   127.0.0.1       SCTP         68 COOKIE_ACK SACK
  269 2.804371  127.0.0.1   127.0.0.1       SCTP        100 DATA
  270 2.804391  127.0.0.1   127.0.0.1       SCTP         64 SACK
  316 6.185637  127.0.0.1   192.168.12.161  SCTP        100 HEARTBEAT
  317 6.185673  127.0.0.1   127.0.0.1       SCTP        100 HEARTBEAT_ACK
  323 8.850384  127.0.0.1   127.0.0.1       SCTP         56 SHUTDOWN
  324 8.850413  127.0.0.1   127.0.0.1       SCTP         52 SHUTDOWN_ACK
  325 8.850428  127.0.0.1   127.0.0.1       SCTP         52 SHUTDOWN_COMPLETE

Frame 267: 376 bytes on wire (3008 bits), 376 bytes captured (3008 bits)
Linux cooked capture
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Stream Control Transmission Protocol, Src Port: 46620 (46620), Dst Port: 36421 (36421)
Data (46 bytes)
```

Figure 11: The setup-request setup-response procedures from the implemented E2 agent and RIC, followed by a graceful shutdown. O-RAN mandates the use of SCTP protocol and the port number 36421. The setup-request sent by the agent is embedded in the COOKIE ECHO DA

E2SM development is still in its early stages, since O-RAN has only developed 3 E2SM: the E2SM-HW, the E2SM-KPI and the E2SM-NI. All of them have already been implemented, and the current focus is on the design of new SMs to fulfil the stringent requirements that xApps are envisioned to fulfil.

ORAN 7.2

The architecture of eNB/gNB using O-DU and O-RU is depicted in [23] and in Figure 12.

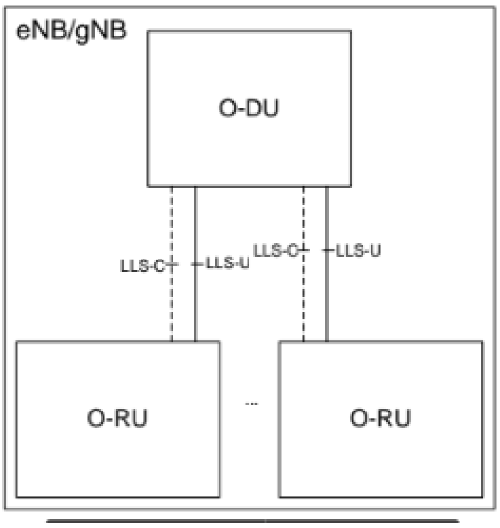


Figure 12: eNB/gNB architecture using O-DU and O-RU. Source [23]

The O-DU controls the operation of O-RUs as well as the real-time aspects of control & user plane communication. LLS-C (Low Layer Split) and LLS-U provide C-plane and U-plane over the LLS interface, respectively. There are two competing interests to be taken into account in the functional split front haul definition:

- Keep an O-RU as simple as possible allows to limit the size, weight, and power draw.
- The benefit of having the interface at a higher level tends to reduce the interface throughput with respect to a lower-level interface. On the other hand, the higher-level the interface, the more complex the O-RU tends to be.

The ORAN split 7.2x option is a functional split between the O-DU and O-RU described by the ORAN WG4 Fronthaul Working group specification [23] and reported in Figure 13.

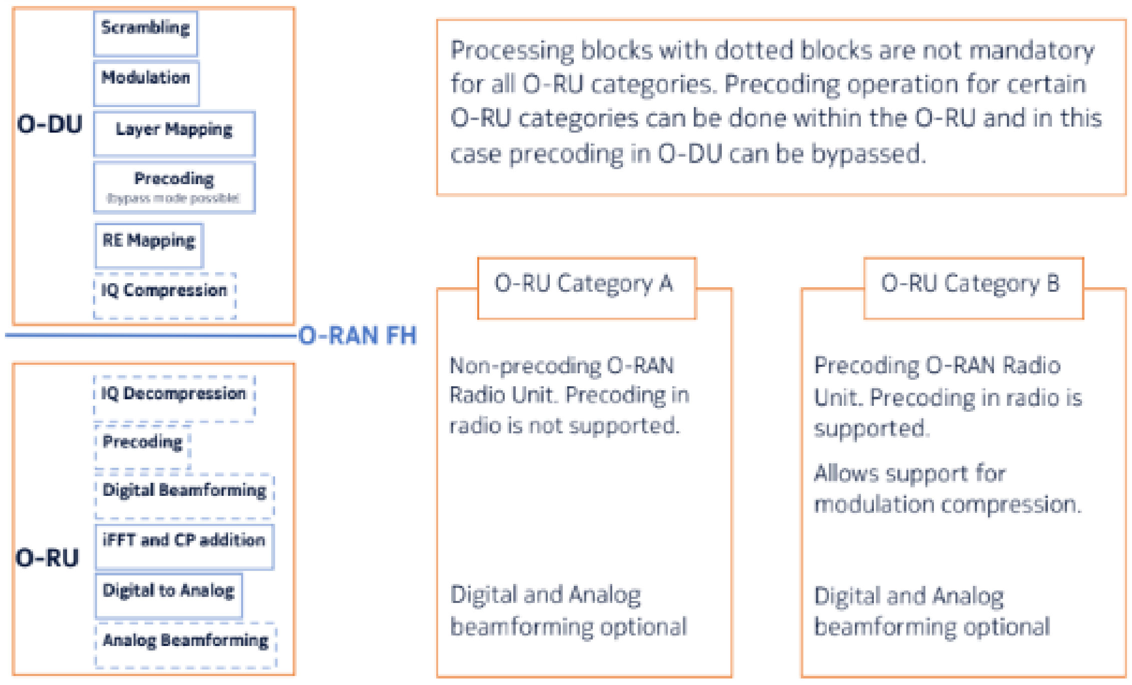

O-DU
Scrambling
Modulation
Layer Mapping
Precoding
(bypass mode possible)
RE Mapping
IQ Compression
O-RAN FH
O-RU
IQ Decompression
Precoding
Digital Beamforming
iFFT and CP addition
Digital to Analog
Analog Beamforming
Processing blocks with dotted blocks are not mandatory for all O-RU categories. Precoding operation for certain O-RU categories can be done within the O-RU and in this case precoding in O-DU can be bypassed.
O-RU Category A
Non-precoding O-RAN Radio Unit. Precoding in radio is not supported.
Digital and Analog beamforming optional
O-RU Category B
Precoding O-RAN Radio Unit. Precoding in radio is supported.
Allows support for modulation compression.
Digital and Analog beamforming optional

Figure 13: O-RAN 7.2x functional split. Source [23]

O-RAN O-DU and O-RU are logic nodes that contain a different subset of eNB/gNB functions. ORAN has selected a single split point, known as the ”7-2x” but allows a variation, with the precoding function to be either “above” the interface in the O-DU or “below” the interface in the O-RU depending on the selected O-RU category.

The O-DU is a logic node containing eNB/gNB functions of:

- Scrambling.
- Modulation.
- Layer mapping.
- Precoding.
- The possibility to perform the precoding operation in the O-RU instead of in the O-DU is not applicable for O-RUs category A.
- Resource element mapping.
- IQ compression (optional operation).

The O-RU is a logic node containing eNB/gNB functions of:

- iFFT and Cyclic Prefix (CP) addition.
- Digital to Analog conversion.
- IQ decompression (optional operation).
- Precoding (optional operation*).
- The precoding operation is optional only if the O-RU is category B*.
- O-RU category A does not support the precoding in RU, for this reason this feature should not be considered.
- Digital and Analog beamforming (optional operations).

In case of lower layer front haul based on split option 7-2x for DL and UL, the required external data flows (excluding M-plane) to exchange information between O-DU and O-RU can be categorized for User-Plane, C-Plane and Synchronization-Plane as depicted in Figure 14.

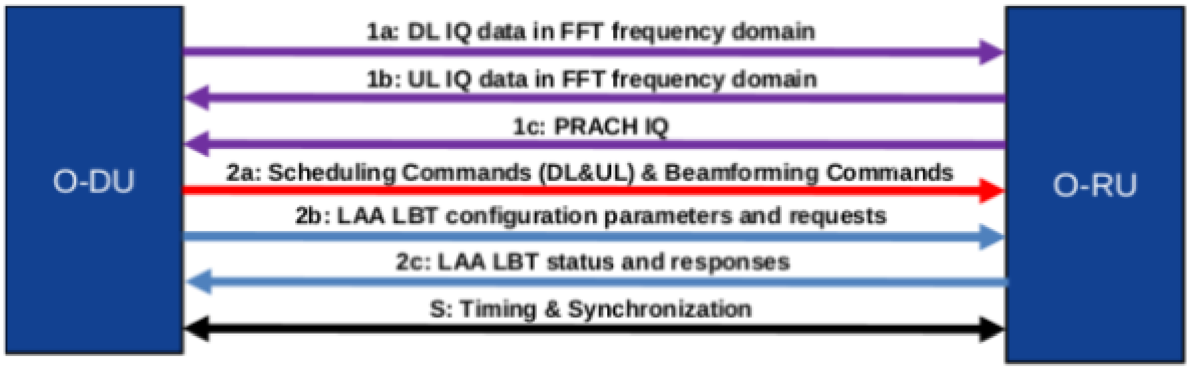

O-DU
O-RU
1a: DL IQ data in FFT frequency domain
1b: UL IQ data in FFT frequency domain
1c: PRACH IQ
2a: Scheduling Commands (DL&UL) & Beamforming Commands
2b: LAA LBT configuration parameters and requests
2c: LAA LBT status and responses
S: Timing & Synchronization

Figure 14: Low layer fronthaul data flow. Source [23]

O-DU and O-RU are generally positioned in different locations and the Ethernet protocol can be used as the transport mechanism for both C-plane and U-plane by using standard encapsulated Ethernet frames.

Delay parameters and latency requirements related to the C-plane and U-plane coordination are detailed in section 2.3 of [23] and the synchronization (S-plane) follows the options available over an Ethernet network (e.g., Frequency synchronization, Phase synchronization, Time synchronization).

ORAN 7.2 integration status

The integration of the ORAN 7.2 in the OAI software follows two main axes:

- The software integration of the O-RAN fronthaul library (ORAN FHI) into the OAI code. The ORAN FHI Library had been created. Currently, it is under integration with OAI 4G and 5G modem.
- The integration of the Xilinx T1 accelerator card [24], for the front haul and L1 optimization. We successfully run BBDEV (Wireless Base Band Device) test application on the Xilinx T1 card, and integrated BBDEV library into OAI as a shared library. Currently, we are writing the Low-Density Parity-Check offloading applications in OAI using BBDEV APIs as a proof of concept, and then move to the ORAN 7.2 interface. Figure 15 shows the layers involved in this integration task.

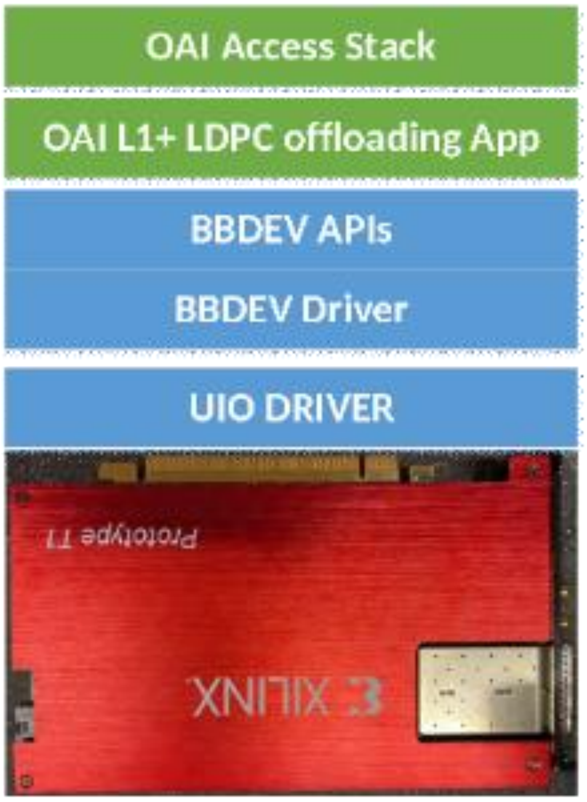


Figure 15: OAI Xilinx integration layers

The block diagram of the Xilinx T1 card is shown in Figure 15. It features two FPGA Zync ultrascale RFSOC for PHY offloading and Microprocessor System-On-Chip (MPSOC) for O-RAN 7.2 Fronthaul interface (FHI). It has one PCIe x16 interface that is bifurcated into two x8 interface, one for RFSOC and the other for MPSOC. Two 25G eCPRI are used for Fronthaul termination with 1588 timing stack, which are directly connected to the MPSOC.

2.1.2.3 Requirements' Mapping

The table below lists the satisfied requirements with respect to those indicated in D1.1 section 3.6.

No.	ID	Requirement	Note
1	REQ-NET-01	The network shall conform to 3GPP 5G network requirements, architectures and interfaces in both the 5GC and NG-RAN	Ongoing
2	REQ-NET-03	The system shall support O-RAN standard architecture & interfacing including the support of RIC VNFs	Done
3	REQ-NET-04	The system shall support 3GPP release Rel. 15 SA specifications	Ongoing
4	REQ-NET-05	The system may support 3GPP release Rel. 16 SA specifications	Planned
5	REQ-NET-07	The system shall support control and user plane separation in the NG-RAN	Planned
6	REQ-NET-20	The system shall support operation in the TS 38.101-1 Frequency Range 1 FR1 (410-7125 MHz) bands including support for bands n77 and n78.	Ongoing
7	REQ-NET-28	Open Fronthaul M-Plane interface shall be protected	Planned. Through IPsec
8	REQ-NET-30	3GPP F1 interface shall be protected	Planned. Through IPsec
9	REQ-NET-42	The system may provide integrity protection of the user data between the UE and the gNB	Ongoing
10	REQ-MAN-01	The architecture shall provide 5G network slicing to support services with diverse QoS requirements	Planned
11	REQ-MAN-02	The solution should be able to establish network slices that are extended from the RAN (ingress point) up to the Core of the network (egress point).	Planned
12	REQ-MAN-03	The system shall provide a way to create, instantiate, update and delete Network Slices	Planned
13	REQ-MAN-33	The system shall support QoS solicitations from the MCS	Planned

Table 2: Requirements' mapping of the DU

2.1.2.4 Technical Challenges

One of the main technical challenges is related to the integration of O-RAN 7.2 fronthaul interface. Currently in the market is difficult to find commercially available RUs compliant with the O-RAN 7.2 since each vendor has its specific control and management operation. In addition, is not yet clear if O-RAN specification is able to cover vendor-specific RU configuration or management, this may imply the need of some custom and/or static configuration outside of the scope of O-RAN 7.2 control plane.

2.1.2.5 Plan for the Integration

The approach we took is to have parallel integration for the three considered interfaces, namely O-RAN 7.2, F1, and E2. The integration plan is summarized as follows:

- Integration of in the fronthaul interface:
    a) Start integrating O-RAN fronthaul library into OAI.
    b) Explore the integration of the Xilinx T1 cards and create a high-level wrapper so that other accelerator cards could be easily integrated later.
    c) IoT of OAI with ORAN FHI (and possibly with the T1 card) with O-RAN RU simulator, and in the second stage with a 3rd party commercial O-RAN 7.2 RU on both UP and CP
    d) Compare and exchange traces with Runel RU. Agree on a RU configuration.
    e) Interoperability testing of OAI with ORAN 7.2 (and possibly with the Xilinx T1) with Runel RU.
- Integration of E2 Agent in DU and CU:
    a) Integration of OAI E2 agent with the O-RAN RIC using helloworld (HW) and KPM service models.
    b) Integration of OAI E2 agent with the OAI FlexRIC using helloworld (HW) and KPM service models version 1.
    c) Integration of OAI E2 agent with ACC dRAX controller using helloworld (HW) and KPM service models version 1.
- Integration of F1 interface between CU and DU:
    a) Interoperability testing of OAI CU and OAI DU with F1-C.
    b) Interoperability testing of OAI CU and OAI DU with F1-U.
    c) Interoperability testing of ACC CU and OAI DU with F1-C.
    d) Interoperability testing of ACC CU and OAI DU with F1-U.

2.1.3 Centralized Unit

2.1.3.1 Description and Motivation

The CU, or Central Unit, implements the ‘L3’ part of the NG-RAN gNB. The CU interfaces to the gNB DU via the F1 mid-haul interface and the mobile core (5GC) via the N2 and N3 backhaul interfaces (NG-C/NG-U).

The CU is split between the control plane and user plane:

- CU-CP: Control-plane of CU, implementing the RRC functions as described in [2] chapter 7. Communicated to DU via F1-C and to the 5GC control-plane via N2.
- CU-UP: User plane function of CU, implementing the PDPC ([TS 38.300 [2] section 6.4) and SDAP (mapping of PDU sessions and their QoS flows to the access stratum

DRB Data Radio Bearers, [2] section 6.5). See Figure 16. The CU-CP controls the CU-UP via the E1 interface.

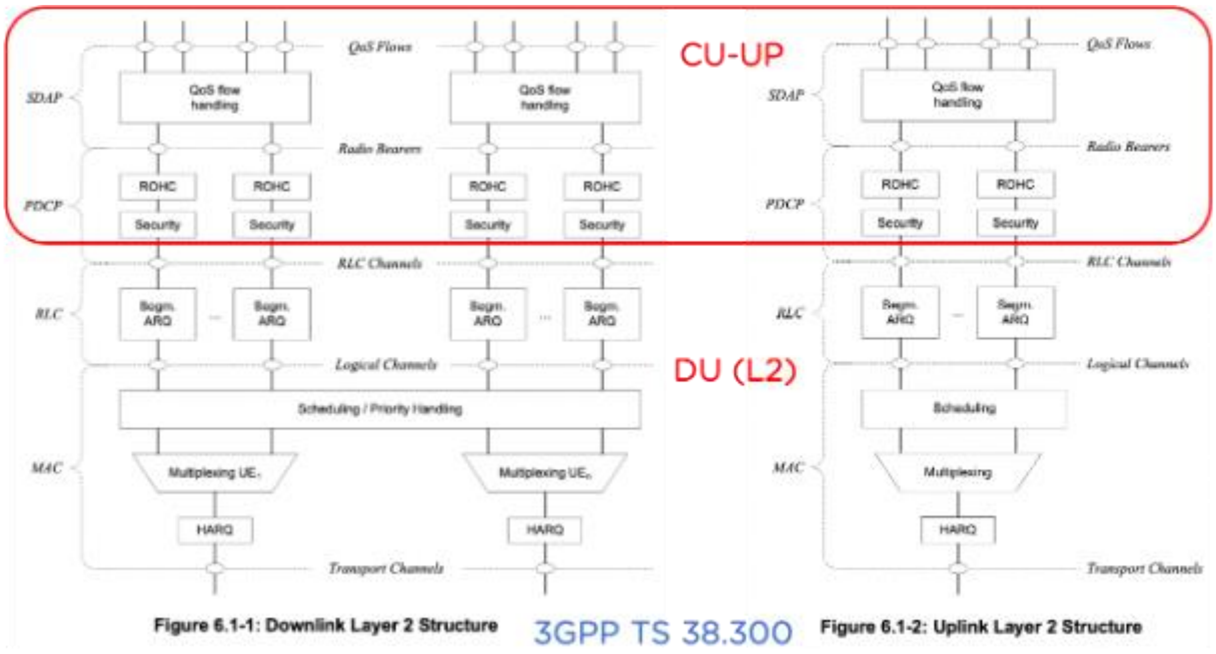


Figure 16: CU-UP functions, part of 3GPP TS 38.300 L2

A single protocol entity of SDAP is configured for each individual PDU session but it is an implementation decision of how to map this to distinct CU-UP container or Virtual Machine (VM) instances (and associated distinct E1, F1-U and N3 connections).

Affordable5G CU will create new CU-UP instances (and corresponding connections) per PLMN/SNPM IDs (for neutral hosting) and per network slice Single Network Slice Selection Assistance Information (S-NSSAI), as described in Affordable5G D1.2 [1] and shown below in Figure 17.

The Afforable5G CU is implemented as a cloud-native solution of multiple CNF (Cloud Network Functions) containers and Kubernetes [25] pods. The software runs on generic CPU architectures (currently x86 and ARM supported) that are deployable in a central or edge cloud or on dedicated edge infrastructure.

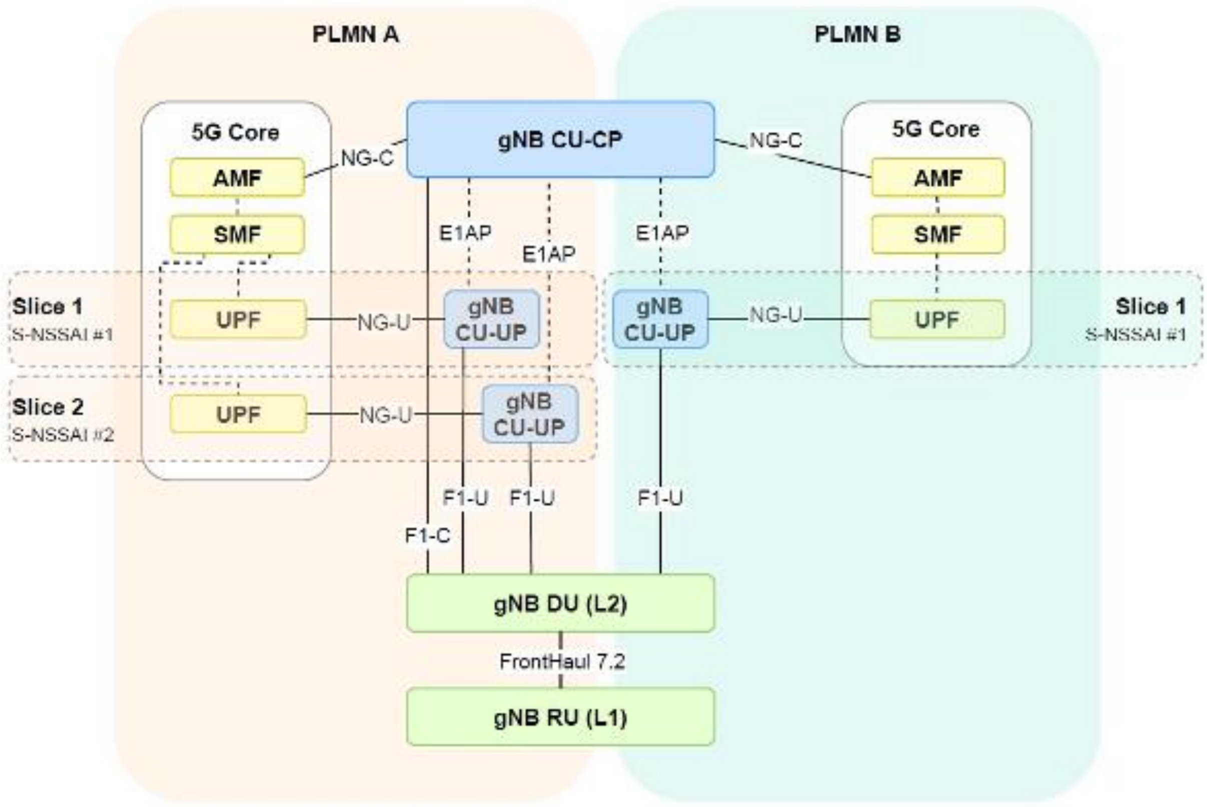


Figure 17: Afforrdable5G network slicing for multiple CU-UP instances for neutral host & slicing

Especially for URLLC slices (with Edge Compute and UPF/DN as the radio or metro edges), the CU-CP and CU-UP instances may be physically separated with the underlay network of the E1 interface potentially running over many different types of (trusted and untrusted) network topologies, illustrated in Figure 18.

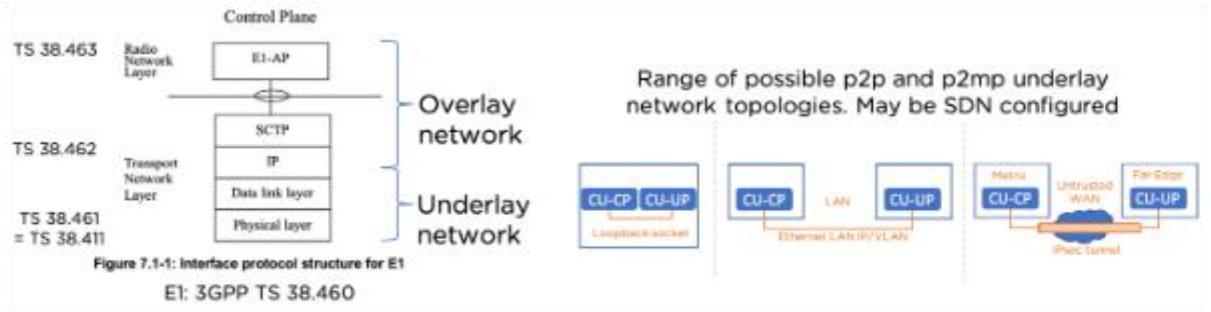


Figure 18: Possible separation of CU-CP and CU-UP via E1 underlay topologies

An O-RAN aligned Affordable5G CU is called an O-CU by O-RAN and implements the O-RAN E2 interface towards the Near-RT RIC as described in 2.1.4.

2.1.3.2 Interfaces

The CU interfaces to the gNB DU via the F1 mid-haul interface and the mobile core (5GC) via the N2 and N3 backhaul interfaces (NG-C/NG-U).

The CU is split between the control plane and user plane:

- CU-CP: Communicates to DU via F1-C and to the 5GC control-plane via N2. Controls the CU-UP via the E1 interface.
- CU-UP: Communicates to DU via F1-U and to the 5GC control-plane via N3.
- Both CU elements communicate to the O-RAN Near-RT RIC via the E2 interface and SMO via O1.

2.1.3.3 Requirements' Mapping

The table below lists the satisfied CU requirements with respect to those indicated in D1.1 section 3.6.

No.	ID	Requirement	Note
1	REQ-NET-01	The network shall conform to 3GPP 5G network requirements, architectures and interfaces in both the 5GC and NG-RAN	Done in CU for F1-U, F1-C, N2, N3, E1
2	REQ-NET-02	The system shall support operation as a 5G SNPN	Rel-15 done, Rel-16 ongoing.
3	REQ-NET-03	The system shall support O-RAN standard architecture & interfacing including the support of RIC VNFs	Ongoing, E2, O1
4	REQ-NET-04	The system shall support 3GPP release Rel 15 SA	Done
5	REQ-NET-05	The system may support 3GPP release Rel 16 SA	Planned
6	REQ-NET-07	The system shall support control and user plane separation in the NG-RAN	Done in CU-UP/CU-CP
7	REQ-NET-08	The system shall support configuration and placement of the CU-UP, UPF and DN per slice	Done in CU. Ongoing in SMO.
8	REQ-NET-09	The system shall support Multi-Operator CU and DU HW and NFVi	Ongoing, MOCN in CU
9	REQ-NET-12	The system shall support virtualized RAN functions	Done (cloud-native functions)
10	REQ-NET-14	The system shall support Multi Band and Multi Operator RRU (MORAN & MOCN)	MOCN ongoing in CU. MORAN planned
11	REQ-NET-17	The system shall support inter 5G cells mobility	Ongoing in CU
12	REQ-NET-18	The system shall support virtualized or cloud-native deployment & orchestration on open COTS hardware for NG-RAN	Done in CU

No.	ID	Requirement	Note
13	REQ-NET-22	In the NG-RAN, the RU/DU/CU network functions shall support HLS/LLS split architecture to support multi-vendor RAN using open interfaces.	Done in CU, HLS/F1
14	REQ-NET-29	3GPP E1 interface shall be protected	Planned. Underlay IP network via IPsec
15	REQ-NET-30	3GPP F1 interface shall be protected	Planned. Underlay IP network via IPsec
16	REQ-NET-39	The system shall support small outdoor cells capabilities	Ongoing
17	REQ-MAN-01	The architecture shall provide 5G network slicing to support services with diverse QoS requirements	Done for Rel-15. Ongoing for full implementation
18	REQ-MAN-02	The solution should be able to establish network slices that are extended from the RAN (ingress point) up to the Core of the network (egress point).	Planned
19	REQ-MAN-03	The system shall provide a way to create, instantiate, update and delete Network Slices	Done in CU. Ongoing via SMO slice manager (see 3.2)

Table 3: Requirements' mapping of the CU

2.1.3.4 Technical Challenges

Being in the ‘center’ of the RAN, communicating to 3rd party DUs and 5GCs, the biggest challenge is in the multi-vendor integration, where DU/5GC’s of different vendors implement different feature subsets of 3GPP Rel-15 and Rel-16 releases.

Additionally, the O-RAN E2 interface and network slicing models are still fairly immature leading to alternative implementations and further integration challenges.

2.1.3.5 Plan for the Integration

In Affordable5G, CU-CP is provided by ACC:

- Integration to DU (OAI from EUR, see 2.1.2) via F1-C is ongoing.
- Integration to 5GC N2 (from ATH, see 2.2) is ongoing.
- Integration to Near-RT RIC via E2 (from ACC, see 2.1.4) is fully functional but full conformance to the O-RAN E2 interface standard is work in progress.

- Integration to Service Management and Orchestration (SMO) via O1/O2 (from i2CAT, see 3.2) is fully functional but full conformance to the O-RAN E2 interface standard is work in progress.
- Integration to CU-UP via E1 is implemented.

In Affordable5G, the initial CU-UP is provided by ACC:

- Integration to DU (OAI from EUR, see 2.1.2) via F1-U is ongoing.
- Integration to 5GC N3 (from ATH, see 2.2) is ongoing.
- Integration to Near-RT RIC via E2 (from ACC, see 2.1.4) is fully functional but full conformance to the O-RAN E2 interface standard is work in progress.
- Integration to SMO via O1/O2 (from i2CAT, see 3.2) is fully functional but full conformance to the O-RAN E2 interface standard is work in progress.
- Integration to CU-CP via E1 is implemented.

In a later stage of the project, an alternative CU-UP will also be available from OAI/EUR. This integration has not yet been planned or started. This should be a drop-in replacement for the ACC CU-UP. While the 3GPP interfaces (F1-U, N3, E1) should be straightforward, the impact on the network slicing model (mapping of PLMN and Network Slice Selection Assistance Information (NSSAI) to CU-UP instances, shown in Figure 17) towards the SMO slice manager (see 3.2) and the E2 towards the Near-RT RIC (from ACC, see 2.1.4) may require special consideration and cause impact.

2.1.4 Near-RT RIC

2.1.4.1 Description and Motivation

The Affordable5G Near-RT RIC (RAN Intelligent Controller) is aligned to the O-RAN Alliance WG3 architecture [22] shown below in Figure 19.

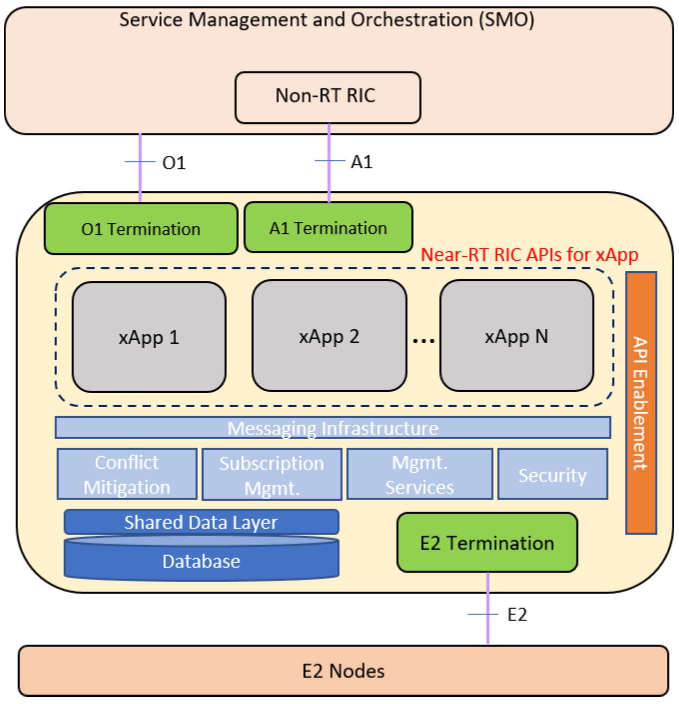


Figure 19: O-RAN WG3: Near-RT RIC Internal Architecture

The Near-RT RIC hosts the following functions:

- Database, which allows reading and writing of RAN/UE information.
- xApp subscription management, which merges subscriptions from different xApps and provides unified data distribution to xApps.
- Conflict mitigation, which resolves potentially overlapping or conflicting requests from multiple xApps.
- Messaging infrastructure, which enables message interaction amongst Near-RT RIC internal functions.
- Security, which provides the security scheme for xApps.
- Management services:
    - Fault management, configuration management, and performance management as a service producer to SMO.
    - Life-cycle management of xApps.
    - Logging, tracing, and metrics collection, which capture, monitor and collect the status of Near-RT RIC internals and can be transferred to external system for further evaluation.
- Interface Termination:
    - E2 termination, which terminates E2 interface from an E2 Node.
    - A1 termination, which terminates A1 interface from non-RT RIC.
    - O1 termination, which terminates O1 interface from SMO.
- Functions hosted by xApps, which allow services to be executed at Near-RT RIC and outcomes to be sent to E2 Nodes via E2 interface.
- API enablement function supporting capabilities related to Near-RT RIC API operations (API repository/registry, authentication, discovery, generic event subscription, etc.).

Within O-RAN, discussions are ongoing to standardize the APIs provided by the Near-RT RIC towards xApps. While the importance of this is understood within the industry, there are currently no agreed or published interface standards (things seem to be aligning around Facebook xApp SDK contributions but not yet standardized). While a standardized API is beyond the scope of the Affordable5G Near-RT RIC, the Accelleran dRAX will be tracking and aligning to this evolving standardization on the product roadmap.

Accelleran has developed a dRAX™ product line which consists of a 3GPP CU and O-RAN Alliance aligned Near-RT RIC, which is extensible through xApps. Accelleran has developed the dRAX xApps programming framework (as used in the Afforadble5G Near-RT RIC), based on highly efficient current industry best-practices patterns (like REST interfaces, Kubernetes and Helm charts, Python, etc), that is fully documented and supported towards 3rd party developers. The intention is to evolve this towards standardized interfaces, as these emerge and become widely accepted. The dRAX xApps Framework consists of the following components, which are shown in Figure 20:

- The xApp REST API
- The xApp Database
- The xApp Core

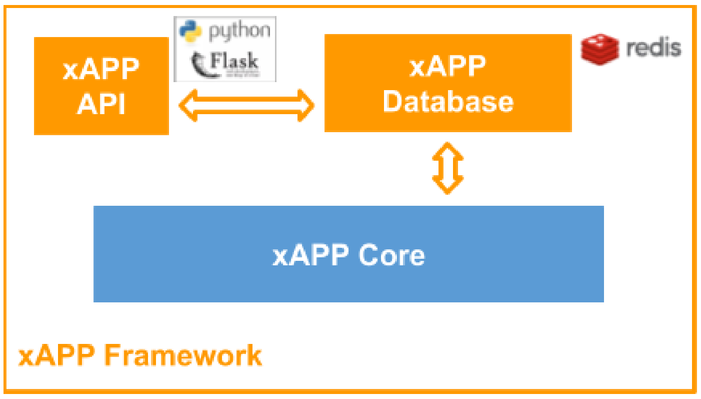


Figure 20: Accelleran Near-RT RIC xApp framework

2.1.4.1 Interfaces

The dRAX xApp framework is a modular solution written in Python and ready for xApp developers to directly use. This framework can be upgraded to add further functionality and/or rewritten in another language of the developer's choice. The xApp REST API is used for communication with the dRAX RIC. It is a prebuilt block by Accelleran, so that the developer does not have to code it. This REST API can be used to configure the xApp. This configuration is then saved in the xApp's local database. The database is also already prebuilt; it is based on Redis. Finally, the main part of the xApp Framework is the xApp Core. Even this block is almost entirely prebuilt, with helper functions that abstract the different possibilities within the dRAX RIC. The developer can just integrate their code within the xApp Core code and make use of this already existing code.

The xApp Core consists of several functional blocks, as shown in Figure 21. Most of these blocks are predefined and pre-programmed for the developers' convenience, such as Redis listener, settings, Kafka listener, in-queue, Kafka producer, out-queue and action taker.

The Redis listener is used to get notifications about changes to the xApp Database of the xApp framework. When a change in the configuration is detected, the Redis listener updates the configuration of the xApp in the Settings block. This block is used throughout the xApp to check the configuration parameters, some of which can also be real-time.

The Kafka listener is used to communicate with the dRAX Databus and retrieve data from it. This data is preprocessed and sent to the In-Queue.

The Kafka producer is a block that can create new Kafka messages and send them out to the dRAX Databus. It reads any new messages found in the Out-Queue and sends it to the dRAX Databus.

The Action taker is a block that can be called on an event-based manner. It contains the logic of sending commands to the dRAX system. There are two types of commands currently supported:

- Handover command - You can manually issue a handover command to move a specific UE from Cell_A to Cell_B.
- Sub-band masking - You can issue a command on a per-cell basis to mask certain sub bands of a cell, making them unavailable for use.

Finally, we have the processor block. This is where the developers actually input their code. This block reads each message from the In-Queue and makes them available to the developer. Following the processing step, the developer can send out their own messages that should be published on the dRAX Databus. This is done by simply putting the messages on the Out-Queue. Also, the developer can call on the Action taker to issue a handover or sub-band masking command.

The processor is the place where the developer can implement their logic or algorithms to process the data available on the dRAX Databus. Using such architecture, we enable the developer to focus only on their own functionality in the processor block, while the other blocks required for the communication and integration with dRAX are already pre-built.

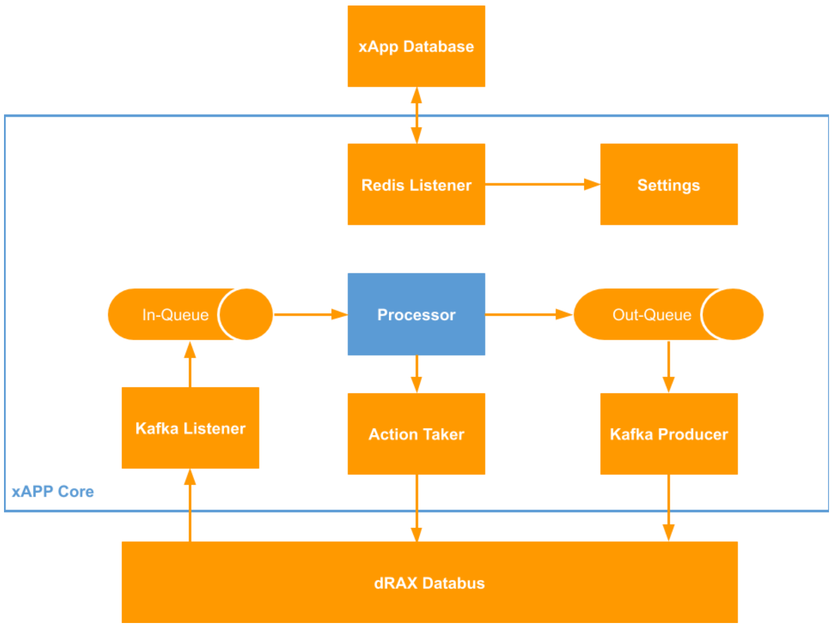


Figure 21: Accelleran Near-RT RIC xApp Core architecture and interfacing

2.1.4.2 Requirements' Mapping

The table below lists the satisfied CU requirements with respect to those indicated in D1.1 section 3.6.

No.	ID	Requirement	Note
1	REQ-NET-01	The network shall conform to 3GPP 5G network requirements, architectures and interfaces in both the 5GC and NG-RAN	Not applicable to Near-RT RIC (O-RAN extension to 3GPP)
2	REQ-NET-02	The system shall support operation as a 5G SNPN	Rel-15 done; Rel-16 ongoing
3	REQ-NET-03	The system shall support O-RAN standard architecture & interfacing including the support of RIC VNFs	Partially done in O1/O2. Ongoing in E2
4	REQ-NET-04	The system shall support 3GPP release Rel 15 SA	Done

No.	ID	Requirement	Note
5	REQ-NET-05	The system may support 3GPP release Rel 16 SA	Planned
6	REQ-NET-07	The system shall support control and user plane separation in the NG-RAN	Done
7	REQ-NET-08	The system shall support configuration and placement of the CU-UP, UPF and DN per slice	Done by design in Near-RT RIC. Full URLLC support planned
8	REQ-NET-09	The system shall support Multi-Operator CU and DU HW and NFVi	Ongoing MOCN in RIC
9	REQ-NET-12	The system shall support virtualized RAN functions	Done (cloud-native functions)
10	REQ-NET-14	The system shall support Multi Band and Multi Operator RRU (MORAN & MOCN)	MOCN ongoing in RIC. MORAN planned
11	REQ-NET-17	The system shall support inter 5G cells mobility	Ongoing in RIC
12	REQ-NET-18	The system shall support virtualized or cloud-native deployment & orchestration on open COTS hardware for NG-RAN	Done
13	REQ-NET-22	In the NG-RAN, the RU/DU/CU network functions shall support HLS/LLS split architecture to support multi-vendor RAN using open interfaces.	Ongoing in CU, HLS/F1
14	REQ-NET-29	O-RAN Operations and Maintenance Interface (O1) shall be protected	Planned. Underlay IP network via IPsec
15	REQ-NET-27	O-RAN Management A1 interface shall be protected	Planned. Underlay IP network via IPsec
16	REQ-NET-31	The system shall be able to detect network failures and malfunctions	Planned. Telemetry ingested from NG-RAN via E2. Exported to SMO/MWDAF
17	REQ-NET-39	The system shall support small outdoor cells capabilities	Ongoing

No.	ID	Requirement	Note
18	REQ-MAN-01	The architecture shall provide 5G network slicing to support services with diverse QoS requirements	Done for Rel-15. Ongoing for full implementation
19	REQ-MAN-02	The solution should be able to establish network slices that are extended from the RAN (ingress point) up to the Core of the network (egress point).	Planned
20	REQ-MAN-03	The system shall provide a way to create, instantiate, update and delete Network Slices	Done in CU. Ongoing via SMO slice manager

Table 4: Requirements' mapping of the Near-RT RIC

2.1.4.3 Technical Challenges

As the Near-RT RIC integrates with multi-vendor CU/DU and SMO functions, and considering the evolving and still early-stage definition of interfaces like E2, the main technical challenges lay in the integration aspects and transparent support for multiple implementations.

2.1.4.4 Plan for the Integration

In Affordable5G, Near-RT RIC is provided by ACC:

- Integration to CU (see 2.1.3) via E2 is fully functional but full conformance to the O-RAN E2 interface standard is work in progress.
- Integration to DU (from EUR, see 2.1.2) via E2 has not yet started.
- Integration to SMO via O1/O2 (from i2CAT, see 3.2) is fully functional but full conformance to the O-RAN E2 interface standard is work in progress.

2.2 5G Core

2.2.1 Description and Motivation

The core network is the “brain” of a mobile network and oversees all its functionalities and processes that are not related to the radio access, like user authentication, subscriber data management, policy control, connectivity and mobility management, exposure of services towards application functions, and traffic forwarding between the radio access infrastructure and data networks. An E2E 5G system is such only if its core network fully complies with the 3GPP standards. Accordingly, Athonet is providing Affordable5G with a SA fully virtualised 5GC, tailored to the project’s requirements to serve at best the project’s use cases. The softwarised architecture of Athonet’s 5GC offers increased flexibility and adaptability. Its 3GPP-compliant (Network Function) NFs, fully developed and programmed in-house at Athonet, are implemented as containerized (Virtual Network Function) VNFs, with the separation of UP and CP as required by 3GPP.

At the moment of writing this deliverable, as reported in D1.2, Athonet has already developed a working SA 5GC based on a minimum set of required functionalities, inherited (and evolved) from the legacy 4G/5G-NSA Software Mobile Core. The VNFs that compose it are:

- The Access and Mobility Management Function (AMF).
- The Session Management Function (SMF).
- The User Plane Function (UPF).
- The Unified Data Management (UDM).
- The Authentication Server Function (AUSF).
- The Unified Data Repository (UDR).

At the same time, other core VNFs are being developed and will be made available to the project in the next months:

- The Policy Control Function (PCF).
- The Network Repository Function (NRF).
- The Network Slice Selection Function (NSSF).

Figure 22 shows the 5GC architecture, including all the VNFs listed above. It also highlights the reference points towards the RAN, the UE, and the data network, labelled as per the 3GPP standard notation [26]. Moreover, as part of the overall architecture showed in Figure 1, we remark that the apparatus will be interfaced with:

- The underlying infrastructure layer and in particular with the Core (Network Function Virtualization Infrastructure) NFVI.
- The management, orchestration & automation layer, providing monitoring information.

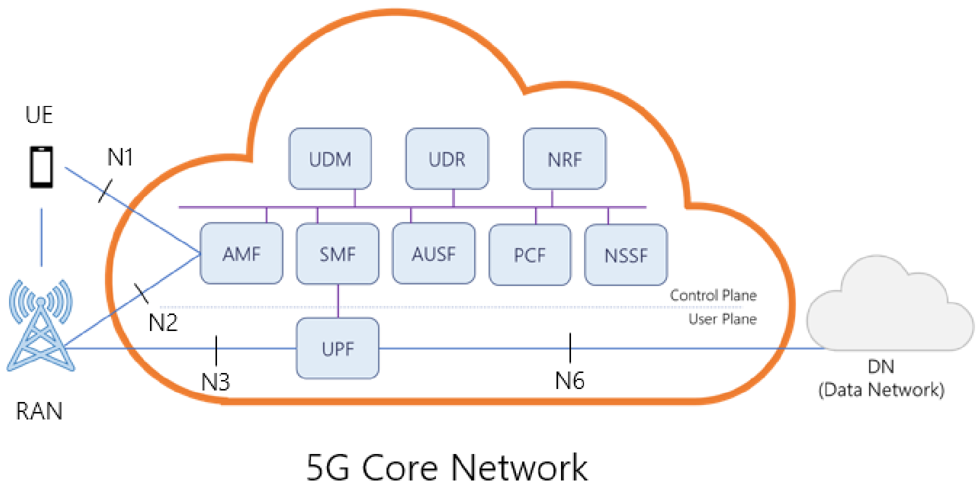


Figure 22. Athonet's 5G virtual core network functions.

In the following, we are going more into the detail of the relevant 5GC functionalities, and we will describe the available specific capabilities and roles of each provided VNF and those under development. As mentioned above, these are compliant with 3GPP's technical specifications (cf. [26]), and whenever opportune, they may be further updated according to the evolution of the standard itself in the scope of the project's use cases. Accordingly, details on the evolution of Affordable5G's CN will be provided in the next project deliverables.

2.2.2 Interfaces

Below the different functions of the CN are detailed together with their respective interfaces.

2.2.2.1 AMF

The AMF is responsible for handling connection and mobility management tasks. The AMF can be accessed by any number of gNBs in the RAN. gNBs are assumed to be locally configured to access the AMF.

The AMF includes the following functionalities:

- Termination of NAS (N1). The interface supports NAS signalling towards the AMF for Mobility Management functions and towards the SMF for Session Management functions.
- NAS signalling ciphering for confidentiality protection.
- NAS signalling integrity protection.
- Registration management to register UEs for 3GPP access over NR.
- Allocation of temporary identifier, the 5G-GUTI, to registered UEs.
- Connection management to establish and release N1 signalling connections between the UEs and the 5G Core.
- Reachability management, through Paging and handling of Periodic Mobility Registration update by UEs.
- Mobility Management, through user location tracking.
- Access Authentication using 5G Authentication and Key Agreement procedure.
- Access Authorization based on UE subscription data.
- Assignment of Allowed NSSAI to the UE based on UE subscription data.

Furthermore, interfacing with the PCF, the NRF, and the NSSF is under development, as well as handling of Xn-based handovers.

2.2.2.2 SMF

The SMF is primarily responsible for interacting with the decoupled data plane, handling Protocol Data Unit (PDU) sessions and managing session context with the UPF over the N4 reference point using the PFCP protocol.

The SMF includes the following functionalities:

- PDU Session establishment and release for PDU sessions of type IPv4 and IPv6
- Allocation and management of User Plane Tunnels between UPF and AN node.
- Dynamic UE IP address allocation & management
- Authorization of PDU Session establishment based on subscription data.
- Configures traffic steering at UPF to route traffic to proper destination.
- Termination of SM NAS messages.

- Handling of Downlink Data Notification from UPF.
- Initiator of AN specific SM information, sent via AMF over N2 to AN.
- PFCP session management over the N4 reference point with UPFs
- GTP-U connection management with the UPFs for PDU sessions of type IPv6.

Currently, the additional functionalities under development are the interfacing with the PCF and NRF, the handling of PDU Sessions of type Ethernet, and the handling of PCC rules.

2.2.2.3 UPF

The UPF is responsible for the processing and forwarding of PDUs between the UEs and the DNs.

The UPF includes the following functionalities:

- PDU Session point of interconnect to external DN over the N6 reference point.
- PDU Session point of interconnect to the NG-RAN over the N3 reference point.
- PFCP session management over the N4 reference point with the SMF.
- GTP-U connection management with the SMF for PDU sessions of type IPv6.
- Packet routing and forwarding of PDUs of PDU Sessions of type IPv4 and IPv6.
- Management of non-GBR QoS Flows.
- Downlink packet buffering.
- Downlink data notification triggering towards the SMF over the N4 reference point.

Traffic usage reporting, handling of PDU sessions of type Ethernet, and forwarding of "end markers" to the source NG-RAN node during Xn-based handovers are under development.

2.2.2.4 UDM

The UDM is a centralized way to control network user data. It includes support for the following functionalities:

- Generation of 5G AKA Authentication Vector.
- User Identification Handling (e.g., storage and management of SUPI for each subscriber in the 5G system).
- ARPF function to store and manage the UE credentials, including the long-term keys.
- SIDF function for the de-concealment of SUCIs
- Subscription management.

Interfacing with the NRF is under development.

2.2.2.5 AUSF

The AUSF performs the authentication function. The AUSF includes support for:

- Authentication for 3GPP access over NR.

- Providing the UE’s SUPI to AMF after the UE has been successfully authenticated.

2.2.2.6 UDR

The UDR is a converged repository of subscriber information. The UDR supports the following functionality:

- Storage and retrieval of subscription data by the UDM.
- Storage and retrieval of policy data by the PCF is under development.

2.2.2.7 PCF

The PCF is currently under development. It is responsible for the determination and communication of operator policy to the CNFs. The PCF will:

- Support unified policy framework for different aspects of network behaviour.
- Provide policy rules to CP functions to enforce them.
- Access subscription information relevant for policy decisions in a UDR.
- Permit the SMF to create an SM Policy Association in order to communicate the IP address assigned to the UE.
- Interface with AFs over the N5 reference point.
- Establish an association with authorized AFs and binds it to the UE’s PDU Session.
- Request the SMF to modify established PDU Sessions upon reception Application information from the AF. The PCF provides the indication in the form of PCC rules with indication of the QoS to enforce. The SMF translates the PCC rule to a set of policy enforcement records that it transfers to the UPF handling the PDU session. This might result in the establishment of additional QoS Flows in the PDU session.

2.2.2.8 NRF

The NRF is currently under development. It maintains an updated repository of all the 5G elements available in the operator's network along with the services provided by each of the elements in the 5GC. The NRF will include the following functionalities:

- It will support NF discovery function. It will also receive NF Discovery Request from NF instance and provide the information of the discovered NF instances to the requesting NF instance.
- It will support P-CSCF discovery (specialized case of AF discovery by SMF).
- It will maintain the NF profile of available NF instances and their supported services. The profiles are either provided through OAM functions or registered by the NF instances upon startup. The profiles include information on the S-NSSAIs supported by the NF instances.

2.2.2.9 NSSF

The NSSF, under development at the moment, has the role to assist the AMF with the selection of the Network Slice instances that will serve a particular device.

The NSSF will include the following functionalities:

- Selecting the Network Slice instances that may serve the UE based on the requested NSSAI.
- Determining the Allowed NSSAI.
- Determining the Configured NSSAI.
- Determining the AMF Set to be used to serve the UE.

The NSSF queries the UDM for UE subscription information and the NRF to determine the AMFs that can serve the Allowed NSSAI.

2.2.3 Requirement's Mapping

Overall, the features available so far are compliant with the list of requirements reported in the table below, defined in D1.1, Section 3.6. Whenever a requirement is not defined exclusively for the CN and involves other network or service components, we mean that Athonet's 5GC satisfies it only as far as the CN is concerned. The list may be updated along the project's lifetime, as far as the 5GC will integrate new functionalities.

No.	ID	Requirement	Note
1	REQ-NET-01	The network shall conform to 3GPP 5G network requirements, architectures and interfaces in both the 5GC and NG-RAN	Done
2	REQ-NET-02	The system shall support operation as a 5G SNPN	Done
3	REQ-NET-04	The system shall support 3GPP release Rel 15 SA	Done
4	REQ-NET-05	The system may support 3GPP release Rel 16 SA	Ongoing
5	REQ-NET-06	The system shall support control and user plane separation in the 5GC and deployment of the UPF (and DN) in both core and edge (MEC)	Done
6	REQ-NET-08	The system shall support configuration and placement of the CU-UP, UPF and DN per slice	Ongoing
7	REQ-NET-13	The system shall support virtualised CN functions	Done
8	REQ-NET-16	The Core Network should report System KPI to the monitoring system	Done
9	REQ-NET-17	The system shall support inter 5G cells mobility	Ongoing

No.	ID	Requirement	Note
10	REQ-NET-23	Security mechanisms shall span across the CN and RAN and shall be compliant to 5G security architecture	Done
11	REQ-NET-24	The system shall support authentication of the UEs and authentication of the network towards the user	Done
12	REQ-NET-25	Access to the private 5G network shall be restricted to authorised UEs and processes only	Done
13	REQ-NET-38	The system may support operation as PNI-NPN	Ongoing
14	REQ-NET-40	The system shall support remote management and separate management domain per operator	Done
15	REQ-NET-44	The system should support data collection from NFs residing on the core and access part of the 5GS	Ongoing
16	REQ-NET-56	A centralised solution should allow to register specific users (authentication) under specific roles (authorisation), while keeping a log of all access attempts to external reference points	Done
17	REQ-MAN-01	The architecture shall provide 5G network slicing to support services with diverse QoS requirements	Ongoing
18	REQ-MAN-02	The solution should be able to establish network slices that are extended from the RAN (ingress point) up to the Core of the network (egress point).	Ongoing
19	REQ-MAN-04	The system shall support several slices over the same infrastructure	Ongoing
20	REQ-MAN-06	The system shall support Multi-Access Edge Computing (MEC)	Done (up to data plane at the edge)
21	REQ-MAN-21	The system shall provide monitoring mechanisms to continuously check the performance and status of the active slices	Ongoing
22	REQ-MAN-22	The system shall provide monitoring mechanisms to continuously check the performance and status of the active VNFs	Done
23	REQ-MAN-25	The system shall provide monitoring mechanisms to continuously check the performance and status of its own system health	Done
24	REQ-MAN-26	The system shall ensure the isolation of the slices	Done

No.	ID	Requirement	Note
25	REQ-MAN-29	The system may support HW and SW security hardening	Done
26	REQ-MAN-30	Edge Infrastructure shall host lightweight version of instances deployed in the main core	Done
27	REQ-MAN-31	The system should support services running in lightweight VMs or Docker containers	Done
28	REQ-MAN-32	The solution should allow the deployment of end-to-end slices able to accommodate the requirements imposed	Ongoing
29	REQ-MAN-33	The system shall support QoS solicitations from the MCS	Ongoing
30	REQ-APP-18	The solution should allow monitoring of security-related events	Done

Table 5: Requirements' mapping of the 5GC

2.2.4 Technical Challenges

Among the numerous features of Athonet's 5GC that Affordable5G it is worth mentioning the possibility of distributing core NFs (and in particular the UPF) to the edge of the network, bringing them as close as possible to the radio access equipment and the end users. This is a key enabler for edge computing, since selected traffic coming from the data plane can be maintained local and routed directly towards the servers where edge applications run, thus implementing the system architectures standardized by the ETSI MEC ISG. Furthermore, 5GC's capability to handle network slicing will allow to deploy isolated and independent services with performance guarantees, whenever the project's use cases will require it.

2.2.5 Plan for the Integration

In particular, as mentioned above, the upgrading work on the core is an ongoing process and, in addition to the features described so far, Athonet is planning to provide within its 5GC support for the following functionalities:

- Depending on the type of transported PDU, multiple PDU session types binding the UEs and the DN (IPv4, IPv6, Ethernet).
- Voice over New Radio.
- 5G SA Mission-Critical Push-To-Talk (MCPTT).
- Intra-AMF Handover.
- Usage reporting.

Athonet's 5GC will serve both the Castellolí and the Malaga testbeds. In both sites, it is deployed as a "core-in-a-box" solution, that is, all the VNFs of the 5GC are running over a single piece of commercial-off-the-shelf HW. The chosen server is a Dell R640, with CentOS as operating system and VMware as hypervisor. The HW has the following technical features:

- Intel Xeon CPU Gold 6140 2.3G, 18C/36T.
- 128 GB RAM.
- 2 x 600 GB 15K rpm SAS 12Gbps (RAID 1).
- 2 x 32 GB microSD (RAID 1).
- 2 x Hot-plug, Redundant Power Supply (1+1), 750W.
- 2 x 10 GbE.
- 8 x 1 GbE.

3 MANAGEMENT, ORCHESTRATION AND AUTOMATION LAYER

3.1 Orchestrator

The virtualization nature of 5G mobile networks has enabled the development of several solutions (both commercial and open source) for network/application orchestration. Among these solutions, in Affordable5G, we have selected to employ NearbyOne orchestrator (provided by Nearby Computing) in Castelloli's platform and Open-Source MANO (OSM) in Malaga's platform. In the following sections, we provide the technical details for these orchestrators.

3.1.1 NearbyOne

3.1.1.1 Description and Motivation

End-to-end orchestration is crucial to meet the requirements of the applications while continuously adjusting platform and system parameters. NearbyOne Orchestrator, depicted in Figure 23, provides a unique and wide range of control knobs that enables the ability to dynamically adjust the QoS of the applications while providing excellent performance predictability in shared and constrained environments.

Automatic service assurance and operation is controlled by an orchestration engine that requires complex mechanisms to continuously control the configurations that ensure optimal service levels. It involves continuous observation of the operational status (inputs) of all components involved in the operation of a running service, as well as advanced mechanism (outputs) to adjust its performance and power consumption. The connection between the inputs and the outputs of the orchestration engine is known as its *control loops*.

Advanced orchestration mechanisms for Edge Computing require specific telemetry and fine-grain resource control mechanisms. Nearby Computing commercializes NearbyOne, an end-to-end orchestration engine with specific functionalities for Edge Computing ecosystems. NearbyOne can manage the lifecycle of infrastructure, the VNFs and the applications, controlling their deployment and associated resources. NearbyOne addresses all orchestration issues in a holistic and intertwined way, not as an aggregation of separate problems.

At the core of an orchestration engine there is the service placement function, which is a critical component responsible for deciding where to run workloads, how to assess their KPIs with respect to established Service Level Objectives (SLOs), and how to configure the services and the infrastructure to meet user expectations.

The service placement function makes decisions based on the following system-wide inputs:

- Scheduling parameters (available workloads, priorities, deadlines).
- Workload characteristics and performance/power models.
- Workload QoS measurements (including platform, network, and application loops).
- Workload established KPIs/SLOs.
- Platform inputs (resource consumption levels, power consumption levels, bottlenecks).
- System inputs (power budget, network latencies and bandwidth).

The placement function can perform actuations over the following control knobs:

- Workload parameters and configurations.

- Workload placement location (site).
- Platform-level control knobs related to the application placement (RDT, core pinning).
- Platform-level control knobs unrelated to the applications (adjust to available power budget).
- System-level reconfigurations (resource pooling using the redfish API).

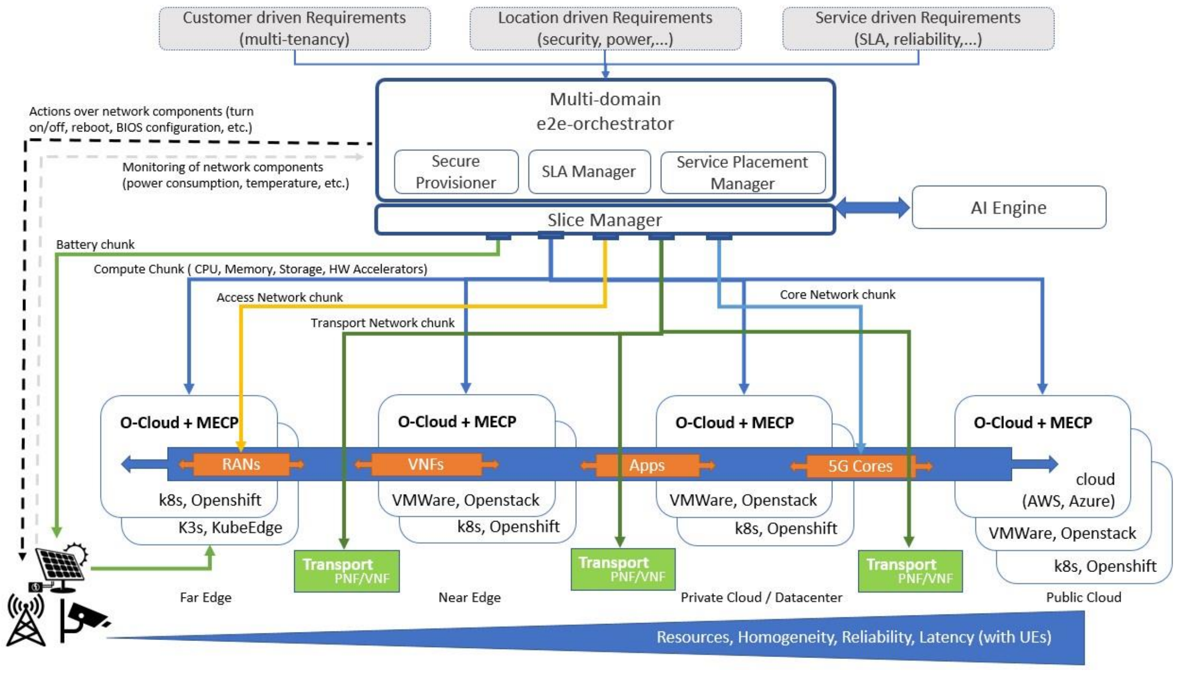

Customer driven Requirements (multi-tenancy)
Location driven Requirements (security, power,...)
Service driven Requirements (SLA, reliability,...)
Multi-domain e2e-orchestrator
Secure Provisioner
SLA Manager
Service Placement Manager
Slice Manager
AI Engine
Actions over network components (turn on/off, reboot, BIOS configuration, etc.)
Monitoring of network components (power consumption, temperature, etc.)
Battery chunk
Compute Chunk (CPU, Memory, Storage, HW Accelerators)
Access Network chunk
Transport Network chunk
Core Network chunk
O-Cloud + MECP
RANs
VNFs
Apps
5G Cores
k8s, Openshift
K3s, KubeEdge
VMWare, Openstack
cloud (AWS, Azure)
Transport PNF/VNF
Far Edge
Near Edge
Private Cloud / Datacenter
Public Cloud
Resources, Homogeneity, Reliability, Latency (with UEs)

Figure 23: System architecture of NearbyOne

3.1.1.2 Interfaces' Description

To manage an affordable 5G network, an end-to-end orchestrator needs to be responsible to integrate with the telco functions (5GC) of the network. ETSI specifies this integration in ETSI MEC standard, corresponding to the "MEC Orchestrator" (MEO), the "MEC Platform Manager" (MEPM), and the "MEC Platform" (MEP) components [27].

The multi-access edge orchestrator is the core functionality in MEC system level management. It is responsible for:

- Maintaining an overall view of the MEC system and its components (i.e., MEC hosts, available resources, available MEC services, and topology).
- Onboarding applications, i.e., checking their integrity and authenticity, validating application rules and requirements, while keeping a record of onboarded application packages, and preparing the VIMs to handle the applications.
- Selecting MEC host(s) for instantiating the application based on parameters, such as available resources, latency, etc.
- Triggering application instantiation, termination, and migration.

The MEC platform manager is responsible for:

- Lifecycle management of applications.

- Providing element management functions to the MEC platform.
- Managing the application rules, such as traffic rules, service authorization, DNS configuration.

Some of the interfaces are internal, while others are exposed northbound to the Operation Support Systems (OSS)

- OSS-facing interfaces (externally exposed):
    - Mm1: This interface is between the MEO and the OSS and is used for triggering the instantiation/termination of MEC applications.
    - Mm2: This interface is between the OSS and the MEPM and is used for the MEC platform configuration, fault, and performance management.
- Internal interfaces:
    - Mm3: It is between the MEO and the MEPM and it is used for the application lifecycle management, application rules and requirements and monitoring the available MEC services.
    - Mm4: It is between the MEO and the VIM and it is used to manage virtualized resources of the MEC host.
    - Mm5: It is between the MEPM and the MEC platform and it is used to perform platform configuration, configuration of the application rules, application lifecycle support procedures, management of application migration, etc.
    - Mm6: It is between the MEPM, and the VIM and it is used to realize the application lifecycle management.
    - Mm7: It is between the VIM and the virtualization infrastructure, and it is used to manage the virtualization infrastructure.
    - Mm8: It is between the user application lifecycle management proxy and the OSS, and it is used to handle device applications requests for running applications in the MEC system.
    - Mm9: it is between the user application lifecycle management proxy and the MEO and it is used to manage MEC applications requested by device application.

Finally, the MEP is responsible for:

- Offering an environment where the MEC applications can discover, advertise, consume and offer MEC services.
- Receiving traffic rules from the MEPM, applications, or services, and instructing the data plane accordingly.
- Receiving DNS records from the MEPM and perform DNS configuration.
- Hosting MEC services.

In addition, the functionality of the MEP is provided through three well-defined interfaces:

- Mp1: It is between the MEP and the MEC applications and provides service registration, discovery, and communication support. It also provides application availability, support for session state relocation, traffic rules and DNS rules activation, etc. This interface can be used for consuming and providing service specific functionality.
- Mp2: It is between the MEP and the data plane of the virtualization infrastructure, and it is used to instruct the data plane with regard to traffic routing.
- Mp3: It is an interface between MEC platforms to facilitate inter-MEP control communication.

3.1.1.3 Requirements' Mapping

With regard to the requirements on pilots and use cases identified in D1.1, the NearbyOne E2E-orchestrator satisfies or contributes to satisfying the ones in the table above:

No.	ID	Requirement	Note
1	REQ-MAN-01	The architecture shall provide 5G network slicing to support services with diverse QoS requirements.	Planned
2	REQ-MAN-02	The solution should be able to establish network slices that are extended from the RAN (ingress point) up to the Core of the network (egress point).	Planned
3	REQ-MAN-03	The system shall provide a way to create, instantiate, update and delete Network Slices.	Planned
4	REQ-MAN-04	The system shall support several slices over the same infrastructure.	Planned
5	REQ-MAN-05	The system shall support management of slices in an end-to-end fashion.	Planned
6	REQ-MAN-06	The system shall support Multi-Access Edge Computing (MEC)	Done
7	REQ-MAN-07	The system shall support MEC deployment topology in the RAN.	Planned
8	REQ-MAN-08	The system shall support the provision of the MEC stack installation and configuration.	Ongoing
9	REQ-MAN-09	The system shall support orchestration of services as well as lifecycle management.	Done
10	REQ-MAN-10	The service orchestrator shall be connected to the system monitoring and to the infrastructure involved in the necessary actions.	Ongoing
11	REQ-MAN-11	The system shall provide an interface to manage the lifecycle of the edge server.	Done
12	REQ-MAN-12	The system shall be able to configure the HW for specific workloads.	Ongoing
13	REQ-MAN-13	The system shall provide an interface to manage the lifecycle of the VNF.	Ongoing
14	REQ-MAN-14	The system shall support VNF lifecycle management at cloud-level platforms as well as low-end devices.	Ongoing
15	REQ-MAN-18	The solution should provide an environment for running software for data processing and service provisioning of multiple NFV services.	Ongoing
16	REQ-MAN-19	The solution should be able to provide real-time KPIs to the Orchestration platform mainly related to the QoS of services in order to guarantee SLAs.	Planned

No.	ID	Requirement	Note
17	REQ-MAN-20	The slice shall respond to application service requirements.	Planned
18	REQ-MAN-21	The system shall provide monitoring mechanisms to continuously check the performance and status of the active slices.	Planned
19	REQ-MAN-22	The system shall provide monitoring mechanisms to continuously check the performance and status of the active VNFs.	Ongoing
20	REQ-MAN-23	The system shall provide monitoring mechanisms to continuously check the performance and status of the active edge servers.	Done
21	REQ-MAN-24	The system shall provide monitoring mechanisms to continuously check the performance and status of the VIMs.	Done
22	REQ-MAN-25	The system shall provide monitoring mechanisms to continuously check the performance and status of its own system health.	Done
23	REQ-MAN-26	The system shall ensure the isolation of the slices.	Planned
24	REQ-MAN-27	The system shall support the provision of different bare metal OSto the edge servers.	Done
25	REQ-MAN-28	The system shall detect VNF health-issues and trigger pre-defined recovery actions.	Planned
26	REQ-MAN-31	The system should support services running in lightweight VMs or Docker containers.	Done
27	REQ-MAN-32	The solution should allow the deployment of end-to-end slices able to accommodate the requirements imposed.	Ongoing
28	REQ-MAN-37	The system shall support multiple NFVI geographically distributed PoP.	Done
29	REQ-MAN-38	The system shall on-board and provision off-the-shelf edge servers with minimal intervention.	Done
30	REQ-MAN-40	The solution should be highly efficient in terms of energy consumption, computing resources and bandwidth.	Planned
31	REQ-MAN-41	The system may use IPMI for remote edge server management	Done

Table 6: Requirements' mapping of the E2E-orchestrator

Notice that the requirements stated above refer to both the whole Affordable5G architecture and the specific Pilot test (and demonstration) where the NearbyOne E2E-orchestrator is involved.

3.1.1.4 Technical Challenges

This module was already deployed and working in Cellnex' Castelloli circuit before the project started. Among the enhancements achieved so far in the context of the Affordble5G, we have integrated into the orchestrator the basic required functionalities to provision the Lenovo Thinksystems SE350 Servers that will be used as edge nodes in Castelloli. This new hardware is specifically designed and built for the edge and is already deployed and available in Castellolí. As part of this initial set of functionalities, the servers can automatically be provisioned from the orchestrator with different Operative Systems and software/Hardware configuration that can be manually tuned by the user, or this can be configured to happen dynamically as part of the workloads and their monitored KPIs. Additionally, the orchestrator was initially deploying a Acceleran's dRax v1 for the controller and cells' firmware. Accelleran dRax and cells have been upgraded to v2.0 with the new capabilities that will be required in Affordable and software upgrades adding support for exposing the metrics required by the AI/ML engines.

All the data that is being gathered from the different orchestrated layers is already available in a Prometheus database and exposed to the other partners involved in Affordable5G for the training of the AI/ML models that will be developed as part of the project.

3.1.1.1 Plan for the Integration

During the next months we will integrate the different containerized VNFs and Applications that need to be orchestrated (ideally as helm charts, or k8s service definitions), collaborating with the partners developing them to get the most benefits from the different acceleration techniques and QoS options enabled by the orchestrator. This will also include the integration with the i2CAT Slice Manager that will handle the RAN chunk of the slice and the integration of the new control loops to be fed by the AI/ML engines provided by UoA/Atos and i2CAT. This may include adding some tags/labels to the KPIs provided by the orchestrator to make it to use by the AI/ML engines.

From our past experiences some of the expected technical challenges include the management of licenses in the VNF/applications, e.g., if the VNF/Application requires a license to run, and the orchestrator is expected to deploy the application dynamically, the VNF/Application should provide a mechanism to be able to load them without no manual intervention, to make the most profit of the orchestration layer.

Also from past experiences, some VNF/Applications that are already containerized and can be deployed with no issues in the partner's development environment, require host privileges and/or access to the host network. To follow the best practices in shared cloud environments that should be avoided but in case it is required by some of the partners and there is a conflict with something else we will need to find a solution.

3.1.2 OSM (Open Source MANO)

3.1.2.1 Description and Motivation

The current release of OSM (Release Nine) allows for the deployment of Kubernetes Network Functions on Kubernetes clusters. Clusters are connected to a Virtual Infrastructure Manager, which acts as the deployment target unit of OSM for KNFs. While it is possible to target specific Virtual Infrastructure Managers (VIMs) for the deployment of KNFs, it is currently not possible to place KNFs (and their individual Kubernetes Deployment Units) to specific nodes of a given

Kubernetes cluster (see Figure 24). This is a big limitation in how OSM could be used alongside Kubernetes, given that it would force users to rely purely on VIMs for placement or set up affinities and anti-affinities in the helm charts and thus outside of the OSM KNF/NS descriptors (Figure 25).

The goal is to enhance OSM's KNF deployment to support placement of KNFs on specific K8s node, as defined in the KNF or NS descriptors provided to OSM. In addition, we aim to propose this enhancement as a new feature for OSM directly to the OSM Technical Steering Committee, in order to make this part of an official future release of OSM. By working with the OSM Technical Steering Committee we ensure the enhancement is compliant with the OSM developer's guidelines, fits within their existing testing and integration pipeline, and in general fits within their future plans for OSM.

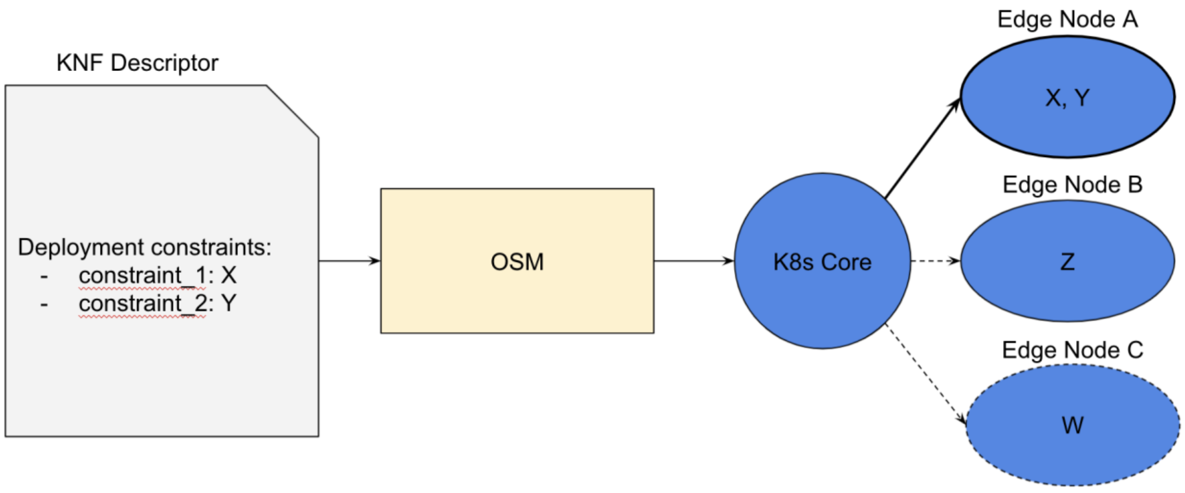


Figure 24: Proposed functionality, from descriptor constraints to deployment on the correct nodes

The challenge with such an approach is that the process of proposing a feature, approving a feature, finalizing a design for the implementation, and implementing the feature (including the required automated tests) for OSM is a time-consuming process, given that said feature comes from external developers, and thus communication with the OSM Technical Steering Committee can be slow. This without taking into account the design and implementation of the feature itself, which proves fairly challenging without a deep understanding of OSM's codebase. While the documentation for OSM is detailed in terms of explaining its functionality, documentation for developers is somewhat lacking (and outdated/incomplete in some areas). This means relying on direct communication with the developers and the overall OSM community for support, which also comes slow.

So far, the feature has been discussed and formalized with the OSM Technical Steering Committee and has been slated for official approval. The design has been partially discussed with the developers, and a plan was outlined by us. The goal is to make use of node selectors and potentially affinities and anti-affinities in Kubernetes to target specific nodes and expand the Information Model of OSM to include the definition of labels that specify these deployment constraints. By having this as part of the OSM IM we provide the option for defining placement directly in the KNF packages rather than confining it to the helm charts or juju bundles. Expanding the IM of OSM would also require extending OSM's Northbound API to account for the placement labels being introduced, and OSM's Lifecycle Manager module to provide this new information to the K8s clusters registered to the target VIMs.

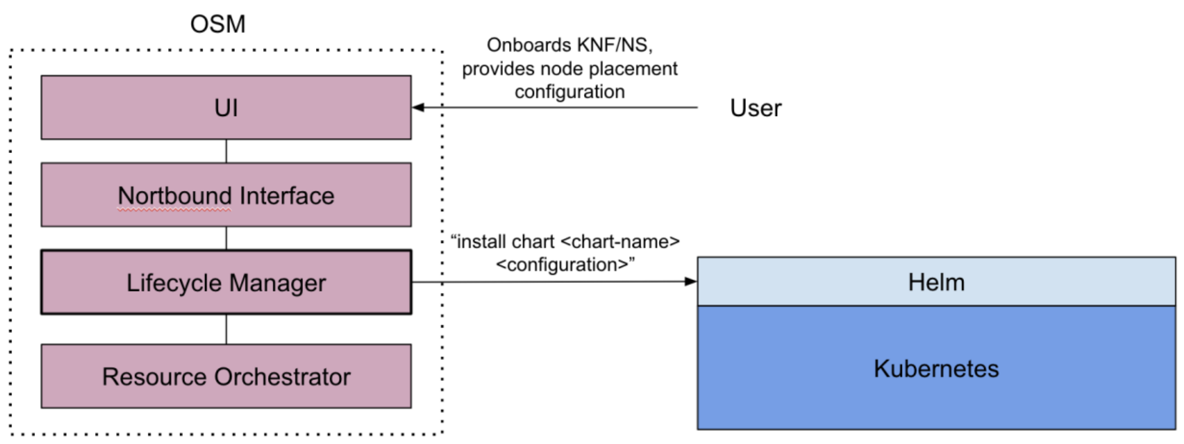

OSM
UI
Nortbound Interface
Lifecycle Manager
Resource Orchestrator
Onboards KNF/NS, provides node placement configuration
User
"install chart <chart-name> <configuration>"
Helm
Kubernetes

Figure 25: User onboards a KNF/NS from the OSM UI (also possible directly from the northbound interface), and this placement configuration is passed to the lifecycle manager, which in turn installs the corresponding helm charts whilst also providing the placement

The current plan is to work alongside the OSM TSC to implement this feature in a way that is compliant with OSM's guidelines and offers functionality that is in line with OSM's future plans. However, given the slow response times of the OSM developers we are proceeding with a separate implementation of the placement feature with less input from the committee for the sake of providing at the very least a "proof of concept" of this functionality for a standalone demo using our own custom-built branch of OSM. As we receive more feedback from the OSM developers we will re-align the design and implementation with their proposals, in order to eventually work towards this becoming a feature in a future OSM release (at the moment aiming for release Eleven of OSM).

3.1.2.2 Interfaces

OSM is made of a series of components that provide different functionalities to the platform.

Self-explanatory are the OSM Light UI, providing a web interface for users to log in (authenticating with Keystone) and interact with the platform, and the Northbound Interface (NBI), which provides a RESTful service acting as the primary northbound API for OSM.

The Lifecycle Manager (LCM), as per name, is the component responsible for managing the lifecycle of the OSM resources: deploying Network Services and their corresponding Network Functions, down to their individual Deployment Units, and maintaining their configuration, state, operations and so forth. It relies on Mongo for storing data.

The Resource Orchestrator (RO) is the primary component for orchestrating the resources within OSM, which are instantiated by the LCM. It relies on a mysql database for storing data. Communication between the LCM and RO modules, as all modules in OSM, happens primarily through Kafka topics.

On top of these modules, OSM provides a monitor component (using Prometheus) and a policy manager for governing resource access.

3.1.2.3 Requirements' Mapping

As per the requirements identified in Deliverable 1.1, OSM stands as follows:

No.	ID	Requirement	Note
1	REQ-MAN-03	The system shall provide a way to create, instantiate, update and delete Network Slices.	Ongoing
2	REQ-MAN-04	The system shall support several slices over the same infrastructure.	Ongoing
3	REQ-MAN-05	The system shall support management of slices in an end-to-end fashion.	Ongoing
4	REQ-MAN-06	The system shall support Multi-Access Edge Computing (MEC)	Ongoing
5	REQ-MAN-07	The system shall support MEC deployment topology in the RAN.	Ongoing
6	REQ-MAN-08	The system shall support the provision of the MEC stack installation and configuration.	Ongoing
7	REQ-MAN-09	The system shall support orchestration of services as well as lifecycle management.	Done
8	REQ-MAN-10	The service orchestrator shall be connected to the system monitoring and to the infrastructure involved in the necessary actions.	Ongoing
9	REQ-MAN-11	The system shall provide an interface to manage the lifecycle of the edge server.	Ongoing
10	REQ-MAN-12	The system shall be able to configure the HW for specific workloads.	Ongoing
11	REQ-MAN-13	The system shall provide an interface to manage the lifecycle of the VNF.	Done
12	REQ-MAN-14	The system shall support VNF lifecycle management at cloud-level platforms as well as low-end devices.	Done
13	REQ-MAN-18	The solution should provide an environment for running software for data processing and service provisioning of multiple NFV services.	Ongoing
14	REQ-MAN-22	The system shall provide monitoring mechanisms to continuously check the performance and status of the active VNFs.	Ongoing
15	REQ-MAN-23	The system shall provide monitoring mechanisms to continuously check the performance and status of the active edge servers.	Ongoing
16	REQ-MAN-24	The system shall provide monitoring mechanisms to continuously check the performance and status of the VIMs.	Ongoing
17	REQ-MAN-25	The system shall provide monitoring mechanisms to continuously check the performance and status of its own system health.	Ongoing
18	REQ-MAN-27	The system shall support the provision of different bare metal OS to the edge servers.	Ongoing

No.	ID	Requirement	Note
19	REQ-MAN-31	The system should support services running in lightweight VMs or Docker containers.	Done
20	REQ-MAN-38	The system shall on-board and provision off-the-shelf edge servers with minimal intervention.	Ongoing

Table 7: Requirements' mapping of OSM

3.1.2.4 Technical Challenges

Part of many Cloud Native infrastructures today is the concept of Infrastructure as Code: treating your Cloud infrastructure deployment the same way as code for a piece of software. This means having the infrastructure defined as code, like for example in a descriptor file, and stored in a repository. This repository is "watched" by a Continuous Deployment operator, which ensures that the state of the infrastructures is in sync with what is described in the repository. This approach enables developers to maintain their infrastructures like they maintain code, including the common GitOps workflow (see Figure 26).

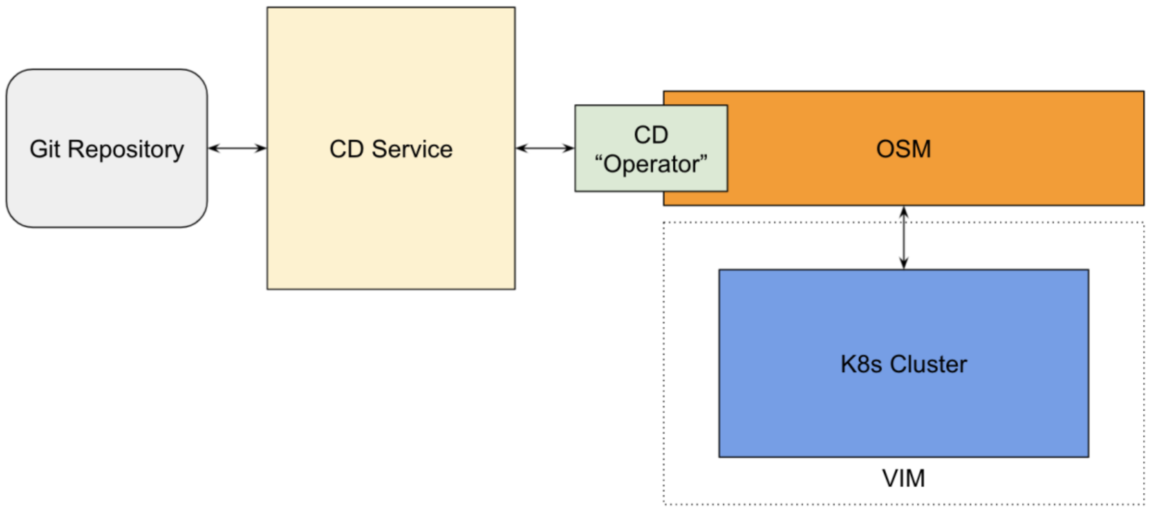


Figure 26: Synchronizing infrastructure descriptors in a git repository with deployment on K8s via a Continuous Deployment Service (and potential "operator" alongside OSM)

With OSM the plan is to setup ArgoCD to work alongside OSM to deploy KNFs based on descriptors stored in a git repository. Unlike with KNF placement this should not require extending OSM directly, as both the OSM API and ArgoCD provide the necessary functionality. However, this might require writing both a "watcher" and "operator" for ArgoCD, in order to parse the descriptors and deploy the KNFs through OSM respectively.

3.1.2.5 Plan for the Integration

With ArgoCD we have been able to deploy helm charts on Kubernetes in a few tests. The next step is to perform the same task with KNF descriptors and OSM as described above. Whilst parsing the descriptors and keeping watch of them from ArgoCD is straight forward, the technical challenge comes with keeping the KNFs in sync with OSM, particularly when an already deployed KNF is changed from the code in the repository.

Some experiments were performed, but a concrete design for the implementation is yet to be finalized. The goal is to at the very least create a standalone demonstration for this feature alongside the KNF placement enhancement.

3.2 Slice manager

3.2.1 Description and Motivation

The requirements gathering phase performed in Affordable5G, and detailed in D2.1, showcased the need to have the capacity to deploy end-to-end slices able to isolate the resources used and requested from the radio to the core network. This same need was reflected in the description of the Pilot #2 “Smart City Edge and Lamp post IoT deployment”, where there exists the need to have various network slices isolating the several services deployed and demonstrated in the Pilot.

In particular, the Castelloli testbed will incorporate the i2CAT solution as slice manager engine to fulfill this requirement. The i2CAT Slice Manager is, together with the non-RT RIC, one of the outputs of the H2020 5GCity project [28]. This slicing engine enables the programmatic partition of infrastructure and network resources into several slices while specifying the configuration and isolation level needed attending to the requirements specified by the Mobile Network Operators (MNOs). In particular, this software component enables the following main capabilities:

- It provides the capacity to handle the lifecycle management of network slices in a dynamic manner in terms of easiness and frequency on which slice (re-)configuration procedures can be performed.
- It enables a transparent and dynamic service provisioning over the resources available for the specific slice.
- It supports the interaction with different technologies for the management of infrastructure resources such as OpenStack, and it is able to interact with diverse MANO frameworks such as Open-Source MANO.
- It handles the O-RAN aligned access network by means of the non-real-time RIC employed in Affordable5G. This already mature integration between the i2CAT Slice Manager and the non-RT RIC makes it an excellent candidate for this project, since it allows the project to build and capitalized on integrations that have been already tested and validated in several H2020 projects and use cases besides the initial 5GCity, such as 5G-CLARITY [29] and 5GVICTORI [30], to name a few.

In Affordable5G, the Slice Manager module has been redesigned to meet the requirements of the architecture and is located at the SMO layer interconnected to the orchestrator and the non-RT RIC. In particular, in this architecture, the role of the Slice Manager is limited to the management of radio resources during the process of the slice creation, while the orchestrator takes care of accepting new slice requests in the form of templates made available by the network operator, process them and offload to the slice manager the tasks related to the radio side. Notice that the orchestrator is here the entity responsible for performing the hard slicing process, guarantee the reservation of infrastructure resources and the deployment of the required services over the slices.

As it can be observed in Figure 27, upon receiving the request from the orchestrator to create a new network slice, the i2CAT Slice Manager is responsible for managing and assigning the radio resources to the slice requested, as well as interacting with the non-RT RIC to reserve the radio resources and add the PLMN ID corresponding to such a slice. Notice that the same procedure applies for the case of 4G and 5G networks. The difference resides in that for 4G, a new PLMN is registered for the new slice following the Multi-Operator Core Networks (MOCN) or Multi Operator Radio Access Network (MORAN) model, while in the 5G case a new S-NSSAI is added in the aforementioned slice. At the moment of writing the 5G integration is being incorporated with the rest of the architecture, and the figures below make reference to the current 4G version following a MOCN approach. However, the interfaces and descriptions given can be easily extended to the native 5G version by specifying, together with the PLMN, the S-NSSAI of the new slice. These details will be included in D4.2. In any of the cases (either if the network is deployed under a 4G or a 5G model), the non-RT RIC then is in charge for configuring the CU-UPs accordingly using the NETCONF protocol (via the O-RAN O1 interface). Further details on this interface are provided in Section 3.2.

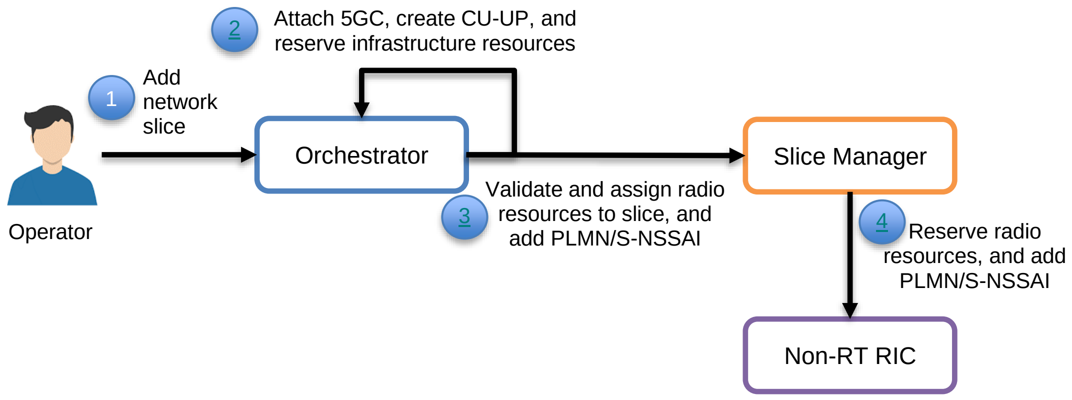

2
Attach 5GC, create CU-UP, and reserve infrastructure resources
1
Add network slice
Operator
Orchestrator
Slice Manager
3
Validate and assign radio resources to slice, and add PLMN/S-NSSAI
4
Reserve radio resources, and add PLMN/S-NSSAI
Non-RT RIC

Figure 27: Architectural role of the Slice Manager in Castelloli

3.2.2 Interfaces

The i2CAT Slice Manager exposes a REST API to implement both its northbound and southbound interfaces in order to interact with external components in the architecture and that allows the access and provision of radio resources. These interfaces in the Affordable5G architecture can be defined as follows:

- Northbound interface (NBi). This interface is considered to be the one from which the Slice Manager receives new incoming requests for reserving and configure radio resources for a certain slice (in addition to validate its availability). More specifically, in Affordable5G this interface is used to interconnect the Slice Manager with the orchestrator. For this purpose, this interface implements various REST calls, based on the following methods:
    - GET RAN Infrastructure. This method provides the information of all the instances of the non-RT RIC that are registered under the control of the Slice Manager, providing information such as URL through which they are reachable, their location and status, etc.
    - GET RAN Topology. This method allows retrieving the RAN resources that are available (i.e., that are not reserved -fully or partially- by other slices according to the PLMN or S-NSSAI model followed) under the control of a specific non-RT RIC instance and that are properly configured and ready to be reserved. In essence, this method allows the Slice Manager to provide the orchestrator, prior to the slice creation, with the resources at the radio level that can be selected for such a slice. It is worthy highlighting that this method is merely informative in order to allow the orchestrator to make an informed matching between the

radio resources requested in the slice template received and the resources actually available at the infrastructure at that moment of time. The information retrieved by this call includes the name of the small cells, their location, their vendor's information and possibly configuration, e.g., reference signal power.

- POST AFFOR5G Slice. This method allows the orchestrator to request the reservation of a set of radio resources for the creation of a new network slice, as well as the configuration via de non-RT RIC of the corresponding PLMN/S-NSSAIs. By means of this method the orchestrator also provides the IP address and the port through which the core network functions can be reached, and it is able to specify the Virtual Local Area Network (VLAN) used for the definition of the end-to-end slice. The return value contains the information of the reserved radio resources or an error message in case that the operation fails.
- DELETE AFFOR5G Slice. This method allows the termination of the slice as well as deactivating the previously reserved radio resources in order to complete the whole lifecycle of the slices. The only return value of this method consists of a confirmation or an error message.

- Southbound interface (SBi). This interface is used for the communication with the instances of the non-RT RIC registered and managed by the Slice Manager. The message interchanged here regard the validation, reservation, and configuration of a certain amount of radio resources under a specific slice. The REST calls implemented for such purposes can be defined as follows:
  - POST RAN Resources Validation. This method makes it possible to validate the integrity of the radio resources requested. In order words, it verifies if the radio resources are controlled by the non-RT RIC selected, and if they are available/ready to be assigned under a certain slice. The only return value of this method consists of a confirmation or an error message.
  - POST RAN Slice Chunk Creation. If the previous verification is correct, this method allows effectively reserving and registering the specified RAN resources under a certain slice. The only return value of this method consists of a confirmation or an error message.
  - POST RAN Slice Chunk Activation. Once the RAN resources have been properly reserved, this method is responsible for requesting the activation of the radio components associated to such slices and for providing the network configuration given by the orchestrator about the core network. As a result, this call returns the information of the radio resources reserved, or an error message.

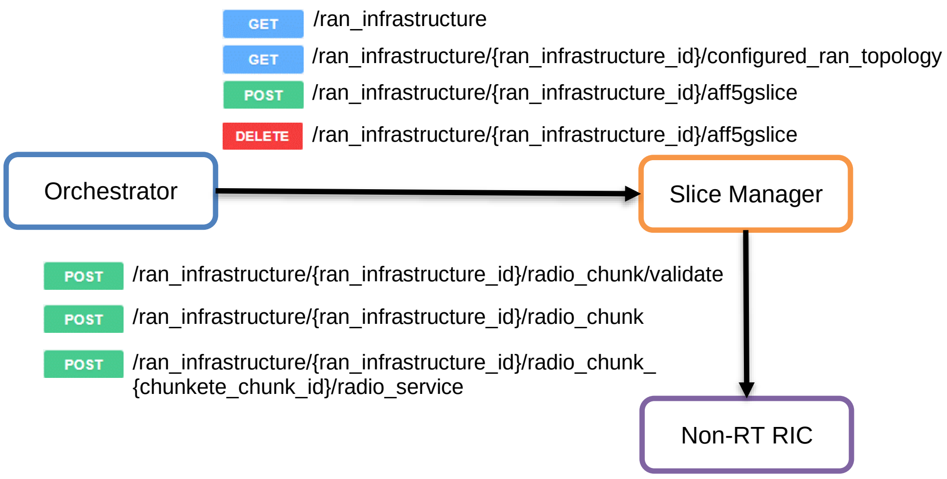

GET /ran_infrastructure
GET /ran_infrastructure/{ran_infrastructure_id}/configured_ran_topology
POST /ran_infrastructure/{ran_infrastructure_id}/aff5gslice
DELETE /ran_infrastructure/{ran_infrastructure_id}/aff5gslice
Orchestrator
Slice Manager
POST /ran_infrastructure/{ran_infrastructure_id}/radio_chunk/validate
POST /ran_infrastructure/{ran_infrastructure_id}/radio_chunk
POST /ran_infrastructure/{ran_infrastructure_id}/radio_chunk_{chunkete_chunk_id}/radio_service
Non-RT RIC

Figure 28: Interfaces and relationships of the i2CAT Slice Manager with other Affordable5G architectural components

The slicing model performed by the i2CAT Slice Manager is based on the 3GPP approach and the Next Generation Mobile Networks Alliance slicing concept, including the part related to the radio resources. The Intellectual Property Rights of this software framework and its REST API belong to I2CAT. It should be highlighted that for integration purposes the calls of the REST API required to facilitate the architecture integration of Affordable5G have been facilitated for the consortium' partners involved in such a task. The interfaces described and their relationship with the rest of the Affordable5G architecture can be seen in Figure 28. Notice that these interfaces are not specified according to any standard and that have been defined based on the requirements imposed by the project.

3.2.3 Requirements' Mapping

With regard to the requirements on pilots and use cases identified in D1.1, the i2CAT Slice Manager satisfy or contributes to satisfying the following ones:

No.	ID	Requirement	Note
1	REQ-NET-12	The system shall support virtualised RAN functions.	Ongoing
2	REQ-MAN-01	The architecture shall provide 5G network slicing to support services with diverse QoS requirements.	Planned
3	REQ-MAN-04	The system shall support several slices over the same infrastructure.	Done
4	REQ-MAN-05	The system shall support management of slices in an end-to-end fashion.	Done
5	REQ-MAN-20	The slice shall respond to application service requirements.	Ongoing

No.	ID	Requirement	Note
6	REQ-MAN-26	The system shall ensure the isolation of the slices.	Done
7	REQ-MAN-32	The solution should allow the deployment of end-to-end slices able to accommodate the requirements imposed.	Ongoing

Table 8: Requirements' mapping of the Slice Manager

Notice that the requirements stated above refer to both the whole Affordable5G architecture and the specific Pilot test (and demonstration) where the i2CAT Slice Manager is involved.

3.2.4 Technical Challenges

During the course of the project the design of the i2CAT Slice Manager has been rearchitected in order to fulfill the necessary functionalities of the Affordable5G project. More specifically, the enhancements and modifications performed are as follows:

- Extended deployment options. The Slice Manager was adapted from its native design (handling directly end-to-end slice request and reserving infrastructure and radio resources) to the Affordable5G architecture, in which the orchestrator is the entity receiving such requests, managing the infrastructure, and offloading to the Slice Manager the RAN resource management. This process has required to redesign the internal structure of this software component as well as its data structure. These modifications, however, allow the Slice Manager to include a different deployment option additional to the one that was implemented, i.e., the role of the orchestrator was initially limited to receive and handle service deployment request on an already deployed slice by the Slice Manager. As a consequence, this component expands also its deployments options.
- Integration changes with non-RT RIC. As a result of the modifications included in the deployments options, the integration between the non-RT RIC and the Slice Manager has also suffered changes. More specifically, the change in its data model and the fact of having the slice services deployed by the orchestrator, e.g., the virtualized core network, have introduced changes in the interaction of these two components. In this case, the Slice Manager must retrieve the details of the core network and the slice itself and interchange them with the non-RT RIC once the resource request has been validated and properly reserved. Although these two software elements had been tested in previous H2020 projects, the addition of these new functionalities has also involved a more rigorous testing and validation process between them.
- REST API. Aligned with the changes mentioned above, the REST API of the Slice Manager has been modified in order to include the new operations in the northbound and the southbound interfaces required in Affordable5G.

3.2.5 Plan for the Integration

One of the main technical challenges and ongoing enhancements in the i2CAT Slice Manager is the evolution from 4G to 5G in its integration in the Affordable5G architecture. More specifically, these enhancements aim to allow the Slice Manager the possibility to work under a native 5G slicing model by requesting the addition of S-NSSAIs for any of the slices deployed under the umbrella of a certain PLMN. Given the implications not only on the Slice Manager itself, but also on the dependency on the modifications required on the orchestrator, the non-RT RIC, the CU-UP and the DU, at the moment of writing this document this enhancement is a work in progress and will be further documented in D3.2.

Furthermore, according to the 3GPP standard, the core network supports network slicing by assigning the required resources to the NFs of each slice identified by an S-NSSAI. Then, the NSSF (cf. Section 2.2.9) is responsible for selecting the S-NSSAI corresponding to the network slice that serves a certain UE. The procedure by which the Athonet's Core Network and the Slice Manager can be integrated for the assignment of S-NSSAIs is still to be discussed within the consortium.

3.3 Non-RT RIC

3.3.1 Description and Motivation

3.3.1.1 Non-RT RIC Features

The non-RT RIC is an orchestration and automation component described by the O-RAN Alliance for non-real-time intelligent management of RAN functions. Its main goal is to support non-real-time radio resource management, higher layer procedure optimization, policy optimization in RAN, and providing guidance, parameters, policies, and AI/ML models to support the operation of near-RT RIC functions in the RAN to achieve higher-level non-real-time objectives.

The non-RT RIC will communicate with near-RT RIC elements in the RAN via the A1 interface. Using the A1 interface, the non-RT RIC will facilitate the provision of policies for individual UEs or groups of UEs; monitor and provide basic feedback on policy state from near-RT RICs; provide enrichment information as required by near-RT RICs; and facilitate ML model training, distribution, and inference in cooperation with the near-RT RICs.

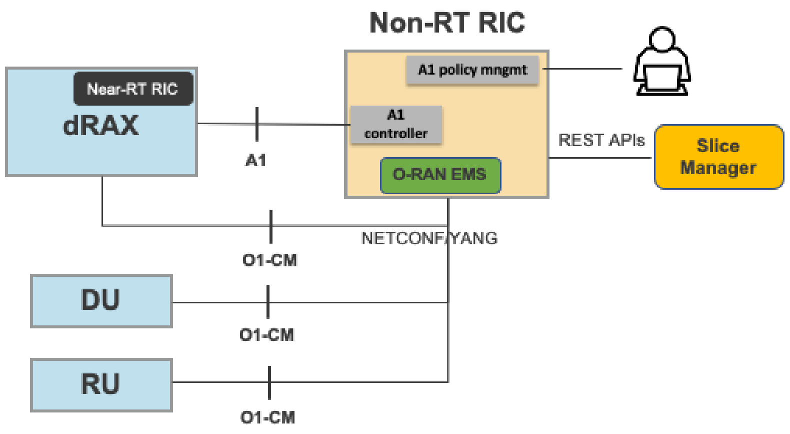


Figure 29: Non-RT RIC internal architecture and interfaces.

As shown in Figure 29, our implementation of the non-RT RIC consists of the A1 interface, including the controller and the policy management, and the O-RAN Element Management System (EMS), which handles the configuration of the O-RAN elements through the O1-CM interface. In the following we describe both components in more details.

3.3.1.2 O-RAN EMS Features

The O-RAN EMS is the software component, sitting in the management, orchestration and automation layer, which performs configuration management (CM) operations through the NETCONF/YANG protocol. Specifically, the EMS exposes the O1-CM interface [31], which allows managing O-RAN elements, such as the near-RT RIC, CUs, DU and RU, provided that a NETCONF server and YANG models defined in O-RAN are available.

In the Affordable5G system architecture, the EMS is also connected to the Slice Manager through REST APIs and processes its requests, including slice instantiation and deployment. As shown in the figure, the EMS currently supports ACC dRAX configuration changes, while support to DU and RU can be potentially extended depending on the development plans of the responsible partner.

The EMS block has been integrated into RACOON ("RAN Controller based on OpenFlow, OvsDB and Netconf"), a management plane service originally developed to configure Wi-Fi physical interfaces and to manage the lifecycle of the virtual APs instantiated over the physical interfaces within the framework of the 5G-PICTURE project (5G-PICTURE D5.4). The NETCONF client running on the EMS is a proprietary SW developed in Python and is part of the I2CAT's technology transfer and intellectual property strategy.

To date, I2CAT is enhancing the EMS by introducing the support of the latest release of the dRAX stack, which includes the 5G protocol stack and a number of new features. In parallel, the integration between the slice manager and the EMS is in progress at I2CAT's premises.

Once the necessary lab integration and validation phase will be completed, the EMS capabilities will be tested in the Castelloli test site. The first stage of testing will be conducted by connecting an EMS instance running in the I2CAT lab with a dRAX instance running in Castelloli through a VPN connection supplied by CELL. In the second stage, the EMS software stack will be deployed in a dedicated VM in Castelloli and will be demonstrated during the Affordable5G project review. This phase is key to identifying potential bugs and interoperability issues and requires technical staff on site to run end-to-end tests with commercial terminals.

Next year, I2CAT plans to extend the EMS support to include other O-RAN elements. REL expressed interest in adding a NETCONF server to the RU, while EURE did not confirm a plan to integrate the O1-CM interface in the DU under development.

3.3.2 Interfaces

As mentioned above, the primal role of the non-RT RIC is to apply AI/ML based algorithms through policies to provide innovative RAN use cases. The policies transferred from non-RT RIC to near-RT RIC are created, modified, and deleted by the non-RT RIC, guided by context information or policy state information received over A1 or O1, e.g., from the RAN indicating intent fulfillment (O1).

The A1 interface is specified by the O-RAN Alliance WG2 [32], in two directions:

- A1 interface General Aspects and Principles v2.0 – it defines A1 policy aspects and the concepts for A1 Enrichment Information.
- A1 interface: Application Protocol v2.0 – it defines the rules for encoding data types on the A1 interface.

Furthermore, the non-RT RIC project is currently under development within O-RAN Software Community [33]. Specifically, the A1 Policy management service and A1 controller are already available.

3.3.3 Requirement's Mapping

In terms of the requirements on pilots and use cases identified in D1.1, the non-RT RIC satisfies or contributes to satisfying the following ones:

No.	ID	Requirement	Note
1	REQ-NET-26	O-RAN Operations and Maintenance Interface (O1) shall be protected.	Done
2	REQ-NET-27	O-RAN Management A1 interface shall be protected.	Planned
3	REQ-MAN-03	The system shall provide a way to create, instantiate, update and delete Network Slices.	Ongoing – applies to RAN part of slice
4	REQ-MAN-32	The solution should allow the deployment of end-to-end slices able to accommodate the requirements imposed.	Ongoing - applies to RAN part of slice

Table 9: Requirements' mapping of the non-RT RIC.

3.3.4 Technical Challenges

The following challenges are identified in the development of the non-RT RIC:

- O-RAN EMS:
  - Modelling in RACOON new O-RAN functional elements, namely CU-CP, CU-UP, DU and RU from different vendors. This requires consuming the respective YANG models and generating corresponding REST endpoints.
  - Adding to ORAN EMS the required business logic to be able to appropriately configure the new elements and orchestrating them when deploying a network slice.
- A1 Policy management:
  - Identifying and extracting from the O-RAN software project the kubernetes services dealing with policy management (discussed in the next section)
  - Integrating the A1 policy components within the Affordable Non-RT RIC software project.
  - Integrate the A1 policy components with the A1 mediator available in the Affordable near-Rt RIC.
  - Defining mechanisms in the non-RT RIC to provide rApps with a consistent view of the O-RAN infrastructure, being able to link xApp policies from the near-RT RIC with the physical network functions that are affected by these policies and are visible through the O-RAN EMS.
  - Define, develop and demonstrate reference xApps and policy types that can be controlled from the non-RT RIC.

3.3.5 Plan for the Integration

The integration plans for I2CAT related to the non-RT RIC are divided in plans for the O-RAN EMS and plans for the A1 policy management system.

Regarding the O-RAN EMS the development and integration is planned in three steps:

- Step 1: Integrate RACOON with dRAX 2.0, which is an earlier version of a near-Rt RIC that can be used to control E1000 Accelleran 4G Small Cells.
- Step 2: Integrate RACOON with the Accelleran's dRAX 5G software that represents a near-RT RIC. In a first step proprietary DU and RU components provided by Accelleran will be used to validate the integration with dRAX 5G.
- Step 3: Replace DU and RU components in Step 2 by those provided by the corresponding partners in Affordable 5G.

Regarding the A1 policy management component, the development and integration will consist of the following steps:

- Step 1: Deploy and validate the ORAN A1 workflow made available in the ORAN Bronze release [34]. This step has already been completed and evidence is provided below in this section.
- Step 2: Deploy ORAN D-release and extract the policy management related components, which have been identified to be the "A1 Policy Management Service" and the "A1 Adapter" depicted in Figure 30
- Step 3: Package the ORAN A1 related microservice extracted in Step 2 and package them with the Affordable Non-RT RIC. Develop an additional micro-service that provides a unifying framework between the xApp policies made available by the policy management service and the configuration of physical network functions enabled by the ORAN EMS component.
- Step 4: Demonstrate the integration between the A1 policy management service in the Non-Rt RIC and the A1 interface available in the near-RT RIC.

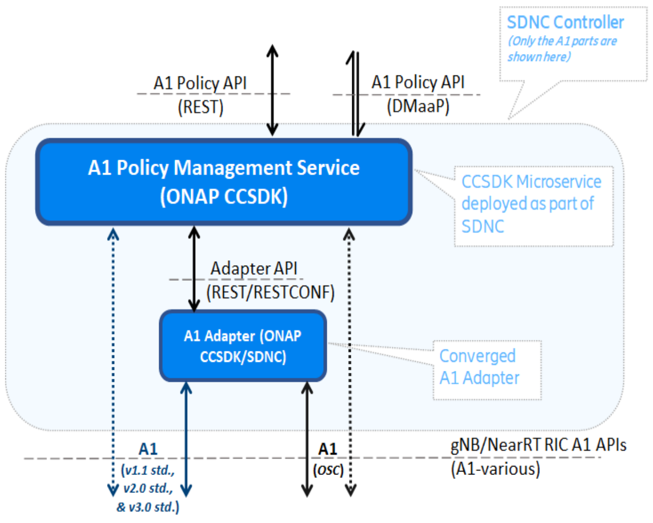

SDNC Controller
(Only the A1 parts are shown here)
A1 Policy API
(REST)
A1 Policy API
(DMaaP)
A1 Policy Management Service
(ONAP CCSDK)
CCSDK Microservice deployed as part of SDNC
Adapter API
(REST/RESTCONF)
A1 Adapter (ONAP CCSDK/SDNC)
Converged A1 Adapter
A1
(v1.1 std., v2.0 std., & v3.0 std.)
A1
(OSC)
gNB/NearRT RIC A1 APIs
(A1-various)

Figure 30: ORAN SMO A1 related micro-services

Next, we provide some evidence related to Step 1 for the A1 policy management component. Figure 31 shows the SMO components available in the O-RAN Bronze release. I2CAT has deployed an instance of the ORAN Bronze SMO RIC Virtual Machines (VMs). Each VM features a kubernetes cluster that is used to deploy all components of the SMO and RIC applications.

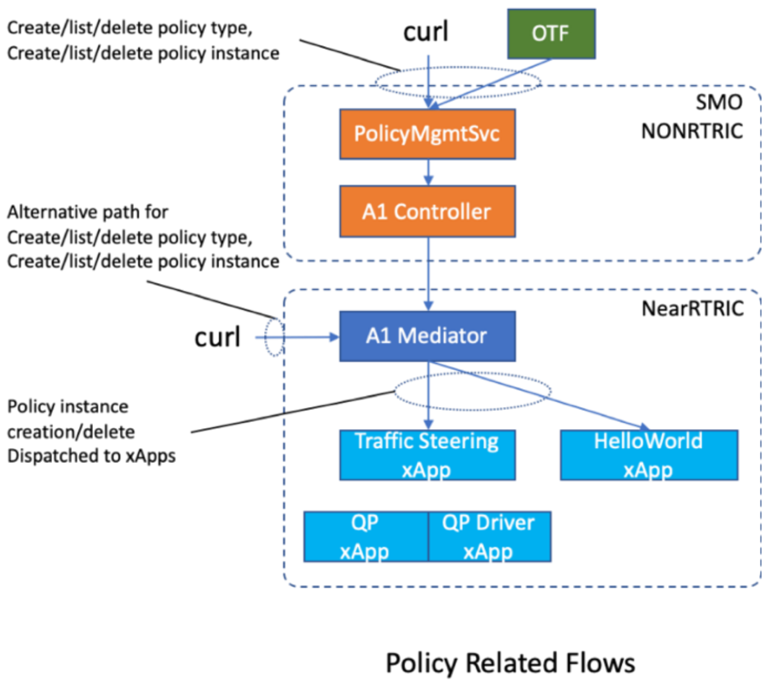
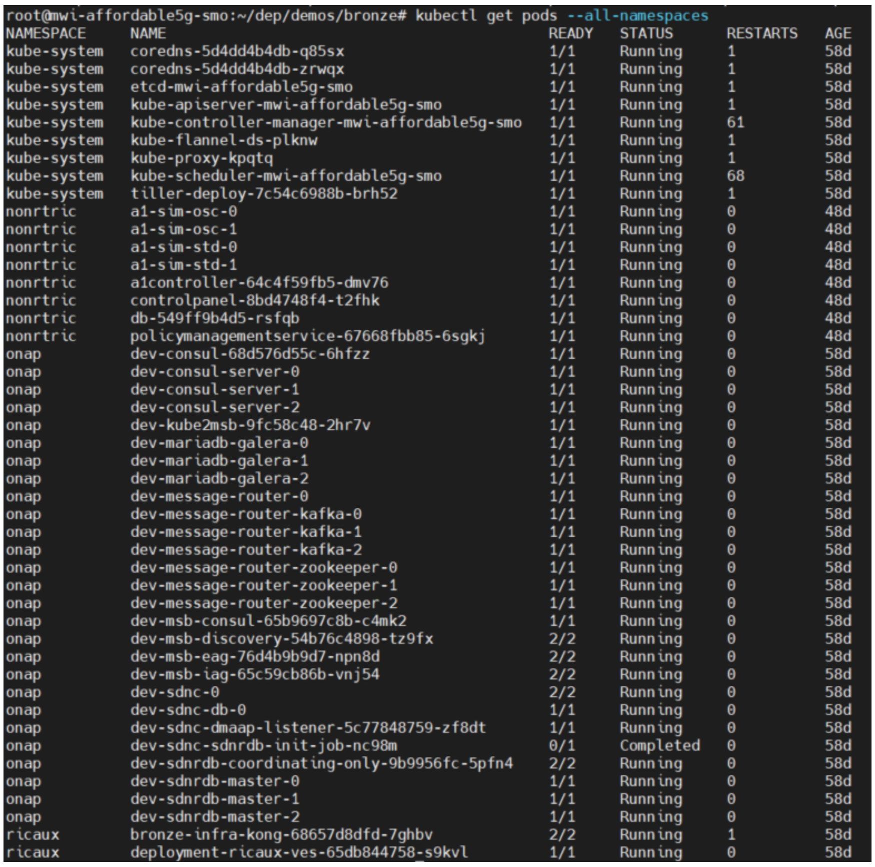


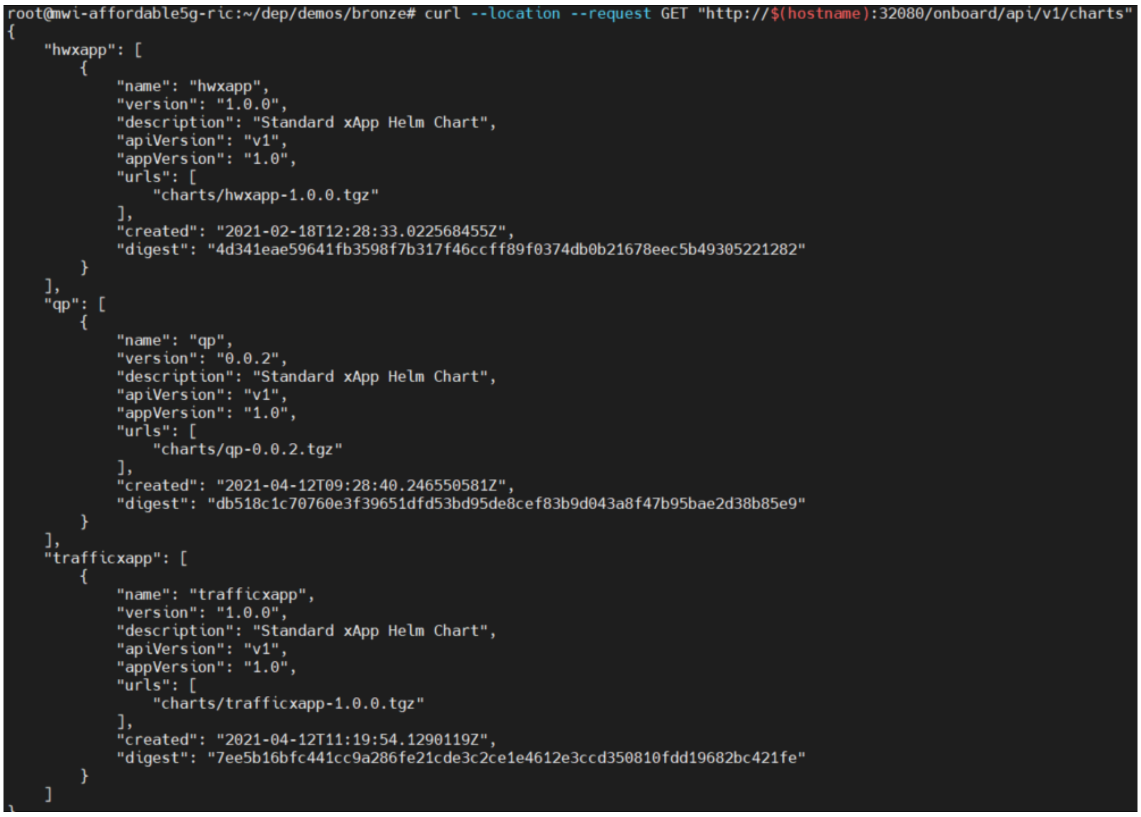
Figure 31: Internal components of the SMO project

Figure 32 lists the Kubernetes pods running. In particular, we can notice the ones related to the nonrtric namespace and the pod corresponding to the a1controller. Finally, Figure 33 illustrates the list of onboarded xApps, specifically “hwxapp” (the Hello World xApp), “qp” (the Quality Prediction xApp) and “trafficxapp” (Traffic Steering xApp). Using this deployment, we validated the discovery of policy types in the near-RT RIC, and the ability to instantiate policy instances from the SMO.



```
root@mwi-affordable5g-smo:~/dep/demos/bronze# kubectl get pods --all-namespaces
NAMESPACE     NAME                                         READY   STATUS      RESTARTS   AGE
kube-system   coredns-5d4dd4b4db-q85sx                     1/1     Running     1          58d
kube-system   coredns-5d4dd4b4db-zrwqx                     1/1     Running     1          58d
kube-system   etcd-mwi-affordable5g-smo                    1/1     Running     1          58d
kube-system   kube-apiserver-mwi-affordable5g-smo          1/1     Running     1          58d
kube-system   kube-controller-manager-mwi-affordable5g-smo 1/1     Running     61         58d
kube-system   kube-flannel-ds-plknw                        1/1     Running     1          58d
kube-system   kube-proxy-kpqtq                             1/1     Running     1          58d
kube-system   kube-scheduler-mwi-affordable5g-smo          1/1     Running     68         58d
kube-system   tiller-deploy-7c54c6988b-brh52               1/1     Running     1          58d
nonrtric      a1-sim-osc-0                                 1/1     Running     0          48d
nonrtric      a1-sim-osc-1                                 1/1     Running     0          48d
nonrtric      a1-sim-std-0                                 1/1     Running     0          48d
nonrtric      a1-sim-std-1                                 1/1     Running     0          48d
nonrtric      a1controller-64c4f59fb5-dmv76                1/1     Running     0          48d
nonrtric      controlpanel-8bd4748f4-t2fhk                 1/1     Running     0          48d
nonrtric      db-549ff9b4d5-rsfqb                          1/1     Running     0          48d
nonrtric      policymanagementservice-67668fbb85-6sgkj     1/1     Running     0          48d
onap          dev-consul-68d576d55c-6hfzz                  1/1     Running     0          58d
onap          dev-consul-server-0                          1/1     Running     0          58d
onap          dev-consul-server-1                          1/1     Running     0          58d
onap          dev-consul-server-2                          1/1     Running     0          58d
onap          dev-kube2msb-9fc58c48-2hr7v                  1/1     Running     0          58d
onap          dev-mariadb-galera-0                         1/1     Running     0          58d
onap          dev-mariadb-galera-1                         1/1     Running     0          58d
onap          dev-mariadb-galera-2                         1/1     Running     0          58d
onap          dev-message-router-0                         1/1     Running     0          58d
onap          dev-message-router-kafka-0                   1/1     Running     0          58d
onap          dev-message-router-kafka-1                   1/1     Running     0          58d
onap          dev-message-router-kafka-2                   1/1     Running     0          58d
onap          dev-message-router-zookeeper-0               1/1     Running     0          58d
onap          dev-message-router-zookeeper-1               1/1     Running     0          58d
onap          dev-message-router-zookeeper-2               1/1     Running     0          58d
onap          dev-msb-consul-65b9697c8b-c4mk2              1/1     Running     0          58d
onap          dev-msb-discovery-54b76c4898-tz9fx           2/2     Running     0          58d
onap          dev-msb-eag-76d4b9b9d7-npn8d                 2/2     Running     0          58d
onap          dev-msb-iag-65c59cb86b-vnj54                 2/2     Running     0          58d
onap          dev-sdnc-0                                   2/2     Running     0          58d
onap          dev-sdnc-db-0                                1/1     Running     0          58d
onap          dev-sdnc-dmaap-listener-5c77848759-zf8dt     1/1     Running     0          58d
onap          dev-sdnc-sdnrdb-init-job-nc98m               0/1     Completed   0          58d
onap          dev-sdnrdb-coordinating-only-9b9956fc-5pfn4  2/2     Running     0          58d
onap          dev-sdnrdb-master-0                          1/1     Running     0          58d
onap          dev-sdnrdb-master-1                          1/1     Running     0          58d
onap          dev-sdnrdb-master-2                          1/1     Running     0          58d
ricaux        bronze-infra-kong-68657d8dfd-7ghbv           2/2     Running     1          58d
ricaux        deployment-ricaux-ves-65db844758-s9kvl       1/1     Running     0          58d
```

Figure 32: List of the Kubernetes pods available



```
root@mwi-affordable5g-ric:~/dep/demos/bronze# curl --location --request GET "http://$(hostname):32080/onboard/api/v1/charts"
{
    "hwxapp": [
        {
            "name": "hwxapp",
            "version": "1.0.0",
            "description": "Standard xApp Helm Chart",
            "apiVersion": "v1",
            "appVersion": "1.0",
            "urls": [
                "charts/hwxapp-1.0.0.tgz"
            ],
            "created": "2021-02-18T12:28:33.022568455Z",
            "digest": "4d341eae59641fb3598f7b317f46ccff89f0374db0b21678eec5b49305221282"
        }
    ],
    "qp": [
        {
            "name": "qp",
            "version": "0.0.2",
            "description": "Standard xApp Helm Chart",
            "apiVersion": "v1",
            "appVersion": "1.0",
            "urls": [
                "charts/qp-0.0.2.tgz"
            ],
            "created": "2021-04-12T09:28:40.246550581Z",
            "digest": "db518c1c70760e3f39651dfd53bd95de8cef83b9d043a8f47b95bae2d38b85e9"
        }
    ],
    "trafficxapp": [
        {
            "name": "trafficxapp",
            "version": "1.0.0",
            "description": "Standard xApp Helm Chart",
            "apiVersion": "v1",
            "appVersion": "1.0",
            "urls": [
                "charts/trafficxapp-1.0.0.tgz"
            ],
            "created": "2021-04-12T11:19:54.1290119Z",
            "digest": "7ee5b16bfc441cc9a286fe21cde3c2ce1e4612e3ccd350810fdd19682bc421fe"
        }
    ]
}
```

Figure 33: List of currently deployed xApps

3.4 Telemetry data collector

3.4.1 Description and Motivation

As described in D1.2, the Affordable5G approach for network telemetry is integrated with the support for data analytics provided by the Network Data Analytics Function (NWDAF) and the C-Management Data Analytics Function (MDAF) components. The network telemetry must be able to create knowledge based on monitoring data collected from all the different sub-systems comprising a 5G system, including Radio Access Network, Core Network, Transport Network and the NFV Infrastructure. The architecture of the telemetry and data analytics framework is depicted in Figure 34.

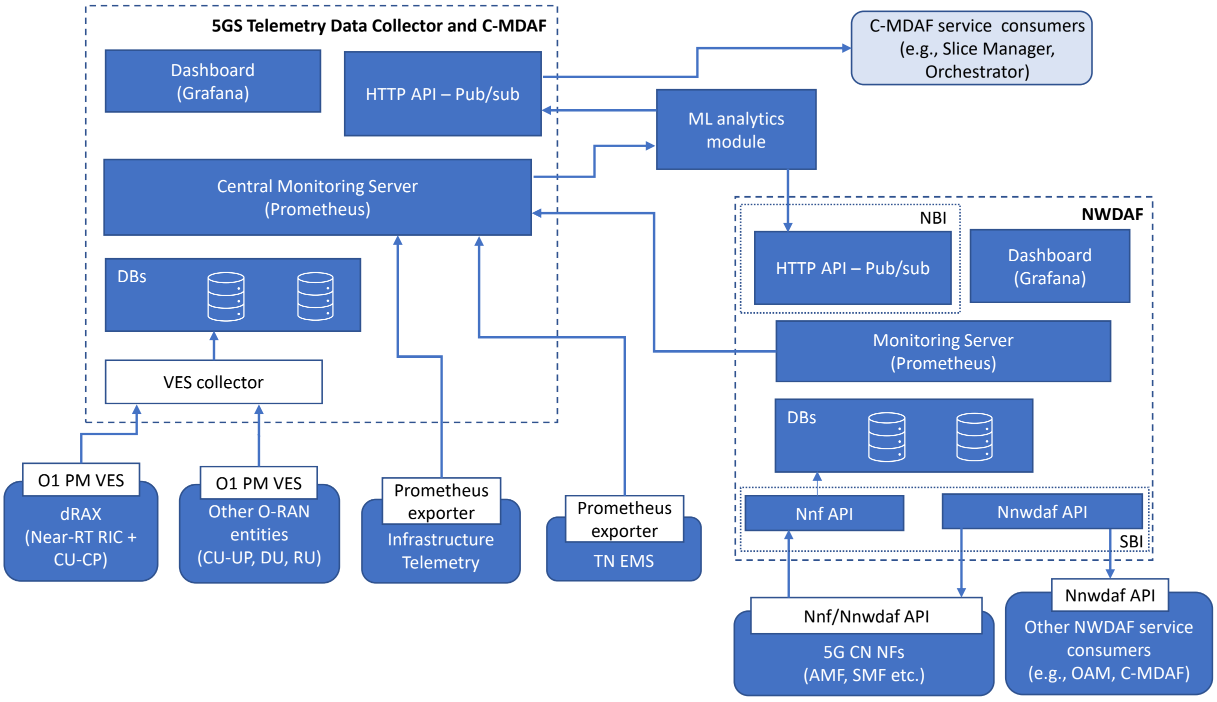

5GS Telemetry Data Collector and C-MDAF
Dashboard (Grafana)
HTTP API – Pub/sub
Central Monitoring Server (Prometheus)
DBs
VES collector
C-MDAF service consumers (e.g., Slice Manager, Orchestrator)
ML analytics module
NWDAF
NBI
HTTP API – Pub/sub
Dashboard (Grafana)
Monitoring Server (Prometheus)
DBs
Nnf API
Nnwdaf API
SBI
O1 PM VES
dRAX (Near-RT RIC + CU-CP)
O1 PM VES
Other O-RAN entities (CU-UP, DU, RU)
Prometheus exporter
Infrastructure Telemetry
Prometheus exporter
TN EMS
Nnf/Nnwdaf API
5G CN NFs (AMF, SMF etc.)
Nnwdaf API
Other NWDAF service consumers (e.g., OAM, C-MDAF)

Figure 34: Network telemetry and data analytics framework

The NWDAF is a well-specified component of the 5G Core Network; on the contrary, although C-MDAF is conceptually described in current 3GPP specifications, the relevant technical details are still under investigation. To that end, the general architectural principles regarding the realization of network telemetry in Affordable5G are demonstrated including the C-MDAF, while the implementation is initially driven by the need to support the NWDAF functionalities.

3.4.1.1 NWDAF

The NWDAF can either provide network analysis data to other NFs (i.e., analytics information) or NFs can request subscription from NWDAF for data delivery (i.e., events subscription) by using Nnwdaf interface (Figure 35).

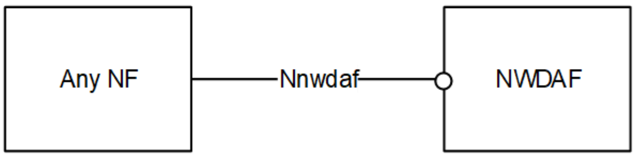


Figure 35: Network Data Analytics Exposure architecture

The 3GPP has defined two models of analytics services provided from NWDAF. The first one involves a subscription model and is used in case periodic analytics information is needed, e.g., periodic load information (current/predicted) of network slice. The second involves a Request/Ad-Hoc model and is used if a one-time analytics computation and information is needed, e.g., experience score of a newly deployed application for a particular day/hour/region.

The NWDAF offers two services (called Nnwdaf services in [35]). The first one is the Nnwdaf_EventsSubscription service, which enables the NF service consumers (like PCF, NSSF, OAM etc.) to subscribe to and unsubscribe from different analytics events provided by the NWDAF, and subsequently enables the NWDAF to notify the NF consumers about subscribed events. The second Nnwdaf service is the Nnwdaf_AnalyticsInfo service, which enables the NF consumers to request and get specific analytics from the NWDAF.

The Nnwdaf services and the associated operations are listed in Table 10.

Service	Service operations
Nnwdaf_EventsSubscription	Nnwdaf_EventsSubscription_Subscribe
	Nnwdaf_EventsSubscription_UnSubscribe
	Nnwdaf_EventsSubscription_Notify
Nnwdaf_AnalyticsInfo	Nnwdaf_AnalyticsInfo_Request

Table 10: Nnwdaf services and the associated operations

In addition, NWDAF fetches data from other NFs using the Nnf interface (Figure 36).

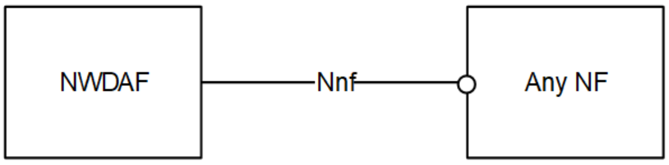


Figure 36: Data Collection architecture from any 5GC NF

The interactions between the NWDAF, the NRF, the AF and the NF consumer and provider are illustrated in Figure 37 and Figure 38.

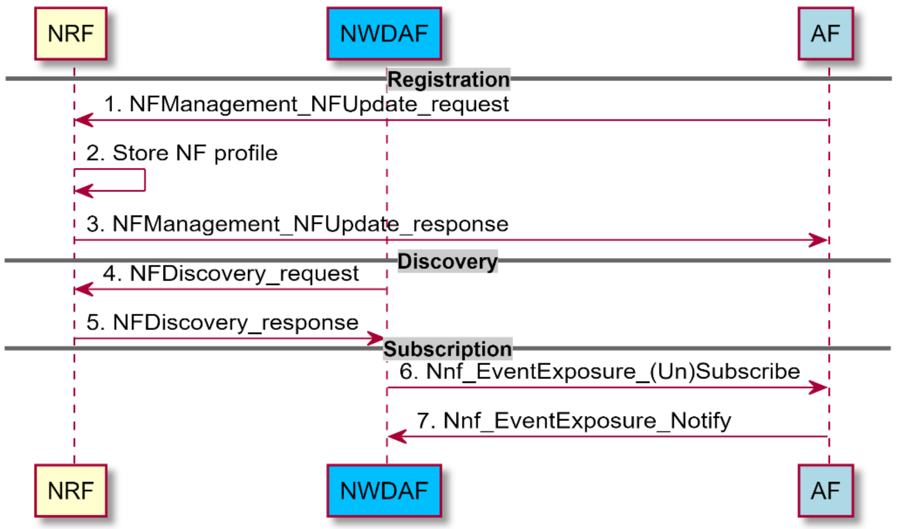

NRF
NWDAF
AF
Registration
1. NFManagement_NFUpdate_request
2. Store NF profile
3. NFManagement_NFUpdate_response
Discovery
4. NFDiscovery_request
5. NFDiscovery_response
Subscription
6. Nnf_EventExposure_(Un)Subscribe
7. Nnf_EventExposure_Notify
NRF
NWDAF
AF

Figure 37: Registration, discovery and subscription phase for the data collection from a trusted AF [36]

Figure 37 divides the procedure of AF procedure into three distinct phases, namely: (i) the registration phase, including the message exchange between AF and NRF, (ii) the discovery phase including the request/response interaction between NWDAF and NRF and (iii) the subscription phase, including the event exposure messages between NWDAF and AF.

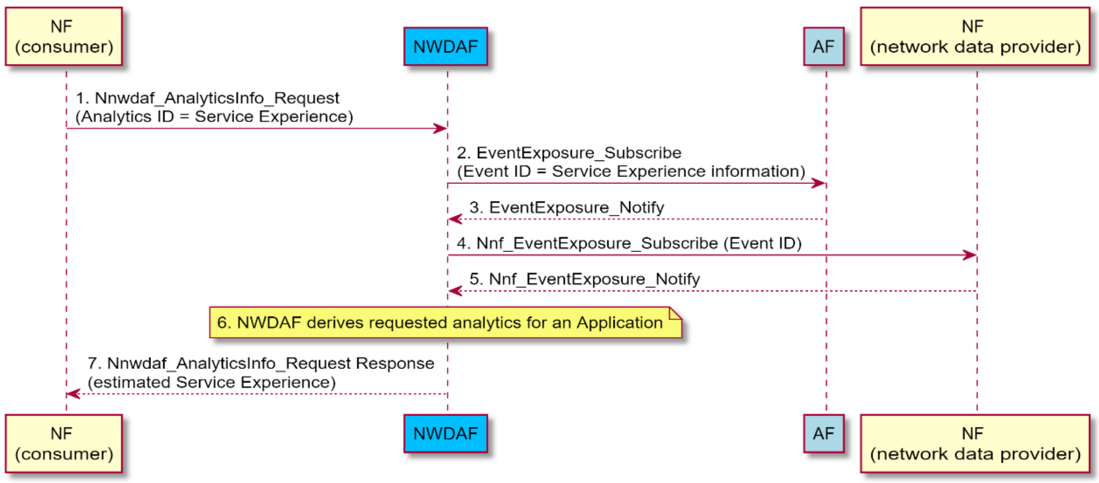

NF (consumer)
NWDAF
AF
NF (network data provider)
1. Nnwdaf_AnalyticsInfo_Request (Analytics ID = Service Experience)
2. EventExposure_Subscribe (Event ID = Service Experience information)
3. EventExposure_Notify
4. Nnf_EventExposure_Subscribe (Event ID)
5. Nnf_EventExposure_Notify
6. NWDAF derives requested analytics for an Application
7. Nnwdaf_AnalyticsInfo_Request Response (estimated Service Experience)
NF (consumer)
NWDAF
AF
NF (network data provider)

Figure 38: NF consumer/provider interaction through NWDAF to provide service experience for an application [36]

Figure 38 depicts the consumer/provider interaction for the provision of service experience to an application. Initially, the consumer sends a request to NWDAF, which interacts with the respective AF to notify the event exposure. Then, NWDAF invokes the NF provider through the appropriate messaging. Finally, the requested analytics for the application are published to the requesting NF consumer.

3.4.1.2 5GS Telemetry Data Collector and C-MDAF

The 5GS Telemetry Data Collector is integrated with the C-MDAF (left side of Figure 34), which is responsible for providing management data analytics to C-MDAF service consumers. The architecture of this component resembles the architecture of NWDAF; however, there are some notable differences that are discussed below.

First, the 5GS Telemetry Data Collector includes a monitoring server that is able to collect performance measurement data from the NWDAF, the virtualized infrastructure and the Transport Network elements by leveraging Prometheus instances organized in a hierarchical structure.

Furthermore, Performance Measurement (PM) data can also be collected from O-RAN NFs using the data streaming service of the O1 interface (Virtual Event Streaming - VES). In case of NFs included in the dRAX (see section 2.1.4) an adaption module is needed to bring performance measurement data received over the Kafka bus to a format compatible with O1.

The O1 Performance Data Streaming is part of the O1 Performance Assurance Management Services, which are described in [31]. The service allows a Performance Assurance Management Service (MnS) Provider to stream high volume asynchronous streaming performance measurements (at a configurable frequency) to a Performance Assurance MnS Consumer using a secure WebSocket connection. The Performance Data Streaming service is implemented according to the 3GPP specifications TS 28.550 [37] and TS 28.532 [38] (in 3GPP terminology, the service is called Streaming data reporting service).

The O1 Performance Data Streaming service operations include the following:

- establishStreamingConnection: the streaming connection is initiated through an HTTPS POST method followed by an HTTP GET (upgrade) method to establish the WebSocket connection.
- terminateStreamingConnection: it is provided through a WebSocket Close Frame to terminate the connection when all stream jobs have ended.
- reportStreamData: it enables the service producer to send (in a binary format encoded in ASN.1) a unit of streaming data to the service consumer.
- addStream: it allows the service producer to add one or more reporting streams to an already established connection.
- deleteStream: it is used to remove one or more reporting streams from an already established connection.
- getConnectionInfo (optional for O-RAN NFs): it allows the service provider to obtain information from the consumer about one or more streaming connections.
- getStreamInfo (optional for O-RAN NFs): it allows the service provider to obtain information from the consumer about one or more reporting streams.

All data collected at the 5GS Telemetry Data Collector are available for feeding ML models used for the provisioning of ML-based data analytics. Analytics services are provided by the C-MDAF to interested consumers (like the Slice Manager or the Orchestrator) over an HTTP REST or a Pub/Sub interface.

3.4.2 Interfaces

3.4.2.1 NWDAF

The interfaces listed below are categorized according to the specific component described in the subsection below.

Based on the use cases that can be covered by the NWDAF (see [39] and D1.2, section 3.3.5.2.5) and the three services it can offer (namely, periodic notification, event notification and one-time predictive information – see D1.2, sections 3.3.5.2.2, 3.3.5.2.3 and 3.3.5.2.4), the NWDAF component consists of the modules depicted on the right side of Figure 34.

The main architectural components can be described as follows.

Southbound Interface (SBI): It provides all the necessary interfaces to accommodate the data collection from different types of data sources. As mentioned above, the NWDAF collects and processes data from the 5G core NFs based on predefined interfaces (Nnf and Nnwdaf).

The Nnwdaf services (Table 10) are provided through RESTful APIs, and the resources and HTTP methods used per service operation are listed in Table 11.

Service operation	API URI	HTTP method	Initiator	Target
Nnwdaf_EventsSubscription_Subscribe	{apiRoot}/nnwdaf-eventssubscription/v1/subscriptions	POST	NF consumer	NWDAF
	{apiRoot}/nnwdaf-eventssubscription/v1/subscriptions/{subscriptionId}	PUT	NF consumer	NWDAF
Nnwdaf_EventsSubscription_UnSubscribe	{apiRoot}/nnwdaf-eventssubscription/v1/subscriptions/{subscriptionId}	DELETE	NF consumer	NWDAF
Nnwdaf_EventsSubscription_Notify	{notificationURI}	POST	NWDAF	NF consumer
Nnwdaf_AnalyticsInfo_Request	{apiRoot}/nnwdaf-analyticsinfo/v1/analytics	GET	NF consumer	NWDAF

Table 11: Resources and HTTP methods used for NWDAF service operations

The Nnwdaf_EventsSubscription_Subscribe service operation is used by an NF service consumer to:

- subscribe for event notifications from the NWDAF using the POST method and the "{apiRoot}/nnwdaf-eventssubscription/v1/subscriptions" URI. The URI where the requested notifications will be received (i.e., “notificationURI”) is provided in the body of the POST request. In response to this command, the NWDAF will assign an event subscriptionId.
- update subscription for event notifications from the NWDAF, using the PUT method and the "{apiRoot}/nnwdaf-eventssubscription/v1/subscriptions/{subscriptionId}" URI.

The Nnwdaf_EventsSubscription_Unsubscribe service operation is used by an NF service consumer to unsubscribe from event notifications. This operation is implemented using the DELETE method on the "{apiRoot}/nnwdaf-eventssubscription/v1/subscriptions/{subscriptionId}" URI.

The Nnwdaf_EventsSubscription_Notify service operation is used by an NWDAF to notify NF consumers about subscribed events. Notifications are sent through POST requests using the "{notificationURI}" resource URI.

The Nnwdaf_AnalyticsInfo_Request service operation is used by an NF service consumer to request and get specific analytics information from the NWDAF. This operation is implemented through GET requests on the "{apiRoot}/nnwdaf-analyticsinfo/v1/analytics" resource URI followed by appropriate query parameters.

Data storage: The NWDAF provides data analytics services based on a mechanism of event subscription. To support these services, NWDAF needs to store and manage data related to the operational status of module (e.g., active subscriptions, etc.), as well as data collected from each data source. Therefore, the data storage of NWDAF consists of two different databases:

- Document DB. A document DB (for example MongoDB) that can be used for the storage of the active subscriptions to 5GCore NFs. The database can also support the stateless design approach of the module.
- Timeseries DB. The timeseries DB (for example InfluxDB) is used for the storage of all the collected data from the connected data sources in order to support complex analytics information services (e.g., inference of ML models, etc.).

Monitoring server/Dashboard: The monitoring server (Prometheus) is one of the core components of the NWDAF module, being responsible for providing analytics services during real time, based on simple mathematical functions (e.g., mean, dev, rate, max, min, delta, etc.) and near real-time alerting based on monitoring rules. Also, a graphical dashboard (Grafana) will provide charts and notifications representing the current operational status of each one of the monitoring sources.

Northbound Interface (NBI): The NBI is used for the connection between the NWDAF with external systems like ML frameworks. In order to support the different needs of each external system, the NBI provides several communication technologies, as alternatives that can fit to any requirements, namely RESTful APIs and Publish/Subscribe schemas. The RESTful APIs are especially useful for exchanging management information between different systems (e.g., alerts notifications, subscriptions to specific services, etc.), while Pub/Sub mechanisms are very convenient on data streaming (e.g., executing ML models, etc.).

3.4.2.2 5GS Telemetry Data Collector and C-MDAF

The resources and methods used in the API for the implementation of the mandatory operations of the Performance Data Streaming service (O1 PM VES) are provided in Table 12. All URIs are preceded by {MnSRoot}/StreamingDataReportingMnS/{version}.

Service operation	API URI	Method
establishStreamingConnection	/connections	HTTP POST
	/connections/{connectionId}	HTTP GET (Upgrade)
terminateStreamingConnection	/connections/{connectionId}	WebSocket Close frame sent and WebSocket Close frame received
reportStreamData	/connections/{connectionId}	WebSocket Data frame sent
addStream	/connections/{connectionId}/streams	HTTP POST
deleteStream	/connections/{connectionId}/streams	HTTP DELETE

Table 12: Resources and methods used for the O1 Performance Data Streaming service

3.4.3 Requirements' Mapping

The telemetry module contributes towards the fulfilment of the following system requirements specified in D1.1:

No.	ID	Requirement	Note
1	REQ-NET-01	The network shall conform to 3GPP 5G network requirements, architectures and interfaces in both the 5GC and NG-RAN.	Ongoing
2	REQ-NET-03	The system shall support O-RAN standard architecture & interfacing including the support of RIC VNFs.	Ongoing
3	REQ-NET-04	The system shall support 3GPP release Rel 15 SA.	Ongoing
4	REQ-NET-05	The system may support 3GPP release Rel 16 SA.	Planned for NWDAF features
5	REQ-NET-06	The system shall support control and user plane separation in the 5GC and deployment of the UPF (and DN) in both core and edge (MEC).	Ongoing
6	REQ-NET-07	The system shall support control and user plane separation in the NG-RAN.	Ongoing
7	REQ-NET-12	The system shall support virtualised RAN functions.	Ongoing
8	REQ-NET-13	The system shall support virtualised CN functions.	Ongoing
9	REQ-NET-16	The Core Network should report System KPI to the monitoring system.	Planned
10	REQ-NET-26	O-RAN Operations and Maintenance Interface (O1) shall be protected.	Planned
11	REQ-NET-31	The system shall be able to detect network failures and malfunctions.	Planned for the telemetry module
12	REQ-NET-44	The system should support data collection from NFs residing on the core and access part of the 5GS.	Ongoing
13	REQ-NET-45	NWDAF should be supported for network load performance computation and future load prediction.	Planned
14	REQ-NET-46	The NWDAF should support subscriptions from multiple network elements and entities.	Done
15	REQ-NET-47	The NWDAF should deliver event information and predictions to its subscribers.	Done
16	REQ-NET-48	The system should support intelligent processing of collected data.	Ongoing

No.	ID	Requirement	Note
17	REQ-NET-49	The system shall support interconnection of the AF with the NWDAF.	Done from NWDAF side
18	REQ-MAN-10	The service orchestrator shall be connected to the system monitoring and to the infrastructure involved in the necessary actions.	Ongoing
19	REQ-MAN-19	The solution should be able to provide real-time KPIs to the Orchestration platform mainly related to the QoS of services in order to guarantee SLAs.	Planned
20	REQ-MAN-22	The system shall provide monitoring mechanisms to continuously check the performance and status of the active VNFs.	Ongoing
21	REQ-MAN-31	The system should support services running in lightweight VMs or Docker containers.	Done for the telemetry module

Table 13: Requirements' mapping of the NWDAF

3.4.4 Technical Challenges

The development of the Affordable5G telemetry framework poses several technical challenges, which are mainly related to the efficient management of the data collected from heterogeneous sources, the need to support standardized mechanisms for deriving performance data from the 5G core and the O-RAN VNFs, and the need to integrate telemetry with ML-based network and management data analytics. In the following, we discuss each one of these challenges.

The telemetry framework depicted in Figure 34 collects performance data from the O-RAN, the 5G core network, the transport network, and the cloud infrastructure. This framework builds around storing time series data and processing them through the Prometheus open-source monitoring and alerting toolkit. In this respect, a technical challenge is to identify the minimum set of information that has to be stored along with the appropriate Prometheus recording and alerting rules that are needed to keep storage requirements low, while preserving the information required for efficient data analytics.

According to the description in section 3.4.1.1, the NWDAF collects data from the 5G core NFs through the corresponding Nnf interface. To be able to use this standardized approach, this interface should be provided by the 5G Core NFs that will be involved in the demonstration of the NWDAF functionality. In case such an interface is not natively implemented by the NFs, an adaptation is needed to convert between the mechanism implemented on the NF side for event transmission and the mechanism specified for event exposure to NWDAF. A similar technical challenge is involved in the support of event data streaming from the NFs of the O-RAN based on the O1 interface discussed in section3.4.1.2.

Integrating ML-based data analytics at the network and management levels of the Affordable5G system, through the NWDAF and the C-MDAF, respectively, poses another technical challenge. This notably has to do with a) the identification of suitable use cases for each one of these functions, b) the development of a mechanism for ML model serving, c) the implementation of standardized interfaces for analytics provisioning, and d) the investigation of possible exploitation of the NWDAF analytics information by the C-MDAF. In this context, the project is constantly monitoring the relevant, ongoing standardization efforts with the goal to provide solutions, which, to the extent possible, will be in line with these efforts.

3.4.5 Plans for the Integration

The current integration plan regarding the telemetry and data analytics framework includes the following:

- Integration of the developed NWDAF with 5G Core NFs. A first step towards this integration is to derive the performance metrics that are available from the Athonet's SA 5G core, which is currently set up in the Malaga platform.
- Integration of the VES data streaming mechanism with the O-RAN NFs. Priority will be given to the performance monitoring of the NFs provided by the dRAX (i.e., near-RT RIC and CU), the latest version of which has been installed in the Castelloli testbed. A VM for the telemetry component, including the NWDAF component, has been set up on the testbed's cloud infrastructure.
- Integration of the C-MDAF with the ML analytics module depicted in Figure 34.
- An ML module (optimizing the power allocation of RUs in a RAN serving users with specific throughput requirements) is currently implemented as an xAPP running on dRAX (see also section 3.5.4 regarding the option to develop the ML model for local execution at the near-RT RIC). After validating the ML module as an xAPP, we will also address the integration with the Affordable5G AI/ML framework depicted in Figure 40.
- Integration of the components providing cloud infrastructure telemetry and transport network performance measurements.

3.5 AI/ML framework

3.5.1 Description and Motivation

The fifth generation of cellular networks is expected to handle a massive number of devices offering unprecedented bandwidth and response time. Such a dense and demanding scenario poses several management challenges due to the increasing network complexity. Notwithstanding, recent mobile architectures are leaning towards a service-based architecture, which in conjunction with the Network Function Virtualization (NFV) and the function disaggregation paradigm is embracing unparalleled levels of flexibility and programmability. These facts, in addition to the new set of analytics functions available in recent mobile architectures (NWDAF, MDAF, etc.) are enabling the proliferation of AI/ML algorithms aiming to empower autonomous network management, heavily facilitating, and automating the orchestration of the network resources.

The development of AI/ML algorithms requires a set of tools in order to facilitate building, training, evaluating and serving procedures. Affordable5G, as an end-to-end 5G solution, is designed to support and facilitate the development and execution of AI/ML algorithms including an AI/ML framework in its architecture. To make easier the development and execution of AI/ML algorithms, Affordable5G has adopted one of the most popular open-source AI/ML libraries, TensorFlow. Due to its popularity, TensorFlow libraries are supported in a huge variety of AI/ML frameworks, having Acumos AI and TensorFlow's own toolkit as reference AI/ML frameworks. Within the scope of Affordable5G an evaluation of both frameworks has been performed to choose the most suitable one to fit in the Affordable5G system architecture.

Acumos AI [40] is an open-source platform for developing, integrating, training and deploying Artificial Intelligence models. Acumos AI project was originally developed by Tech Mahidra, Amdocs, AT&T, Orange and other members, being now a hosted project of the LF AI & Data foundation and freely distributed under Apache 2.0 license. Acumos is language agnostic and supports toolkits and libraries such as TensorFlow, SciKit Learn, RCloud or H2O among others and a huge variety of languages including Python, Java or R. Acumos offers a set of extensible mechanisms for packaging, sharing, licensing, and deploying AI models and publishing in a marketplace facilitating the distribution of AI models among different systems. The model packaging is done in containers independently of the coding language, an ideal approach for microservice-based architectures. Acumos supports several cloud-based run-time environments including AIC, Azure, AWS or private datacentres, requiring only a container management tool (e.g., Docker) for executing the containerized models.

TensorFlow [41] is an open-source set of libraries and toolkits for numerical computation and large-scale machine learning. It was created by the Google Brain team for internal use in Google and was publicly released in 2015 under Apache 2.0 license for freely utilization. TensorFlow ingest the data in the so-called *tensors*, multi-dimensional arrays of high dimensions, allowing TensorFlow to easily handle large amounts of data. TensorFlow works using data flow graphs that includes nodes and edges. Such mechanism allows TensorFlow to execute the code in a distributed way across a cluster of computers while using GPUs, supporting the CUDA toolkit for GPU acceleration in NVIDIA GPUs. The available libraries and toolkits in TensorFlow cover all the AI/ML workflow, including the development of AI/ML pipelines with *TensorFlow Extended* (TFX), as well as the model execution and serving with *TensorFlow Serving* (TFS).

TensorFlow Extended [42] is a platform for building and managing AI/ML workflows/DAGs in production-ready environments, it provides a set of components for building an AI/ML pipeline, a set of libraries that provides functionalities to the different components and a toolkit for building the pipeline that can be orchestrated with several platforms such as Kubeflow or Apache Airflow.

A TFX pipeline defines the series of operations performed in the AI/ML workflow/DAG. The set of TFX components available in TFX are depicted in Figure 39 and explained as follows:

- ExampleGen: the starting component of the AI/ML pipeline in charge of ingesting the data and splitting into datasets.
- StatisticsGen: is in charge of calculating the statistics for the AI/ML dataset.
- SchemaGen: analyzes the statistics generated by the StatisticsGen component and creates a data schema.
- ExampleValidator: examines the datasets looking for anomalies and missing values.
- Transform: is at the helm of performing feature engineering on the datasets.
- Trainer: is in charge of training the AI/ML model.
- Tuner: tunes the hyperparameters of the AI/ML model.
- Evaluator: evaluates the training results, helping to validate the AI/ML models, ensuring the required performance for production.
- InfraValidator: is responsible for checking that the model is servable from the infrastructure.
- Pusher: exports of the model on a servable format.

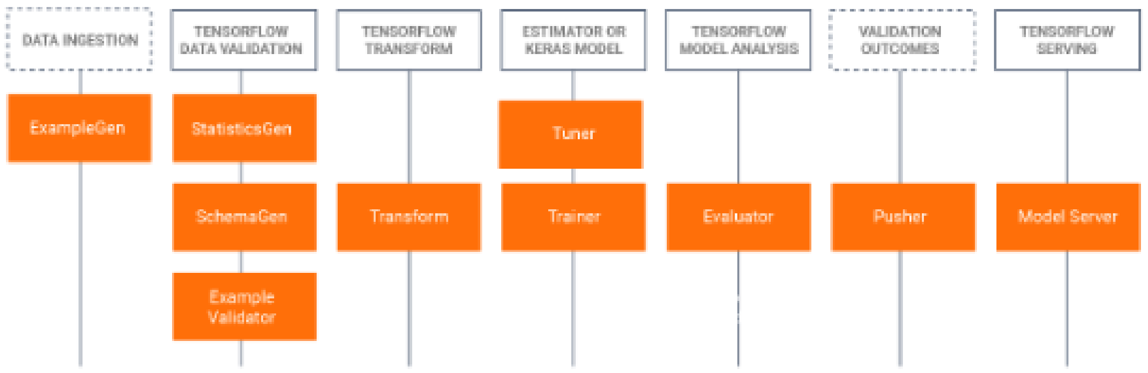


Figure 39: TFX libraries (image extracted from [42])

Once the model is developed and exported in a servable format the model is deployed in a serving infrastructure, TensorFlow offers *TensorFlow Serving* as a flexible, high-performance serving system for production environments. TFS can be installed natively with *pip*, *apt* or it can be built from source or executed in a Docker container provided by TensorFlow. In addition, TensorFlow offers a set of libraries in C++ for constructing a customized model server. Once TFS is running it can serve one or multiple models exported in *SavedModel* format simultaneously. *SavedModel* is the universal serialization format for TensorFlow models, *SavedModels* contain the complete TensorFlow program, including trained parameters, variables, and additional files such as vocabularies. As a result, it does not require to build the original model in every run, simplifying the sharing and deploying procedures either in TFS or other TensorFlow solutions such as TFLite, TensorFlow.js or TensorFlow Hub. TFS supports inference request through either gRPC or REST, allowing the access to the serving models in a language-agnostic manner.

Once having evaluated both solutions, and after an extensive analysis, we have decided to focus on the TensorFlow ecosystem in the Affordable5G project. The main reason for this selection is the maturity and stability of the project, in addition to the quality and quantity of available documentation, making easier the integration of TensorFlow into the Affordable5G architecture.

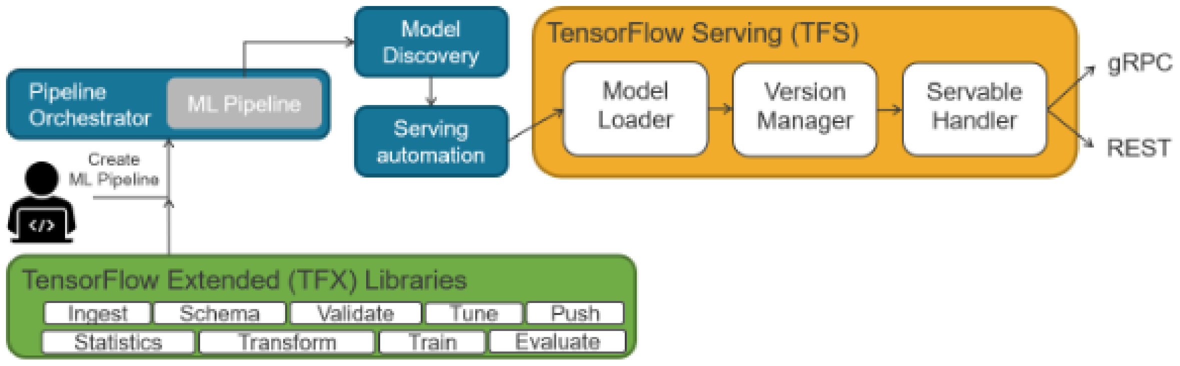


Figure 40: Affordable5G AI/ML Framework architecture

Therefore, for the Affordable5G AI/ML framework we architected a solution based on a pipeline orchestrator for TFX libraries in addition to TFS for the inference requests, as depicted in Figure 40.

For the pipeline orchestrator we have evaluated the most used alternatives: Apache Airflow [43] and Kubeflow [44]. Based on the study performed in [45] we decided to use Airflow as pipeline orchestrator since it is the most mature and popular orchestrator available, having more and better documentation and stability than other solutions. The AI/ML orchestrator will allow the AI/ML developer to programmatically and sequentially run, schedule and monitor AI/ML Direct Acyclic Graphs (DAGs).

A DAG is nothing but a collection of tasks you want to run, properly organized and connected, reflecting relationships and dependencies. Figure 41 depicts the DAG extracted from the exemplary TensorFlow Chicago taxi AI/ML model [46]. This DAG is developed using the TFX libraries, including modules for ingesting the dataset (CsvExampleGen), statistics (StatisticsGen and SchemaGen), validation (ExampleValidator), dataset transforming (Transform), training (Trainer), evaluation of the training results (Trainer), and exporting as *SavedModel* (Pusher).

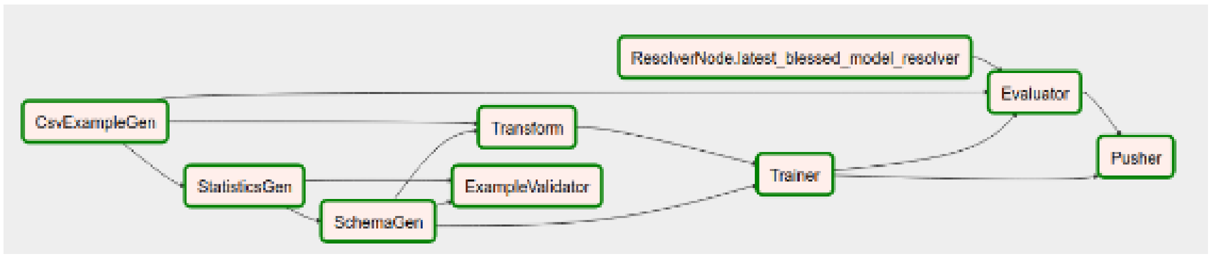


Figure 41: DAG from Chicago taxi A/ML exemplary model

The Pusher TFX module just exports the model in a servable format (*SavedModel*) but does not automate the serving procedure. Thus, once the AI/ML model is exported by the Pipeline Orchestrator, the *Model Discovery* and *Serving Automation* modules depicted in Figure 40, specifically developed for Affordable5G, automatically import the model into TFS allowing the model to be accessible either in a centralized manner or distributed in different components of the architecture. The *Model Discovery* module will periodically scan the model folder to find newer models, or newer versions of already deployed models, while *Serving Automation* module triggered by the *model discovery* module will copy the model files (if needed) and update the *model.config* file of TFS in order to serve the new exported models. TFS allows to serve multiple models simultaneously, managing the versioning and model labelling, allowing to separate testing from production environments. To do it TFS includes three modules:

- The model loader that loads the AI/ML into models from the model.config file into the model server application. It continuously monitors the the model.config files and the directory where the models are located.
- The version manager that handles the versioning of the AI/ML models.
- The servable handler that is in charge of serving the models. This module supports two different communication protocols, gRPC and REST.

The implementation of a centralized AI/ML model serving infrastructure will be done in a dedicated server accepting inference requests either through gRPC or REST, the two communication protocols supported by default in TFS.

When the local execution of AI/ML models is needed the TFS model server can be easily adapted either by using the already provided model server application or by building a model server from scratch using the available C++ libraries. This integration work will be performed in Affordable5G for the local execution of AI/ML models in the dRAX (O-RAN near-Real-Time RIC). For the dRAX integration we will create an xAPP based on the xAPP framework and architecture detailed in Figure 41 and Figure 42. TFS xAPP initial design is presented in Figure 42, the proposed application executes the model server inside the xAPP Core and extracts all the required data for the AI/ML models from the dRAX RIC Databus using the provided *in queue* connected to the *databus listener*. Once the data are properly extracted from the *in queue*, the data are processed by the xAPP *processor* and sent via REST/gRPC to the TFS model server instance for inference requests. After that, the model output received from the model server is then processed by the xAPP *processor* and pusblished on the dRAX RIC Databus through the provided *out queue* connected to the *databus publisher*, allowing other xAPPs connected to the dRAX RIC Databus to consume these inferenced data for network optimization purposes. In addition, the xAPP includes the xAPP API that enables the communication with the dRAX RIC, allowing to access and modify the configuration of the xAPP that is saved in the xAPP database.

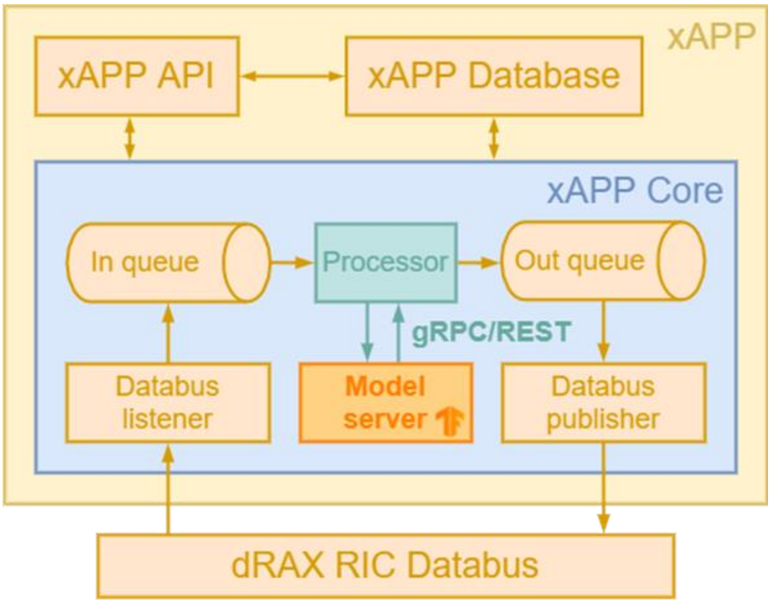


Figure 42: TensorFlow Serving in xAPP

3.5.2 Interfaces

The Affordable5G AI/ML model serving infrastructure, either in the centralized solution or internally in the TFS xAPP, uses the gRPC and REST interfaces provided by TFS:

- Google Remote Procedure Call or gRPC is a lightweight and high-performance communication protocol based on Protocol Buffers (protobuf), a mechanism to serialize structured data. GRPC provides faster inferences since it is more network efficient than REST, requiring smaller payloads, hence introducing less overhead in the data communication. In consequence, gRPC is the most recommended option when the server will handle a big number of requests or when the maximum performance is needed, requiring low-latency and high-speed throughput communications. The TFS gRPC implement remote procedure calls for the model serving of the following APIs:
    - Model Status API: follows the *ModelService.GetModelStatus* gRPC of the *model_service* protocol buffer and returns the status of the model.
    - Model Metadata API: follows the the *PredictionService.GetModelMetadata* gRPC of the *prediction_service* protocol buffer and returns the metadata of the model.
    - Classify API: developed for classification problems, it follows the Classify method of *PredictionService.Classify* gRPC of the *prediction_service* protocol buffer.
    - Regress API: created for regression tasks, it follows the Regress method of *PredictionService.Regress* gRPC of the *prediction_service* protocol buffer.

   - Predict API: focused on prediction requests, it follows the Predict method of *PredictionService.Predict* gRPC of the *prediction_service* protocol buffer.

  More details about the available TFS protocol buffers and libraries can be found in the TensorFlow Serving official documentation [47].

- Representation State Transfer or REST is a synchronous communication method based on HTTP. REST is format agnostic, and the payload is commonly encoded using the human-readable JSON format, but more format are supported. The main benefit of using REST over gRPC is the development simplicity, REST is easier and faster to implement, and debug compared to gRPC. Consequently, the utilization of REST is recommended when the server will not handle a big number of requests and the performance is not the main concern, prioritizing a less complex and simpler development. As well as the TFS gRPC API, the REST API implements the following endpoints:
   - GET Model Status API: returns the status of the model running in the model server, this API is placed in http://host:port/v1/models/${MODEL_NAME}
   - GET Model Metadata API: returns the metadata of the served model, the URL of this API is http://host:port/v1/models/${MODEL_NAME}/metadata
   - POST Classify API: it supports structured calls to TFS Classification API for classification problems, the API URL is http://host:port/v1/models/${MODEL_NAME}:classify
   - POST Regress API: it supports structured calls to TFS Regression API for regression problems, the URL where this API is listening is http://host:port/v1/models/${MODEL_NAME}:regress
   - POST Predict API: it supports structured calls to TFS Prediction API for prediction tasks, the URL where this API is accessible is http://host:port/v1/models/${MODEL_NAME}:predict

  All this REST API supports different model versions and label, serving by default the latest version of the model, an specific version or label can be accessed just including /versions/${VERSION} and/or /labels/${LABEL} just after ${MODEL_NAME} in the URL (e.g., http://host:port/v1/models/${MODEL_NAME}[/versions/${VERSION}|/labels/${LABEL}]:predict for prediction). More information about the TFS REST API can be found in the official TensorFlow documentation [41].

In addition to the REST and gRPC APIs, the TFS xAPP also implements the xAPP API that enables the communication with the dRAX RIC. This API is included in the xAPP Framework, and it is detailed in Section 2.1.4.

3.5.3 Requirements' Mapping

The Affordable5G AI/ML framework will satisfy top-level requirements identified in D1.1 [5] with regard to the NG-RAN edge, supporting open extensibility and intelligence in both the control-plane (near-RT RIC) and the user-plane (MEC), in addition to the network and MANO requirements listed in the following table:

No.	ID	Requirement	Note
1	REQ-NET-45	NWDAF should be supported for network load performance computation and future load prediction.	Ongoing

No.	ID	Requirement	Note
2	REQ-NET-47	The NWDAF should deliver event information and predictions to its subscribers.	Ongoing
3	REQ-NET-48	The system should support intelligent processing of collected data	Ongoing
4	REQ-MAN-19	The solution should be able to provide real-time KPIs to the Orchestration platform mainly related to the QoS of services in order to guarantee SLAs.	Ongoing

Table 14: Requirements' mapping of the AI/ML framework

3.5.4 Technical Challenges

The utilization of ML/AI algorithms will enable autonomous network management, allowing higher degrees of network automation and optimized resource orchestration. Therefore, there is need for tools and platforms in order to facilitate the development process of AI/ML algorithms, including building, training, evaluating and serving capabilities. For this reason, Affordable5G has adopted one of the most popular open-source AI/ML libraries, TensorFlow, as well as an AI/ML orchestration platform based on Apache Airflow. However, the tools and libraries provided by TensorFlow are for general purpose and are not specifically tailored for the purpose of this project, posing several technical challenges to its adoption in the 5G ecosystem. Hence, the TensorFlow and Airflow ecosystems should be adapted and seamlessly integrated with the Affordable5G architecture, and the different components involved in the resource orchestration procedures.

This challenging integration will consist firstly in the integration of Airflow for the AI/ML model development in the Affordable5G architecture, providing a framework that will easily orchestrate AI/ML workflows. Airflow provides the executable files of the developed model, so secondly, we will integrate and develop different solutions based on the TensorFlow model server application for serving and sharing the exported model. One serving solution will be centralized, allowing all the different components of the Affordable5G architecture to consume and utilize AI/ML models through open APIs, enabling higher degrees of automation and efficiency. An alternative serving solution will be developed specifically for the O-RAN near-Real-Time RIC, allowing to consume the models directly from the dRAX as an xAPP, this will allow to directly access the data provided by the dRAX RIC Databus and publish back the model prediction/inferences directly as a new Kafka topic in the dRAX RIC Databus, offering this information to other xAPPs for intelligent RAN orchestration. Affordable5G will also provide serving automation solutions for the seamless integration of Apache Airflow and both models serving solutions (centralized and xAPP).

As a conclusion, Affordable5G is addressing the challenge of including the utilization of AI/ML algorithms in the Management and Orchestration layer, including O-RAN through the near-RT RIC, with the purpose of making these orchestration and management processes intelligent and adaptative, incorporating the monitoring and telemetry information coming from the recently introduced network analytics functions and the RIC Databus. Such innovation allows to react and flexibly adapt the resource utilization to the network changes and stringent requirements of the new generation of mobile networks.

3.5.5 Plan of the Integration

The AI/ML framework development and integration tasks can be divided in four different branches, that will follow the subsequent order:

1. AI/ML model development and orchestration framework: it is based on Apache Airflow and will be developed and integrated with the architecture in the first place, allowing the partner consortium to start developing AI/ML algorithms in a dedicated platform. It is a task in progress and an initial version of the platform will be provided before the first year of the project.
2. Centralized model server: it will allow all the architectural components to consume the developed models in the AI/ML model development and orchestration framework through open APIs based on REST and gRPC. At the initial stages of the development and integration process, the model files provided by the AI/ML model development and orchestration framework will be manually loaded into the server application, an automated process will be delivered lately when the task 4 is finished. At this moment It is an ongoing task, and an initial version of the centralized model server will be provided before the first year of the project.
3. xAPP model server: it will allow to develop models specifically for the O-RAN near-Real-Time RIC and accessing the model output from other xAPPs. At the initial stages of the development and integration process, the model files provided by the AI/ML model development and orchestration framework will be manually loaded into the xAPP server application, an automated process will be delivered lately when the task 4 is finished. It is a task in development progress and an initial version of the xAPP model server is expected to be provided at the beginning of the first year of the project.
4. Serving automation mechanisms: once all the previous components are developed and integrated with the architecture, we will develop automation mechanisms for automatically serve the model files provided by Airflow in the two different model server applications (centralized and xAPP). This task has not started yet, and it is expected to be provided during the second year of the project.

4 INFRASTRUCTURE LAYER

4.1 Transport and synchronization

4.1.1 Description and Motivation

Open Fronthaul requires very tight synchronization between O-RU and O-DU. PTP protocol is the main component of O-RAN synchronization plane.

Typically, an O-RU function is a closed combination of a hardware and software; time usually is provided by an embedded GNSS receiver or, alternatively via Fronthaul Ethernet interface. In Affordable 5G project, O-RU synchronization software is out of scope.

O-DU in turn is based on a COTS platform, running Linux software. In Affordable5G project we use a combination of open and proprietary software in order to discipline the Linux System time to an external PTP Grand Master (GM).

Figure 43 illustrates O-RAN Synchronization path, planned for Castelloli site.

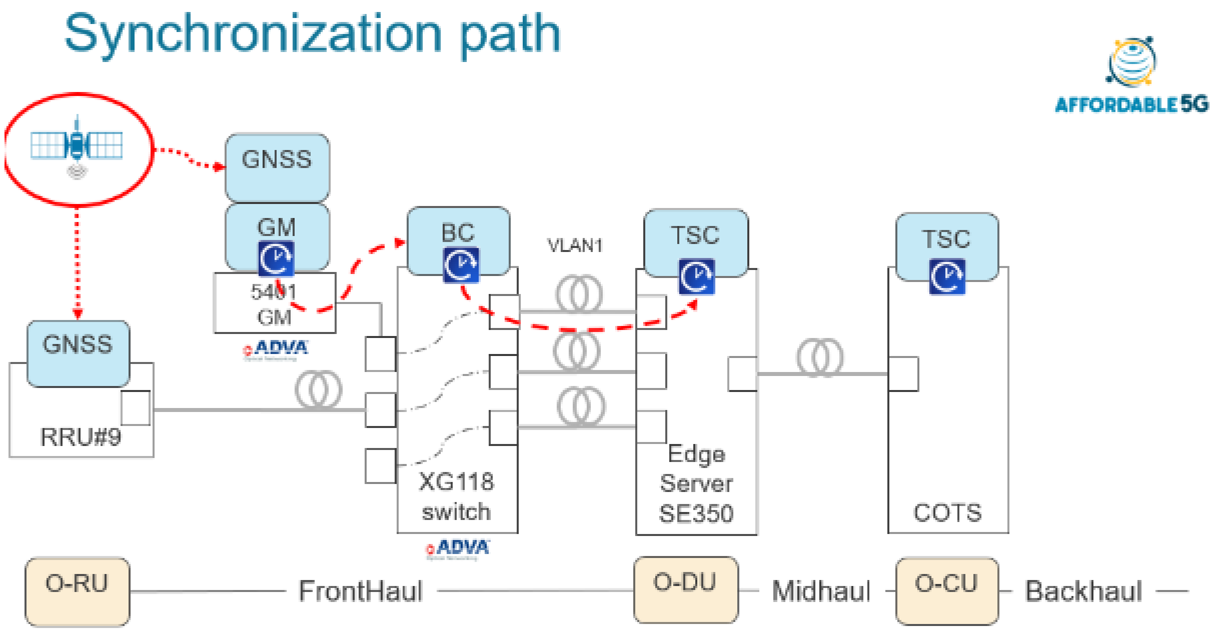


Figure 43: RAN Synchronization path example (Castelloli)

Legend:

- GM – PTP Grand Master (OSA 5401)
- BC – PTP Boundary Clock (ADVA XG-118 FrontHaul switch)
- TSC – Telecom Slave Clock

GNSS is the source of the synchronization, used by O-RU and O-DU (O-CU).

The GM sends PTP streams towards BC, and latest distributes the timing streams towards one or more Telecom Slaves (TSC). TSC running on Linux platform periodically updates system time to be aligned with BC time. All software components of O-DU (and O-CU) are using Linux system time, which is aligned with GNSS and thus with O-RU.

We use *linuxptp* open-source package as a basic timing component for either O-DU or O-CU. This package implements PTP Telecom Slave Clock.

Required software version is 3.0 or higher. The package can be compilated from source files: https://github.com/openil/linuxptp.

The package includes two user-space applications:

- ptp4l – implements PTP slave function.
- phc2sys – the code for disciplining system time to the time clock recovered by ptp4l.

Besides the standard *ptp4l* software, we are improving our proprietary TSC software product, called *SoftSync Client*. This product originally was developed for financial market, and we are adopting its functionality to fit to the O-RAN requirements. Currently we are performing a verification of *SoftSync* as well as its comparison to *ptp4l*, using pre-integration staging setup created as a part of WP2.

Ptp4l can be used for POC, but it lacks functionality, which is important for a real 5G RAN deployment.

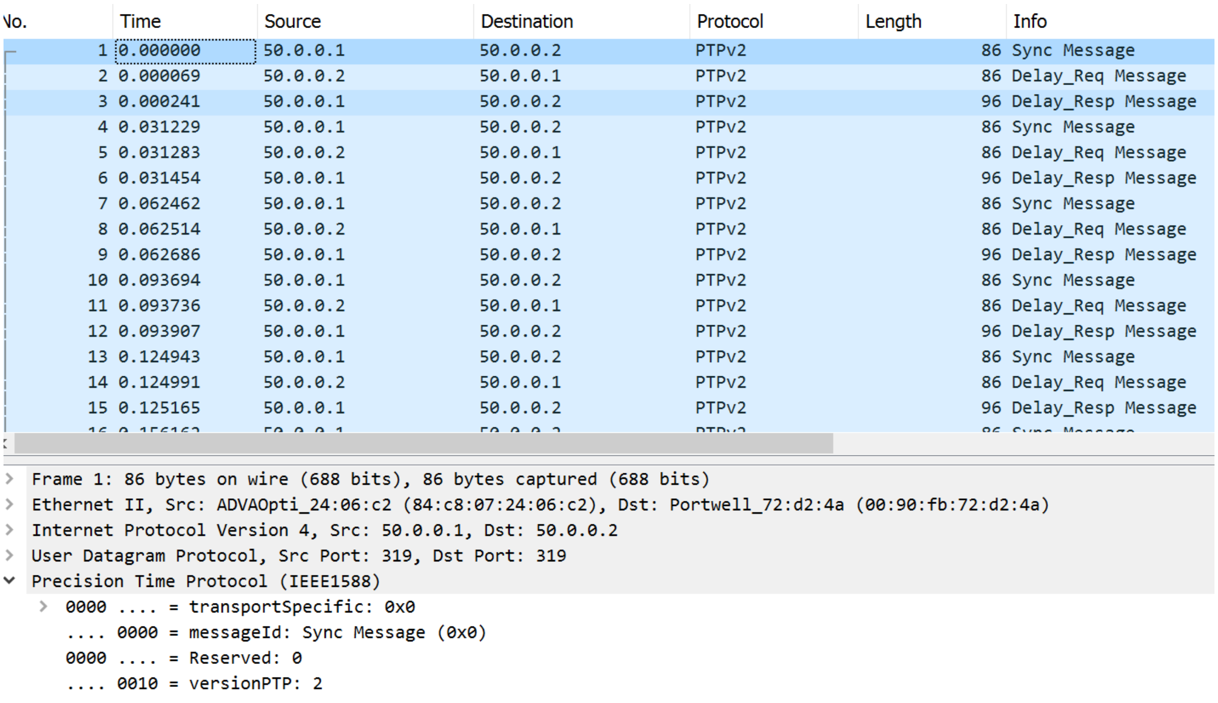
Main advantage of *SoftSync* product is its manageability: an operator can receive statuses and alarms reflecting the quality of the system time, thus simplifying the troubleshooting of synchronization plane. *SoftSync* has GUI, SNMP interface and can be integrated into large network managing hierarchy.

Additionally, Telecom PTP profile differs from Enterprise PTP profile, which was originally supported by *SoftSync*.

We are integrating the packages as system services; currently it works on CentOS 7 and Fedora Linux. Using Debian (Ubuntu) distro will require some extra tailoring.

4.1.2 Interfaces

The PTP protocol flavour used for the project is PTPv2 with UDP IPv4 encapsulation. A typical PTPv2 message exchange is illustrated in the WireShark capture in Figure 44.



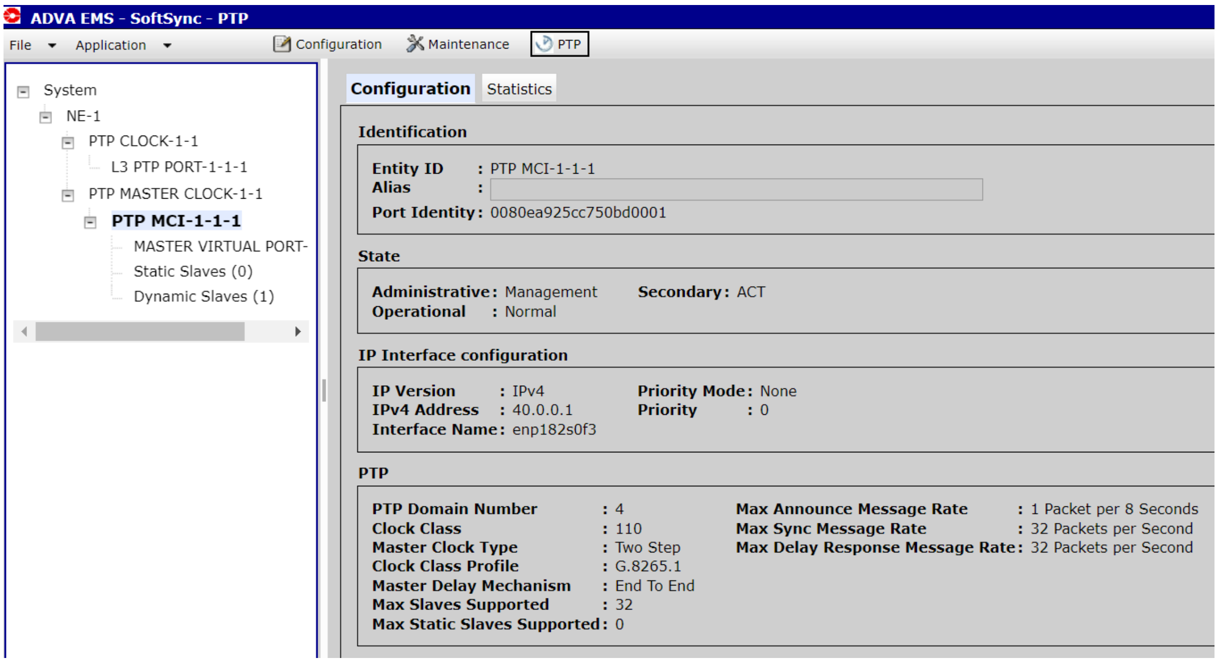
```
No.   Time       Source     Destination  Protocol  Length  Info
    1 0.000000   50.0.0.1   50.0.0.2     PTPv2         86  Sync Message
    2 0.000069   50.0.0.2   50.0.0.1     PTPv2         86  Delay_Req Message
    3 0.000241   50.0.0.1   50.0.0.2     PTPv2         96  Delay_Resp Message
    4 0.031229   50.0.0.1   50.0.0.2     PTPv2         86  Sync Message
    5 0.031283   50.0.0.2   50.0.0.1     PTPv2         86  Delay_Req Message
    6 0.031454   50.0.0.1   50.0.0.2     PTPv2         96  Delay_Resp Message
    7 0.062462   50.0.0.1   50.0.0.2     PTPv2         86  Sync Message
    8 0.062514   50.0.0.2   50.0.0.1     PTPv2         86  Delay_Req Message
    9 0.062686   50.0.0.1   50.0.0.2     PTPv2         96  Delay_Resp Message
   10 0.093694   50.0.0.1   50.0.0.2     PTPv2         86  Sync Message
   11 0.093736   50.0.0.2   50.0.0.1     PTPv2         86  Delay_Req Message
   12 0.093907   50.0.0.1   50.0.0.2     PTPv2         96  Delay_Resp Message
   13 0.124943   50.0.0.1   50.0.0.2     PTPv2         86  Sync Message
   14 0.124991   50.0.0.2   50.0.0.1     PTPv2         86  Delay_Req Message
   15 0.125165   50.0.0.1   50.0.0.2     PTPv2         96  Delay_Resp Message

> Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
> Ethernet II, Src: ADVAOpti_24:06:c2 (84:c8:07:24:06:c2), Dst: Portwell_72:d2:4a (00:90:fb:72:d2:4a)
> Internet Protocol Version 4, Src: 50.0.0.1, Dst: 50.0.0.2
> User Datagram Protocol, Src Port: 319, Dst Port: 319
v Precision Time Protocol (IEEE1588)
    > 0000 .... = transportSpecific: 0x0
      .... 0000 = messageId: Sync Message (0x0)
      0000 .... = Reserved: 0
      .... 0010 = versionPTP: 2
```

Figure 44: PTPv2 protocol messages example

The GNSS disciplined system time is available via Linux system calls.

The management interfaces of SoftSync are WebGUI and SNMP. An example of the WebGUI is shown in Figure 45.



Identification

Entity ID	: PTP MCI-1-1-1
Alias	:
Port Identity	: 0080ea925cc750bd0001

State

Administrative	: Management	Secondary	: ACT
Operational	: Normal		

IP Interface configuration

IP Version	: IPv4	Priority Mode	: None
IPv4 Address	: 40.0.0.1	Priority	: 0
Interface Name	: enp182s0f3		

PTP

PTP Domain Number	: 4	Max Announce Message Rate	: 1 Packet per 8 Seconds
Clock Class	: 110	Max Sync Message Rate	: 32 Packets per Second
Master Clock Type	: Two Step	Max Delay Response Message Rate	: 32 Packets per Second
Clock Class Profile	: G.8265.1		
Master Delay Mechanism	: End To End		
Max Slaves Supported	: 32		
Max Static Slaves Supported	: 0		

Figure 45: SoftSync WebGUI

4.1.3 Requirements' Mapping

Regarding the requirements on pilots and use cases identified in D1.1, transport and synchronization contributes to satisfying the following ones:

No.	ID	Requirement	Notes
1	REQ-NET-10	The system shall support several synchronization means: GNSS, PTP and SyncE.	Ongoing: GNSS and PTP
2	REQ-NET-19	The system shall support Ethernet based fronthaul with TSN functionality	Ongoing
3	REQ-NET-60	RU and DUs should be deployable in GNSS-less environment	Ongoing, DU only
4	REQ-MAN-40	The solution should be highly efficient in terms of energy consumption, computing resources and bandwidth	Ongoing

Table 15: Requirements mapping of transport and synchronization

4.1.4 Technical Challenges

The challenge to achieve the desired synchronization accuracy was to support 10GE NIC with low-quality (but cost-efficient) internal oscillator; those NICs are widely used in many modern COTS platforms, which is the reason why they are selected for Affordable5G. For example, Xeon-D Skylake CPU is one we are focusing on, and it has integrated X772 NIC. To find and validate a suitable i40e driver and its integration with the system took several iterations. The ptp clock recovery has been validated with the i40e driver version 2.14.13 on CentOS 7 Linux (3.10.0-1160.6.1.rt56.1139.el7.x86_64).

4.1.5 Plan of the Integration

The following integration steps should be performed for Castelloli site:

- Installation of FSP 150XG 118 Fronthaul switch, pre-configured.
- Installation of OSA 5401 GM, connected to GNSS antenna.
- Installation of the desired i40e driver and SoftSync on Edge Server SE350.

4.2 Time-sensitive network

4.2.1 Description and Motivation

Nowadays, the private 5G network are gaining relevance in industrial environments. This is the main reason for including TSN concept in Affordable5G. TSN adds mechanisms in order to evolve actual ethernet networks, such as low latency, deterministic communications or QoS.

The UXM 5G setup shown in Figure 46 allows emulating a full 5G network, including the radio and core part. In addition, a UE is remotely controlled inside of the anechoic chamber using the 2 PCs. This setup achieves to provide a whole 5G test environment with plenty of configuration parameters that can be modified easily.

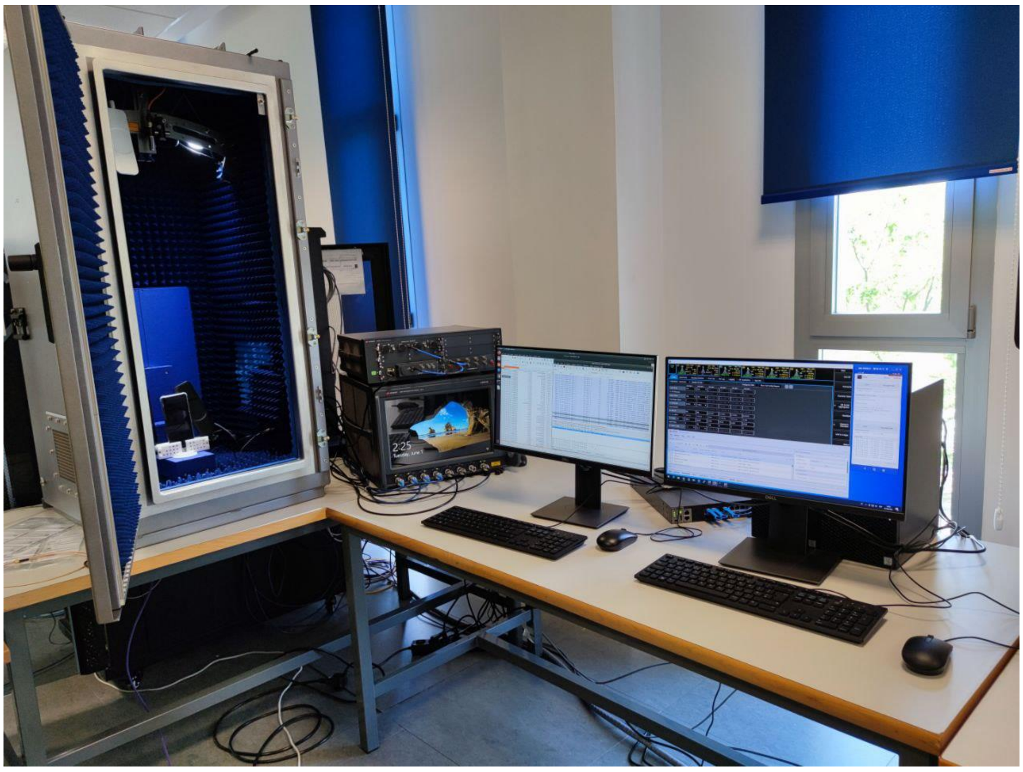


Figure 46: Keysight UXM 5G setup

Figure 47 represents the Relyum TSN setup, which consists of 1 TSN bridge that acts as a switch and 2 TSN PCIe cards [48] that can act as switch or endpoint. In this case they will act as 2 TSN endpoints, well known in TSN like talker and listener.

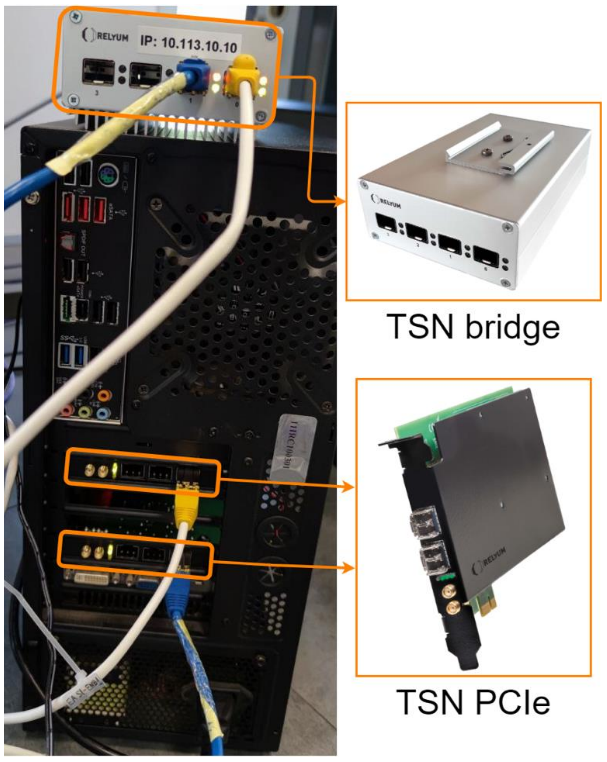


Figure 47: Relyum TSN setup

4.2.2 Interfaces

The relevant interfaces between TSN and 5G domains are the following:

- TSN end station – CUC: It allows that the TSN end stations can communicate with CNC for making request related to the deterministic communications.
- CUC – CNC: It allows the forwarding the requests from TSN end stations.
- CNC – TSN bridge: It allows to configure the TSN bridges, determining the performance of TSN traffic flows.
- CNC – AF: Communication between TSN domain and 5G network.

TSN bridge / end station – TSN TT: It allows the translation of the TSN traffic from/to TSN domain and 5G network.

4.2.3 Requirements' Mapping

The following table shows the requirements that are actually ongoing and the planned ones for the future. These requirements are indicated in D1.1, section 3.6.

No.	ID	Requirement	Note
1	REQ-NET-05	The system may support 3GPP release Rel 16 SA	Ongoing
2	REQ-NET-10	The system shall support several synchronization means: GNSS, PTP and SyncE	Planned
3	REQ-NET-19	The system shall support Ethernet based fronthaul with TSN functionality	Ongoing
4	REQ-NET-51	The 5G system should be totally compatible with TSN domain	Ongoing
5	REQ-NET-52	The infrastructure shall support high performance Ethernet applications	Planned

Table 16: Requirements mapping of the time-sensitive network

4.2.4 Technical Challenges

The main technical challenge is related to the compatibility between TSN domain and the 5G system. In Rel16 [2] is described the integration process of TSN in 5G networks in order to support the synchronization and traffic scheduling challenge. Currently, the testing and researching about this topic are essentials to achieve a full setup, whose features can be similar to the desired ones in terms of synchronization and traffic scheduling. Affordable5G is addressing this challenging integration, in addition to other technical challenges, such as the development for supporting several time sources or transporting ethernet traffic alongside the 5G network.

4.2.5 Plan of the Integration

Figure 48 depicts the current mapping of the work planned for TSN over 5G, previously described in D1.2. The idea is to combine different TSN parts from vendors and include all necessary components for reaching a full TSN setup. Some components will be added to this architecture for achieving TSN over 5G, such as TSN translators, NW-TT and DS-TT.

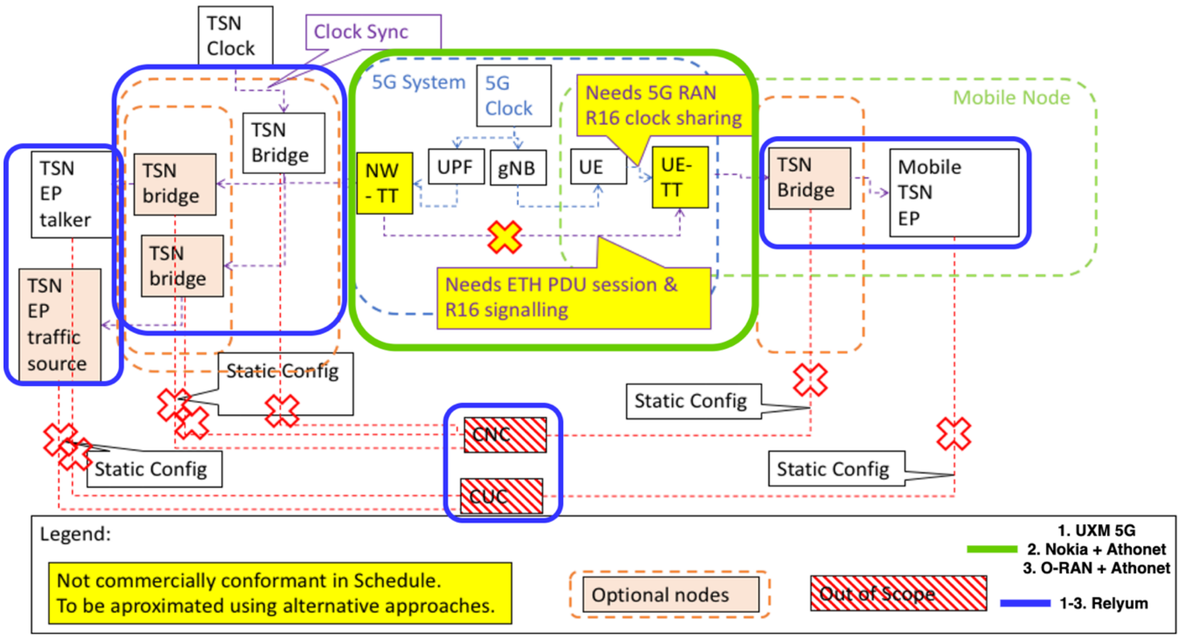


Figure 48: TSN mapping with commercial components

The integration process will consist of three steps. In the first step the UXM 5G emulator will be used for replacing the 5G system, hence it will act as TSN bridge. In the second one, the Nokia outdoor remote heads and the Athonet 5G core will replace the UXM 5G, acting as 5G system. In the last step the radio part will consist in the implementation achieved in the Affordable 5G scope, the O-RAN system connected to Athonet 5G core and the rest of TSN network will be part of the full TSN setup at UMA.

5 CONCLUSIONS AND NEXT STEPS

This deliverable presented the analysis and discussion of the software components adopted to build the Affordable5G architecture and its requirements specified in D1.2 to enable a cost-efficient roll-out of 5G private and enterprise networks. To this end, this document has studied and identified for each of the architectural layers of the Affordable5G system the specific software tools that satisfy the aforementioned requirements, the required interfaces, the encountered technical requirements and the roadmap towards completing the final version of the software capabilities of the project.

At the moment of writing this deliverable, the consortium has determined all the specific software artifacts that will provide the required capabilities to perform the pilots. Although such artifacts are currently at a different completion stage with regard to the final view of the project, all of them have reached an initial state mature enough to start the integration and validation of the technical developments in several scenarios at the test sites before conducting the Affordable5G pilots.

The methodology followed to select and start the integration of the software components of the Affordable5G system has encountered several technical challenges worth mentioning. On the one hand, the implementation and integration of the O-RAN stack, and in particular the integration of O-RAN 7.2 Fronthaul interface between RU/DU, as well as the existence of commercial 5G UEs for testing and validation, have risen several issues that must be tackled in order to bring this innovative point into the project breakthroughs. Moreover, the E2 interface able to offer multivendor support, especially with respect to network slicing is still an immature point that requires further integration steps, especially on features from 3GPP Rel-15 and Rel-16. Related to this, the ability of the 5GC in Affordable5G to distribute NFs at the edge and steer the data plane traffic, will play a key role on edge computing capabilities and also on the ability to deploy isolated network slices with independent services. Running applications at the edge requires of lightweight environments supporting the cloud native concept, which is being carried out in the project through the deployment of container-based network functions. One of the most sensitive points is the deployment of the software components at the pilot sites. This activity involves not only the deployment itself but also the integration of the components including, among others, the O-RAN stack, the orchestrators, and the monitoring capabilities and the telemetry functions from the 5GC required to train and feed the AI/ML engines that will become an essential part of the architecture.

At this stage, it is expected that the on-going developments in the different components of the network function layer, management, orchestration & automation layer, and infrastructure layer will be completed in line with the time plan as described in the Description of Action (DoA). In this way, the first functional and performance validation tests can start in September 2021. Based on the outcomes of these tests, required enhancements and modifications in each component will be performed to have a more sound and functional updated release. In addition, the components will be deployed in the testbeds for the complete system test and use case validations in WP4.

The timely and successful completion of WP3 work relies on the integration plans that have been detailed in this document for each of the components and that involve multiple cooperation activities between partners, including: integration between the RU of RunEL and the DU of Eurecom for the O-RAN 7.2 interface; integration of the OAI DU of Eurecom and the CU of ACC through the F1 interface; integration of the OAI DU of Eurecom and the near-RT RIC at the ACC dRAX through the E2 interface; integration between the CU of ACC and the 5GC of ATH via N2/N3 interfaces; integration between the O-RAN EMS of i2CAT and the O-RAN components (CU,DU,RU) of the different partners via O1 interface; integration between the near-RT RIC of ACC and the non-RT RIC of i2CAT via A1 interface; integration of the NBC orchestrator with the different containerized VNFs and applications of the different partners that need to be orchestrated; integration between the slice manager of i2CAT and the orchestrator of NBC; integration between the slice manager of i2CAT and the 5GC of ATH; integration between the non-RT RIC of i2CAT; integration between the NWDAF of NKUA and the 5GC of ATH; integration between the VES data streaming mechanism of NKUA and the dRAX of ACC; integration of the AI/ML framework of ATOS with the components of the architecture consuming AI/ML models; integration of the transport and synchronization equipment of ADVA together with the RU, DU and CUs deployed at the pilots sites; and integration of the TSN network with the 5GC of ATH and with the O-RAN system. Some of these integration activities are already in initial stages while some others are being just planned. Anyway, the Affordable5G consortium is confident that the challenges embraced by these integration activities will be accomplished. The final developments of each component and the outcomes of the integration will be reported in deliverable D3.2.

6 REFERENCES

[1] Affordable5G Consortium, “D1.2: Affordable5G building blocks fitting in 5G system architecture,” 2021.

[2] 3GPP, “TS 38.300 - version 16.2.0 Release 16: "NR; NR and NG-RAN Overall Description",” 2020.

[3] O-RAN Alliance, “O-RAN Architecture Description v1.0,” 2020.

[4] RunEL, “The O-RU (RRH) Brochure,” [Online]. Available: https://6f2268e7-1cdf-47df-b1d1-7ac1e2feef2a.filesusr.com/ugd/dd0425_a6565a02de5642e58e7859f1cb785a1a.pdf. [Accessed 19 07 2021].

[5] Affordable5G Consortium, “D1.1. State of the art, technical system requirements analysis and pilot element descriptions,” 2020.

[6] 3GPP, “TS 38.401 version 15.5.0 Release 15; “5G; NG-RAN; Architecture description,” 2019.

[7] Open Air Interface, “OAI 5G RAN Project,” Open Air Interface, [Online]. Available: https://openairinterface.org/oai-5g-ran-project/. [Accessed 14 07 2021].

[8] 3GPP, “TS 38.213 version 15.6.0 Release 15: 5G; NR; Pysical layer procedures for control,” 2019.

[9] 3GPP, “TS 38.331 version 15.6.0 Release 15: 5G; NR; Radio Resource Control (RRC); Protocol specification,” 2019.

[10] Small Cell Forum, “5G FAPI: PHY API Specification,” [Online]. Available: https://scf.io/en/documents/222_5G_FAPI_PHY_API_Specification.php3GPP. [Accessed 14 07 2021].

[11] 3GPP, “TS 38.321: "NR; Medium Access Control (MAC) protocol specification",” 2019.

[12] 3GPP, “TS 38.322 version 15.5.0 Release 15: 5G; NR; Radio Link Control (RLC) protocol specification,” 2019.

[13] 3GPP, “TS 38.323 version 15.6.0 Release 15: 5G; NR; Packet Data Convergence Protocol (PDCP) specification,” 2019.

[14] 3GPP, “TS 36.331 version 15.6.0 Release 15: LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification,” 2019.

[15] 3GPP, “TS 36.423 version 15.6.0 Release 15: LTE; Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 Application Protocol (X2AP),” 2019.

[16] 3GPP, “TS 36.413 version 15.6.0 Release 15: LTE; Evolved Universal Terrestrial Radio Access Network (E-UTRAN); S1 Application Protocol (S1AP),” 2019.

[17] 3GPPP, “ TS 38.425 version 15.3.0 Release 15: 5G; NG-RAN; NR user plane protocol,” 2018.

[18] 3GPP, “TS 38.470 version 15.5.0 Release 15: “5G; NG-RAN; F1 general aspects and principles”,” 2019.

[19] 3GPP, “TS 38.473 version 16.3.1 Release 16: 5G; NG-RAN; F1 Application Protocol (F1AP),” 2020.

[20] Techplayon, “Open Midhaul F1 interface,” [Online]. Available: https://www.techplayon.com/open-midhaul-f1-interface-f1-u-and-f1-c/. [Accessed 14 07 2021].

[21] “O-RAN Alliance,” [Online]. Available: https://www.o-ran.org. [Accessed 09 07 2021].

[22] O-RAN, “O-RAN Working Group 3, Near-Real-time RAN Intelligent Controller (Near-RT RIC) E2 Application Protocol (E2AP) v1.1,” 2020.

[23] O-RAN, “O-RAN Fronthaul Working Group. Control, User and Synchronization Plane Specification,” 2020.

[24] Xilinx, “Xilinx T1 accelerator card specifications,” [Online]. Available: https://www.xilinx.com/publications/product-briefs/xilinx-t1-product-brief.pdf. [Accessed 14 07 2021].

[25] “Kubernetes,” [Online]. Available: https://kubernetes.io/. [Accessed 14 07 2021].

[26] 3GPP, “Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 15), 3GPP TS 23.501 V15.12.0,” 2020.

[27] ETSI, “ETSI GS MEC 003 V2.2.1 - Multi-access Edge Computing (MEC); Framework and Reference Architecture,” 12 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.02.01_60/gs_MEC003v020201p.pdf. [Accessed 09 03 2021].

[28] 5G City Consortium, "5GCity Final Architecture and Interfaces (D2.4)," July 2019. [Online]. Available: https://zenodo.org/record/3751592#.YCvX0mhKiUl.

[29] “5G-CLARITY,” [Online]. Available: https://www.5gclarity.com/. [Accessed 22 03 2021].

[30] “5GVICTORY,” [Online]. Available: https://www.5g-victori-project.eu/. [Accessed 14 07 2021].

[31] O-RAN, “O-RAN Working Group 1, O-RAN Operations and Maintenance Architecture. v03.00,” 2020.

[32] O-RAN, “ORAN Working Group 2: The Non-Real-Time RAN Intelligent Controller (Non-RT RIC) and A1 Interface Workgroup,” 2020.

[33] “O-RAN Software Community,” [Online]. Available: https://wiki.o-ran-sc.org/display/RICNR/Project+Scopes). . [Accessed 14 07 2021].

[34] O-RAN, “O-RAN A1 and O1 workflows,” [Online]. Available: https://wiki.o-ran-sc.org/display/GS/Running+A1+and+O1+Use+Case+Flows. [Accessed 19 07 2021].

[35] 3GPP, “TS 29.520 version 15.3.0 Release 15: 5G; 5G System; Network Data Analytics Services; Stage 3,” 2019.

[36] 3GPP, “TS 23.288 version 16.4.0 Release 16: 5G; Architecture enhancements for 5G System (5GS) to support network data analytics services,” 2020.

[37] 3GPP, “5G; Management and orchestration; Performance assurance. 3GPP TS 28.550 version 15.3.0,” 2020.

[38] 3GPP, “TS 28.532 version 16.6.0 Release 16: 5G; Management and orchestration; Generic management services,” 2021.

[39] 3GPP, “TR 33.866 Release 17: Study on security aspects of enablers for Network Automation (eNA) for the 5G system (5GS) Phase 2,” 2020.

[40] “Acumos AI,” [Online]. Available: https://www.acumos.org. [Accessed 14 07 2021].

[41] “Tensorflow,” [Online]. Available: https://www.tensorflow.org/] . [Accessed 14 07 2021].

[42] “Tensorflow Extended,” [Online]. Available: https://www.tensorflow.org/tfx. [Accessed 14 07 2021].

[43] “Apache Airflow,” [Online]. Available: https://airflow.apache.org. [Accessed 14 07 2021].

[44] “Kubeflow,” [Online]. Available: https://www.kubeflow.org. [Accessed 14 07 2021].

[45] “Airflow vs. Kubeflow,” [Online]. Available: https://www.datarevenue.com/en-blog/airflow-vs-luigi-vs-argo-vs-mlflow-vs-kubeflow. [Accessed 14 07 2021].

[46] “TensorFlow Chicago Taxi AI/ML Model,” [Online]. Available: https://github.com/tensorflow/tfx/tree/master/tfx/examples/chicago_taxi_pipeline. [Accessed 14 07 2021].

[47] “Tensorflow Serving API,” [Online]. Available: REF https://github.com/tensorflow/serving/tree/master/tensorflow_serving/apis].. [Accessed 16 07 2021].

[48] Relyium, “Relyum TSN Setup,” [Online]. Available: https://www.relyum.com/web/rely-tsn-pcie/. [Accessed 19 07 2021].