D2.2: Orchestration, AI techniques, End-to-end slicing and signalling for the core enablers – implementation

Revision: v.3.0

Work package	WP 2
Task	Task 2.1, 2.2, 2.3 and 2.4
Due date	31/10/2024
Submission date	30/10/2024
Deliverable lead	i2CAT
Version	3.0
Authors	Adriana Fernández Fernández (i2CAT), Javier Fernández Hidalgo (i2CAT), Diego San Cristobal (ERI), Rocío Domínguez (ERI), Aurora Ramos, Javier Godás (CGE), Antti Pauanne (UOULU), Sabbir Ahmed (UOULU), Hamid Malik (UOULU), Kenichi Komatsu (UOULU), Jani Pellikka (UOULU), Fernando Pargas Nieto (TID), Rafael Rosales (INTEL)
Reviewers	Jarno Pinola (VTT), Valerio Frascolla (INTEL), Mohammed Al-Rawi (TM Review, IT)
Abstract	This deliverable provides a detailed account of the core network enabler implementations, infrastructure updates, and technical mechanisms essential for the 6G-XR project, setting the foundation for future validation and development of the project's use cases.
Keywords	Extended Reality (XR), Holographic Communications, Digital Twin, Energy Optimization, Edge Computing, Edge Federation, Network Exposure, Network Slicing, IMS Data Channel

Co-funded by the European Union

6GSNS

WWW.6G-XR.EU

Grant Agreement No.: 101096838
Call: HORIZON-JU-SNS-2022

Topic: HORIZON-JU-SNS-2022-STREAM-C-01-01
Type of action: HORIZON-JU-RIA

Document Revision History

Version	Date	Description of change	List of contributors
V0.1	26/04/2024	1st version defining table of contents	Javier Fernandez Hidalgo (i2CATi2CAT)
V1.0	23/09/2024	First draft for internal review	All partners
V2.0	30/09/2024	Document after first review	Valerio Frascolla, Jarno Pinola
V2.1	15/10/2024	Document reviewed by TM	Mohammed Al-Rawi
V3.0	30/10/2024	Final Version Ready for submission	Diego San Cristobal, Javier Fernandez

DISCLAIMER

Co-funded by the European Union

6GSNS

Project funded by
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra
Swiss Confederation
Federal Department of Economic Affairs, Education and Research EAER
State Secretariat for Education, Research and Innovation SERI

6G-XR (*6G eXperimental Research infrastructure to enable next-generation XR services*) project has received funding from the **Smart Networks and Services Joint Undertaking (SNS JU)** under the European Union's **Horizon Europe research and innovation programme** under Grant Agreement No 101096838. This work has received funding from the **Swiss State Secretariat for Education, Research, and Innovation (SERI)**.

COPYRIGHT NOTICE

Project co-funded by the European Commission in the Horizon Europe Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	*Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)*	✓
SEN	*Sensitive, limited under the conditions of the Grant Agreement*	
Classified R-UE/ EU-R	*EU RESTRICTED under the Commission Decision No2015/ 444*	
Classified C-UE/ EU-C	*EU CONFIDENTIAL under the Commission Decision No2015/ 444*	
Classified S-UE/ EU-S	*EU SECRET under the Commission Decision No2015/ 444*	

* R: Document, report (excluding the periodic and final reports)

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

DATA: Data sets, microdata, etc.

DMP: Data management plan

ETHICS: Deliverables related to ethics issues.

SECURITY: Deliverables related to security issues

OTHER: Software, technical diagram, algorithms, models, etc.

EXECUTIVE SUMMARY

This deliverable provides a comprehensive overview of the core network enablers developed to support the 6G-XR project use cases (UC): UC1 *Resolution Adaptation or Quality on Demand*, UC2 *Routing to the Best Edge*, UC3 *Control Plane Optimizations*, UC4 *Collaborative 3D Digital Twin-like Environment*, and UC5 *Energy Measurement Framework for Energy Sustainability*. Each enabler is presented as either a description of the capabilities that it brings to the project, or as a functional description of the enabler followed by a detailed list of the methods available in the exposed Application Programming Interface (API) and concluded with a set of Unified Modeling Language (UML) diagrams that illustrate the internal workflows and interactions, offering the reader a clear understanding of how these enablers operate and how to make use of them.

6G-XR operates through geographically distributed experimentation facilities located in Finland and Spain, which house radio, network, and compute resources (Edge Nodes). The South Node is split between the 5Tonic facility in Madrid, which hosts an Edge system developed by CGE, and i2CAT in Barcelona, which hosts a second federated Edge. In the North Node, resources are co-located at the 5G Test Network (5GTN) in Oulu, Finland.

This deliverable begins by updating the reader on the current infrastructure at each of these sites, covering Madrid, Barcelona, and Oulu, offering insight into the development and capabilities of these resources since their initial description. Then the focus is put on the main project enablers: introducing the Northbound API, a key enabler that allows applications to interact with Edge resources, facilitating seamless resource management, and the enabler for federating Edge systems is explored, enabling operators to extend their network presence by utilizing resources from other operators - an important step for scalable deployments across multiple sites.

Next, enablers related to infrastructure configuration as the Network Exposure Function (NEF) APIs are presented, highlighting their role in network slicing and radio equipment configuration. The description of slicing functionalities is further expanded with details on Cumucore's slice management solution, including APIs for Slice Creation and Management that enable dynamic resource orchestration.

From an energy efficiency perspective, the deliverable introduces an enabler developed for the energy measurement framework, crucial for validating UC5. This open source 5G solution, OAIBOX, integrates a gNodeB (gNB) and 5G core, providing the capability to monitor energy consumption and fine-tune scenarios to assess their impact on sustainability.

Lastly, the deliverable covers enhancements in control plane communications, specifically targeting Extending Reality (XR) scenarios. The Data Channel Server (DCS) network enabler, including the Data Channel Signalling Function (DCSF) and the Media Resource Function (MRF), is introduced to support media forwarding and control logic, ensuring optimal performance for immersive XR applications.

TABLE OF CONTENTS

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

3GPP	3rd Generation Partnership Project
5GTN	5G Test Network
AF	Application Function
AMF	Access and Mobility Management Function
API	Application Programming Interface
B2B	Business-to-Business
CAD	Computer-Aided Design
CBC	Connection Break Count
CDF	Congestion Detection Function
DC	Data Channel
DCSF	Data Channel Signalling Function
DCS	Data Channel Server
DNN	Data Network Name
DL	Download
E2E	End-to-End
ETSI	European Telecommunications Standards Institute
EWBI	East-west bound Interface
gNB	gNodeB
GPSI	Generic Public Subscription Identifier
GSMA	GSM Association
GUI	Graphical User Interface
HO	Holo Orchestrator
HTTP	Hypertext Transfer Protocol
IEAP	Intelligence Edge Application Platform
IMS	IP Multimedia Subsystem
IP	Internet Protocol
K8S	Kubernetes
KPI	Key Performance Indicator
LCM	Lifecycle management
MCS	Modulation and Coding Scheme
MEC	Multi-Access Edge Computing
MEF	MEC Federator
MEO	Mobile Edge Orchestrator
MIMO	Multiple-Input Multiple-Output
MSISDN	Mobile Station International Subscriber Directory Number
MRF	Media Resource Function
NBI	North Bound Interface
NEF	Network Exposure Function
NNA	North Node Adapter
NPN	Non-Public Network
NR	New Radio
NST	Network Slice Template
OPG	GSMA Operator Platform Group
OP	Operator Platform
OSM	Open-Source MANO
PCF	Policy Control Function
PDU	Protocol Data Unit
POC	Proof-of-Concept

PV	Photo Voltaic
QCI	Quality Class Indicator
QoD	Quality on Demand
QoS	Quality of Service
RAN	Radio Access Network
SDP	Session Description Protocol
SMF	Session Management Function
SNA	South Node Adapter
S-NSSAI	Single Network Slice Selection Assistance Information
SUPI	Subscription Permanent Identifier
TAC	Tracking Area Code
TCP	Transmission Control Protocol
TDD	Time-Division Duplex
TI	Traffic Influence
TS	Technical Specification
TSN	Time-Sensitive Networking
UC	Use Case
UDR	Unified Data Repository
UE	User Equipment
UL	Upload
UPF	User Plane Function
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URSP	UE Route Selection Policy
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VNF	Virtualized Network Functions
VNFD	Virtual Network Function Descriptor
VR	Virtual Reality
WP	Work Package
WSGI	Web Server Gateway Interface
XR	Extended Reality

1 INTRODUCTION

1.1 OBJECTIVE OF THE DELIVERABLE

This deliverable presents an overview of the implementation and functionalities of the 6G-XR core network enablers required to validate the use cases defined in D1.1[1] : UC1 *Resolution Adaptation or Quality on Demand*, UC2 *Routing to the Best Edge*, UC3 *Control Plane Optimizations*, UC4 *Collaborative 3D Digital Twin-like Environment*, and UC5 *Energy Measurement Framework for Energy Sustainability*. The focus is on providing a detailed report on the operational mechanisms, architectural design, and workflows of such enablers. It also summarizes the current status and describes the final implementations, highlighting modifications and improvements made since the initial versions described in D2.1 [2] were published. Any updates related to the test facilities are also included to highlight changes that might impact on the experimentation environments. This deliverable also aims to equip stakeholders with a clear understanding of the technical aspects of the enablers and their roles within the overall system architecture.

The next deliverable (D2.3) will provide a comprehensive view of the integrated system, detailing its overall functionality and performance.

1.2 STRUCTURE OF THE DELIVERABLE

As described in D2.1, the 6G-XR experimentation facilities are distributed across two geographically separated nodes. The South Node, located in Spain, consisting of two sites: Madrid and Barcelona. The North Node is based in Finland. Chapters 2, 3 and 4 describe the capabilities of these nodes and highlight any updates or upgrades that have occurred since D2.1 was published.

The subsequent sections cover the individual enablers implemented in Work Package 2 (WP2). These include 5GTN Resource Orchestration APIs (Chapter 5), Edge Northbound Interface (NBI) Application Programming Interface (API) (Chapter 6), Edge Federation (Chapter7), Network Exposure Function (NEF) APIs for slicing (Chapter 8), Cumucore Slice Management (Chapter 9), Open Source 5G Solution for the sustainability framework (Chapter 10), and Control Plane Innovations (Chapter 11). For each enabler, a functional description is provided, followed by a detailed explanation of the implemented APIs. Unified Modeling Language (UML) diagrams are included to illustrate the main workflows and operational processes.

1.3 TARGET AUDIENCE OF THE DELIVERABLE

This deliverable is intended for project consortium partners, academic and research institutions, EU Commission services, and other stakeholders with a technical background in wireless networks, in 5G and 6G technologies, network orchestration, and Extended reality (XR) applications, particularly in the context of system and resource optimization.

2 MADRID EDGE CAPABILITIES

The infrastructure in Madrid, hosted at 5Tonic Laboratory comprises computing resources (Madrid Edge) and networking equipment (details on the overall infrastructure can be found in Deliverable D1.3 [3].

As planned in D2.1 [2], the edge infrastructure at the 5Tonic Madrid lab has been extended in the last months to cope with 6G-XR requirements by means of the installation of two new servers with the following characteristics collected in Table 1:

Table 1: Hardware Specifications of new Servers at 5Tonic

	Tower 1	Tower 2
Processor	Intel Xeon Silver 4310 2.1G, 12C/24T, 10.4GT/s, 18M Cache, Turbo, HT (120W) DDR4-2666	Intel Xeon Silver 4310 2.1G, 12C/24T, 10.4GT/s, 18M Cache, Turbo, HT (120W) DDR4-2666
Memory	16GB	16GB
GPU	NVIDIA(R) Tesla(TM) T4 Single Slot, Full Height GPU	NVIDIA(R) Tesla(TM) T4 Single Slot, Full Height GPU

The two new servers plus the previous ones (details can be found in Deliverable D2.1 [2]) have been used to set the whole installation and configuration of Capgemini's Multi-Access Edge Computing (MEC) orchestrator (*IEAP – Intelligence Edge Application Platform*), and multiple nodes of edge infrastructure at 5Tonic. As it is represented in Figure 1, an Openstack setup has been used to host IEAP, and a Kubernetes cluster provides two different edge locations. Figure 1 also includes the virtualization characteristics for each of the Edge nodes, e.g. single root I/O virtualization (SRIOV), Graphics Processing Unit (GPU) or CEPH as storage platform.

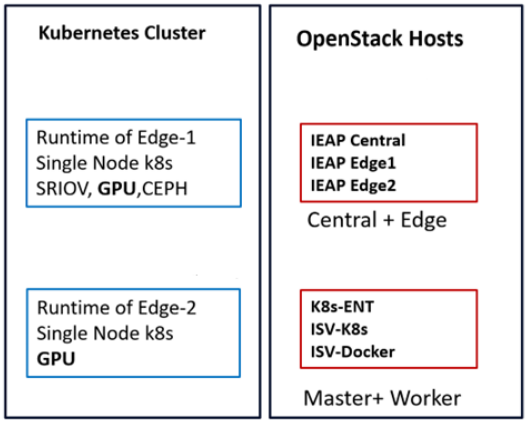

Figure 1: Virtualized infrastructure setup in 5Tonic (Madrid)

Figure 2 represents the connections between the IEAP and the two Edge Sites at 5Tonic, as well as the integration planned with the 5G Network Core components and the federation with Barcelona MEC orchestrator.

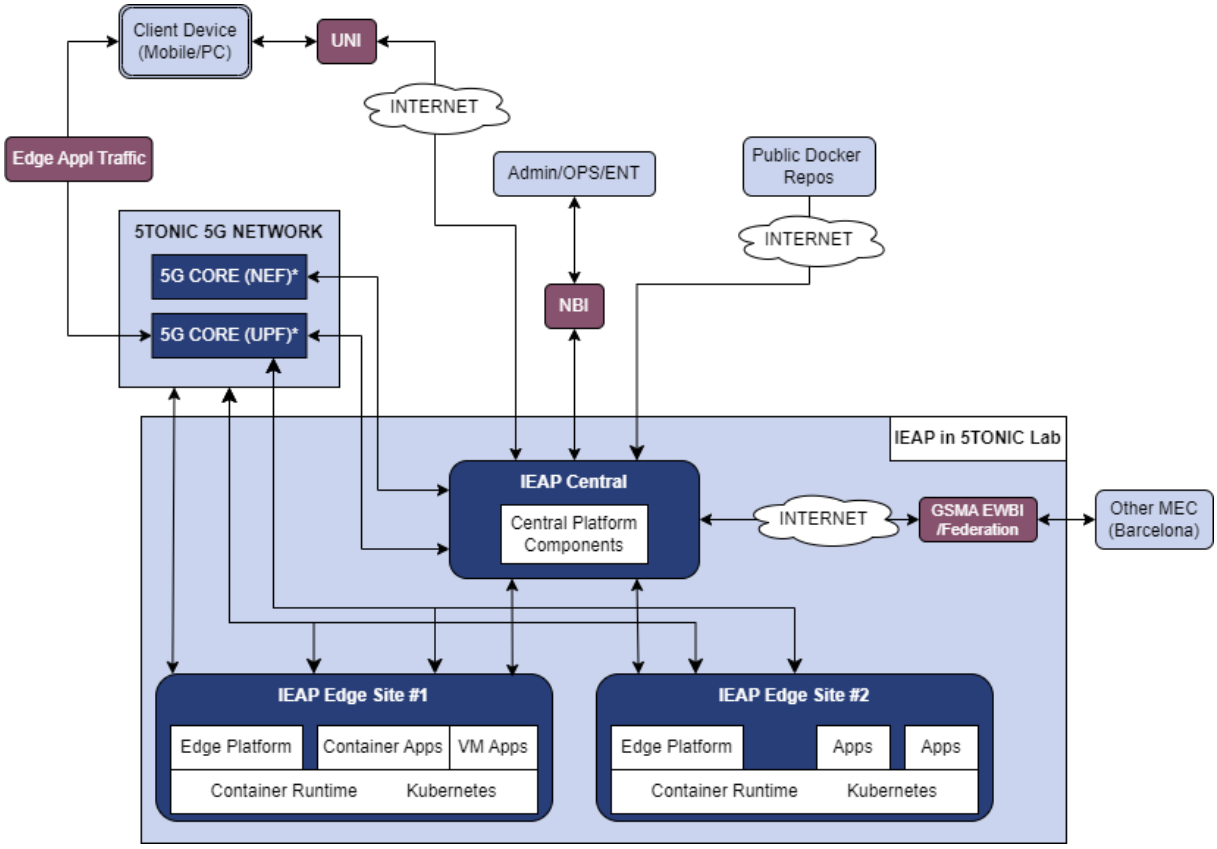

Figure 2: IEAP and edge infrastructure configuration at 5Tonic (Madrid)

The amount of compute and storage resources for each 5Tonic edge infrastructure node is detailed in Table 2.

Table 2: Compute and storage resources at Edge infrastructure nodes at 5Tonic

VM/BareMetal	Resources
IEAP Central	vCPU-5, RAM-8, Disk-500GB
Edge -1	vCPU-5, RAM-8, Disk-500GB
Edge - 2	vCPU-5, RAM-8, Disk-500GB
Bootstrap	vCPU-5, RAM-8, Disk-500GB
K8s Cluster -ENT-1	vCPU-6, RAM-20, Disk-400GB
ISV – Dedicated K8s Cluster	vCPU-6, RAM-20, Disk-400GB
ISV – Docker Node	vCPU-6, RAM-20, Disk-400GB
Single node K8s Cluster – SRIOV, GPU, CEPH	CPU-20, RAM-128, Disk-1.2TB
Single node K8s Cluster –GPU	CPU-20, RAM-128, Disk-1.2TB

The functional gaps at 5Tonic identified in D2.1 [2] are further detailed as follows:

- The NBI APIs implementation details are covered in Chapter 5.
- The East-west bound Interface (EWBI) to implement federation with other MEC orchestrators, e.g., 5GBarcelona, is covered in Chapter 7.
- Details on integration with the 5G core will be covered in future deliverable D2.3 due in May 2025.

3 BARCELONA EDGE CAPABILITIES

This chapter will focus on the infrastructure at the Barcelona site, which is part of the South Node. The Barcelona infrastructure, hosted by i2CAT, includes computing, network, and radio resources.

Following the submission of D2.1 [2], there have been minor additions to the lab setup in Barcelona. Notably, a small mobile rack provided by Ericsson has been integrated. This rack functions as a User Plane Function (UPF), connecting the core in 5Tonic in Madrid to the radio equipment in Barcelona.

Further experiments with the Graphics Processing Unit (GPU) have been conducted after the preparation of deliverable D2.1. The final deployment of the Virtual Infrastructure has been completed using an OpenStack setup, which hosts a Kubernetes cluster. It has now been validated that, by using the Kubernetes GPU operator, the GPU can be shared across multiple pods. This was one of the main uncertainties at the time of writing D2.1 , but it has since been resolved.

Regarding federation, the team has implemented the MEC Federator (MEF) Manager, which facilitates interactions with other federation entities via the east-west bound interface. Additionally, it communicates directly with the MEC platform manager through its north-bound interface to manage resources. While the development is complete, integration with the Edge node in Madrid is still pending. The results of this integration will be documented in D2.3.

In terms of edge computing, new personal computer (PC) towers equipped with GPUs have been integrated into the setup, forming a new cluster. This enhancement facilitates the deployment of more complex scenarios and services. These towers will be configured as a Virtual Infrastructure Manager (VIM) for the edge orchestrator. Specifically, they will constitute a Kubernetes (k8s) cluster, managed and orchestrated by the Edge orchestrator (i2EDGE).

The following table [Table 3] outlines the most relevant HW specifications of the PC towers.

Table 3: Hardware specifications for new VIM

	Tower1	Tower2
Processor	AMD Ryzen threadripper 3970X 32-Cores 3.70Ghz	AMD Ryzen threadripper 3970X 32-Cores 3.70Ghz
Memory	16GB	16GB
GPU	NVIDIA GeForce RTX 3060Ti	NVIDIA GeForce RTX 3060Ti

4 5GTN (UOULU AND VTT) EDGE CAPABILITIES

This chapter describes the WP2 enablers within the North Node of 6G-XR project use cases, 3D Digital Twin and Energy Management. A 3D digital twin enables collaborative work by integrating cyber-physical systems within a 3D environment. The edge computing gaming engine server needs to synchronize human and machine interactions using 5G radio and TSN (Time-Sensitive Networking) functionality to ensure seamless communication and real-time performance. Energy Management is focusing on the sustainability experimentation framework. The 5G Test Network (5GTN) facility integrates the energy measurement framework for the North Node E2E datapath modules. This section highlights the key 5GTN capabilities of energy measurement as an enabler.

4.1 3D DIGITAL TWIN

The system architecture is defined including 3D Digital Twin functions and network connections between remote location, Fab Lab room, network components, management and orchestration and edge services. A 3D game engine includes XR functionality within the Virtual Reality (VR) environment, a user interface for the web server, and establishes data streaming sessions, as depicted on Figure 3. The network components drive connectivity within the 5GTN of the North Node, including a 3GPP-based core, Radio Access Network (RAN) and Wi-Fi. The remote location setup consists of VR glasses and a PC, which facilitates the upload of stereolithography (STL) files (a file format commonly used for 3D printing and computer-aided design (CAD)) and establishes a 5G connection with a 5G modem. In the Fab Lab room, in addition to VR glasses and a PC, a 3D printer and an arm robot are set up to enable remote fabrication machine control with TSN functionality.

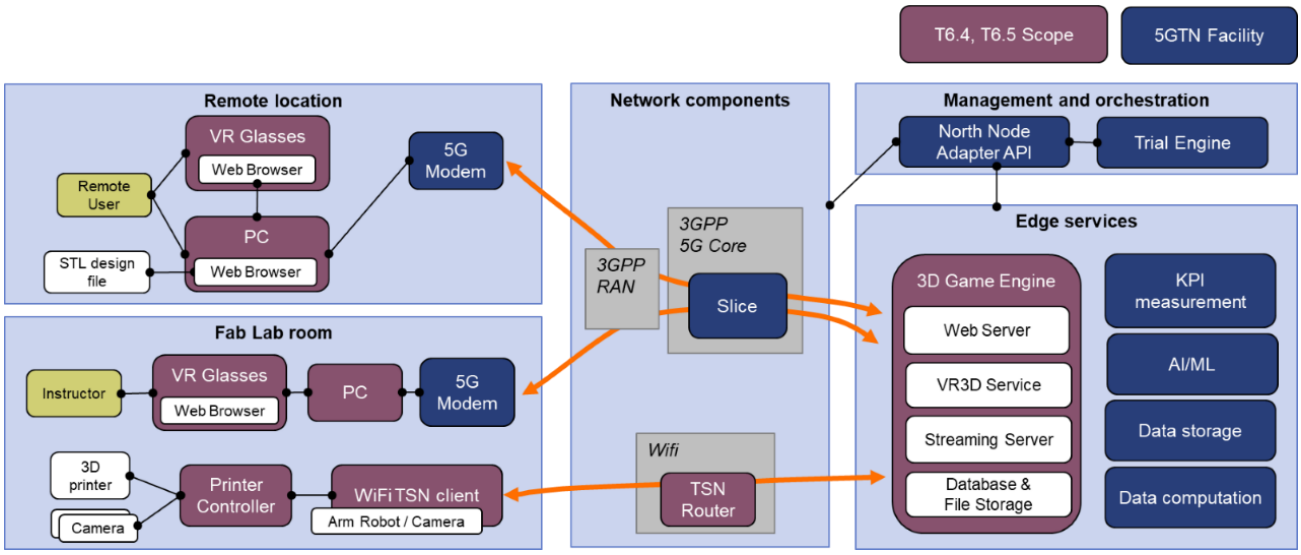

Figure 3: 3D Digital Twin system architecture

We established a comprehensive Fab Lab environment by integrating together XR devices and a 3D printer. We developed two scenarios employing XR functionalities to enable collaborative remote work effectively:

1. VR users, Remote Instructor and Remote User, remotely review the digital twin of a 3D object in the VR scene before printing.

2. VR users control the remote-controlled 3D printer and observe the 3D printing process via video streams in the VR scene.

The first scenario involves conducting a remote review in a VR scene. As depicted in Figure 4, this process begins with user registration. The user distinguishes between Remote User or Remote Instructor, facilitating the creation of personalized avatars within the VR environment. Subsequently, Remote User uploads STL files, as shown on Figure 5, to introduce 3D models into the VR scene. STL). In this use case it is commonly used for the review in the VR scene and production by 3D printer. At the start menu of the application, users can register and create their avatars. Using VR glasses, the system synchronizes the user's movements with the avatar, providing an immersive and seamless experience. Additionally, audio streaming sessions allow users to communicate verbally within the VR scene. For reviewing a 3D model, user Avatars and uploaded 3D models are spawned in the VR scene. The VR hand-tracking feature of the VR glasses allows users to replicate their hand motions, enabling them to manipulate and review the uploaded 3D objects directly within the VR environment.

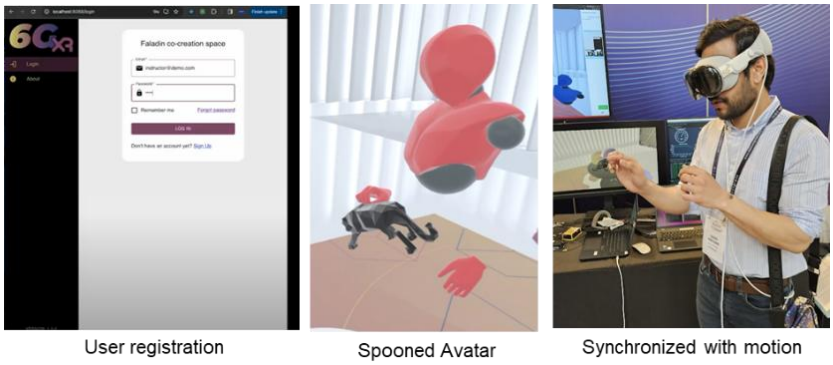

Figure 4: User registration and integration in the VR scene

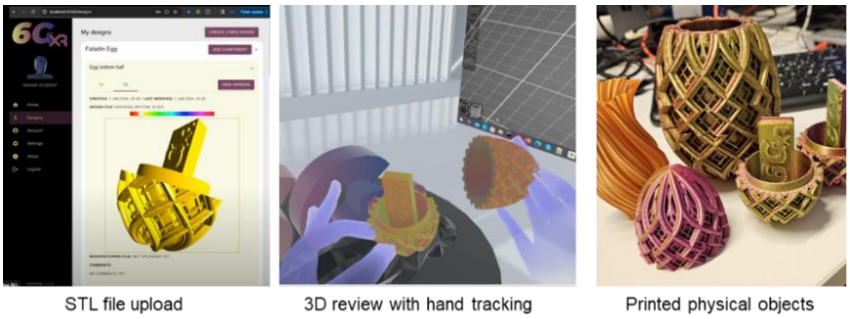

Figure 5: 3D print object upload and VR review

Another scenario is real-time surveillance on the 3D printer of Fab Lab in a VR scene. As shown in Figure 6, two cameras are set up with Raspberry Pi, located at the side panel and top cover. OctoPrint (an open-source 3D printer controller) is installed on one Raspberry Pi, allowing the 3D printer to be controlled from the VR scene via backend services. In the VR environment, users have the capability to remotely control a 3D printer and observe the production process. This is achieved through real-time video streaming from both the top cover view and the side panel view, which are seamlessly integrated into the VR scene with accurate 3D orientations as shown in Figure 6.

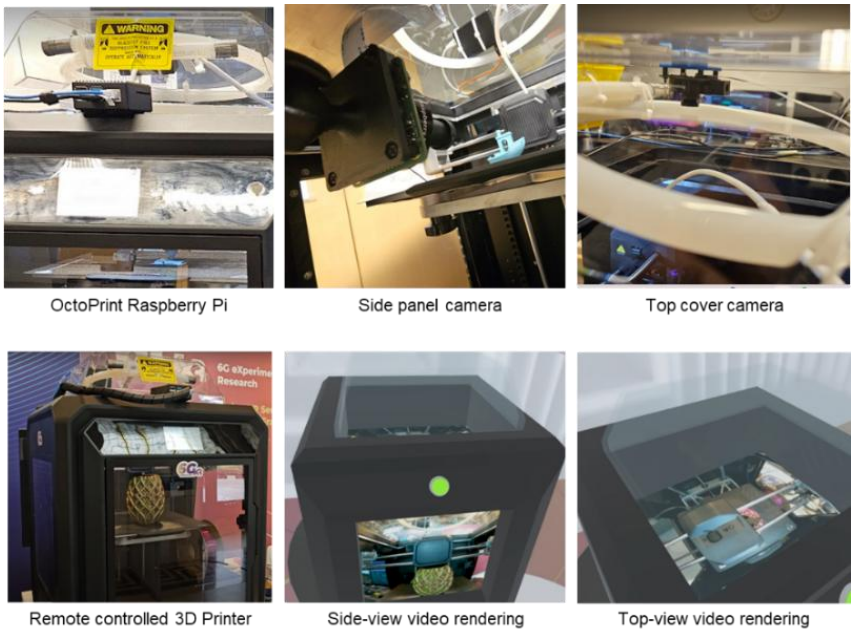

Figure 6: 3D printer set-up

4.2 ENERGY MEASUREMENT

The North Node system architecture integrates energy measurement within the 5GTN, utilizing updated assets from UOULU and VTT. The sustainability experimentation framework, as defined in D5.1 [4], includes a 5G network powered by renewable energy sources, with Photo Voltaic (PV) modules at the North Node gNodeB (gNB) sites, on-site sensors, and solar charge controllers. The deployed energy measurement framework enables real-time energy monitoring of on-site sensors and the entire End-to-End (E2E) system. It also includes a bidirectional controllable grid interface, electricity storage, and uninterruptible power distribution systems, which incorporate metering and protection using external power meters and devices as defined in D5.1 [4]. The E2E scope also covers open source 5G (OAIBOX) and Universal Software Radio Peripheral (USRP) (n310/b210), integrated within the North Node energy measurement framework. Real-time energy monitoring is enabled using external power meters (Netio PowerBox [5]), as shown in Figure 7.

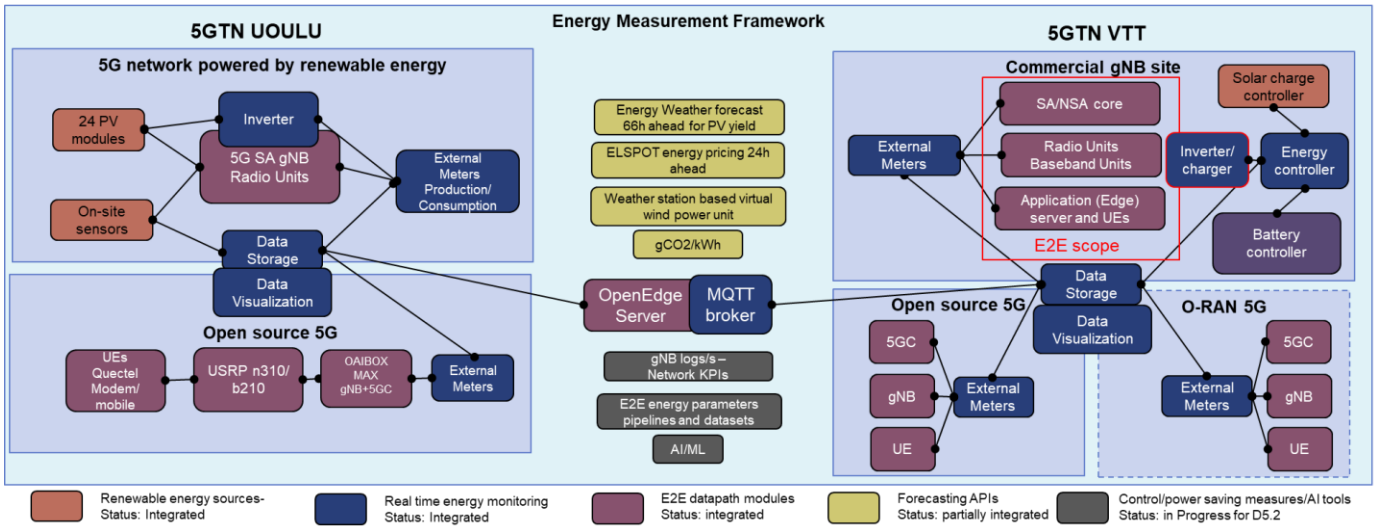

Figure 7: North Node high level architecture

D5.1 includes ongoing data collection of real-time energy monitoring, covering PV yield and energy consumption of various components within the E2E data path, as shown in Figure 7. These datasets lay the foundation for control and power-saving measures for experimentation research, which will be reported in D5.2. Key developments thus far from the perspective of energy management as an enabler and E2E energy efficiency include:

- Almost real-time electricity metering system covers the whole E2E-system.
- Integration of central controller (OpenEdge server) with the energy measurement framework to enable bridge connection between north node sites for data exchange, control and power saving measures.
- Integration of forecasting APIs such as energy weather forecast, Absolute gCO_2/procured kWh's and on-site sensors within energy measurement framework.
- System is integrated to external data-services (forecasting APIs) producing 66 hour ahead PV-yield forecast(s), indirect CO_2 emission estimates to grid intake and day ahead hourly price of electricity to assist control decisions.
- Monitoring and control systems are available to manage energy and power balance and visualize impacts of initiated changes.
- Impacts of control actions (e.g., triggered changes or developed control patterns) to overall power and energy consumption can be visualised by means of Grafana panels.
- Setup of Mosquito Message Queues Telemetry Transport (MQTT) broker for pub/sub, data exchange and control commands and enabling bridge connection within North Node.

5 5GTN RESOURCE ORCHESTRATION APIS

This chapter details the WP2 enablers available within the North Node 5G Test Network (5GTN) facility, focusing on the North Node Adapter (NNA). The NNA facilitates the setup and management of network slices and KPI (Key Performance Indicator) measurements, serving as a crucial link between the North Node Web Portal and core 5GTN components such as Open-Source MANO (OSM), Cumucore, etc. This section highlights the key 5GTN capabilities, including AI-based network resource optimization, which are essential enablers for WP2 within the 5GTN research infrastructure.

5.1 NORTH NODE ADAPTER

The NNA will be implemented in T4.4 “Deployment of trial controller, APIs and novel orchestration”. The 5GTN Facility Adapter that was created in the 5G!Drones project [6] was supposed to be used as a starting point for the implementation of the NNA. However, in T4.4 a much simpler design was devised which differs from that of the 5GTN Facility Adapter. The updated design and architecture of NNA are presented in this section.

5.1.1 Functional Description

The NNA is a crucial REST API that plays a vital role in the setup and management of network slices and KPI measurement jobs within the 5GTN. It acts as a communication bridge between the North Node Web Portal, the underlying 5GTN resources and management technologies, such as the OSM to create applications and Cumucore to manage network slice instances demanded by a human user of the portal. The NNA has an interface to interact with Qosium Storage for KPI collection and Artificial Intelligence / Machine Learning (AI/ML) component responsible for optimizing the slice parameters (e.g., throughput and latency) on-the-fly based on the collected KPIs. Figure 8 illustrates the North Node architecture and the role of the NNA in it.

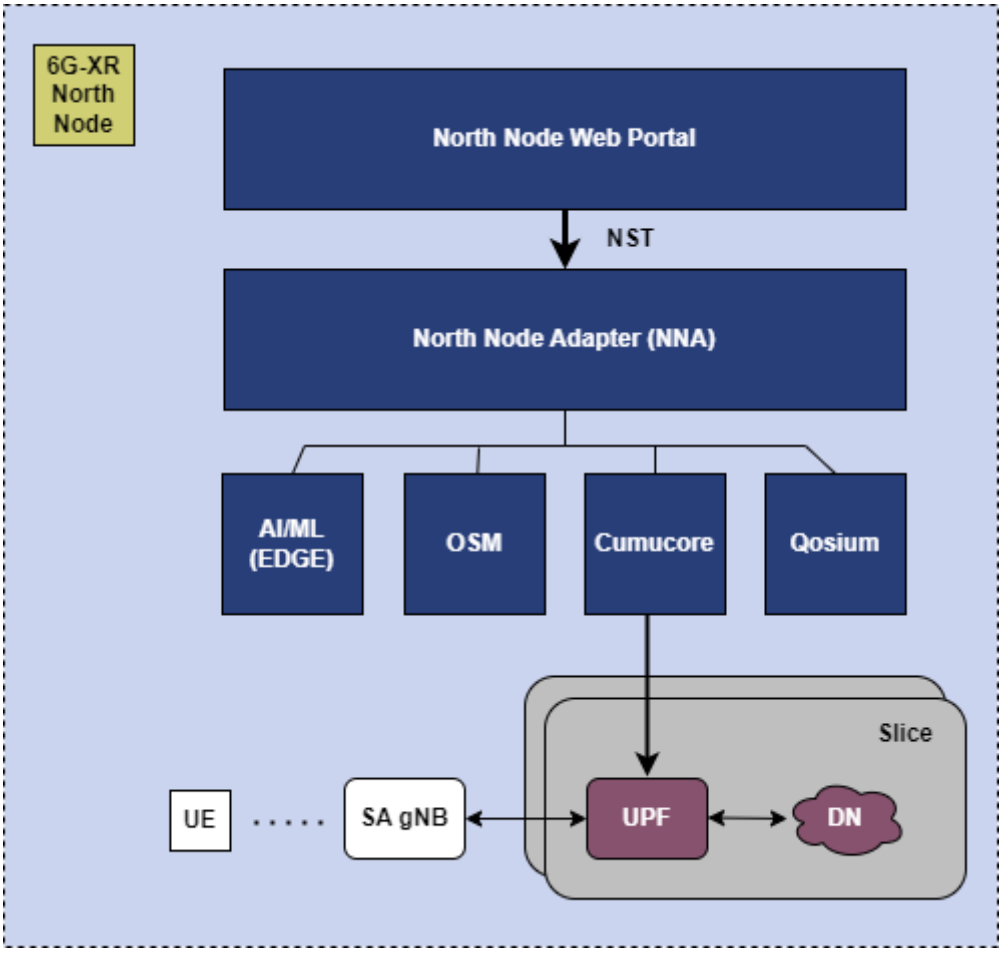

Figure 8: North Node architecture

The NNA is responsible for managing the lifecycle of network slices in the 5GTN facility. From the North Node web portal, the NNA receives Network Slice Template (NST), a JSON formatted document that specifies the maximum of two types of two slices used in an experiment. Allowed slice types are “eMMB” and “uRLLC”. Upon the receipt of an NST, the NNA configures slices to be used in the experiment by using the Cumucore REST API. During the experiment, the NNA also continuously reconfigures Cumucore core network parameters based on the suggestions from the AI/ML component responsible for determining the optimal throughput and latency for the network slices used in the experiment.

The NNA oversees the lifecycle of KPI measurement jobs. The NNA uses Qosium Storage for measuring the KPIs per slice. The collected KPI values (jitter, latency, throughput, and packet loss ratio) are received from Qosium in one second intervals using the REST API provided by Qosium Storage server and forwarded to the AI/ML component for request of optimizing the latency and throughput of the slices. In addition to receive real-time KPI information of a slice, Qosium provides a REST API for starting and stopping measurements. Upon the receipt of an NST, the NNA creates and starts a KPI measurement per slice using this API. At the end of the experiment, the API is used to stop and tear down all the per slice measurements.

The NNA also communicates with the OSM, which is responsible for the entire lifecycle management of Virtualized Network Functions (VNFs) including Virtual Machine (VM) provisioning, application

deployment, and VNF service management. The OpenStack VIM is responsible for provisioning and managing infrastructure resources like VMs and storage.

5.1.2 API Methods

The NNA provides a REST API, which is used by the North Node web portal. Through the REST API the portal can start an experiment, get status of one specific or all running experiments, and stop/delete the experiment when finished. The stop/delete method of the API returns a log file stating the actions (such as slice reconfigurations in Cumucore according to the suggestions from the AI/ML component) taken during the experiment. The create/start method takes NST as a parameter. The NNA REST API is defined as follows:

- <REST API basic URL>/experiment
  - Use: Create and start a new experiment based on NST
  - Method: POST
  - Request params: none
  - Request content: x-form-url-encoded params
    - <nst> – NST in JSON format
  - Possible responses:
    - 200 – OK: <id>, <sliceId1>, <sliceId2>
    - 400 – Error in NST values

- <REST API basic URL>/experiment
  - Use: Get the status of all running experiments
  - Method: GET
  - Request params: none
  - Request content: none
  - Possible responses:
    - 200 – OK: List of all experiments with <id>, <state>, <error message>

- <REST API basic URL>/experiment/<id>
  - Use: Get the status of a specific experiment
  - Method: GET
  - Request params:
    - <id> – The ID generated for the experiment by NNA
  - Request content: none
  - Possible responses:
    - 200 – OK: <id>, <state>, <error message>
    - 404 – Submitted experiment ID not found

- <REST API basic URL>/experiment/<id>
  - Use: Stop and remove experiment
  - Method: DELETE
  - Request params:
    - <id> – The ID generated for the experiment by NNA
  - Request content: none
  - Possible responses:
    - 200 – OK: <log data> (Content-Type: text/plain; charset: utf-8)
    - 404 – Submitted experiment ID not found

The NNA uses a REST API [7] provided by the Qosium component to start and stop measurements, and to get real-time measurement results per slice for the AI component to analyse. The Qosium REST API methods used by the NNA are described in Table 4.

Table 4: Qosium REST API methods

Method	Path	Description
POST	{apiRoot}/measurement/start	Start measurement. Example: curl -k -X POST https://192.168.9.120:8080 /measurement/start -d ' {"measurement_description":"","robust_mode_max_cbd":"5","user_id":"10","primary_probe_hostname":"172.29.6.17","primary_probe_port":"8177","use_secondary_probe":"true","secondary_probe_hostname":"172.29.6.18","secondary_probe_port":"8177","primary_probe_placement":"10","primary_probe_interface_index":"0","secondary_probe_interface_index":"0","secondary_probe_placement":"10","nat_between_probes":"false","packet_filter_mode":"240","packet_filter":"host 172.29.6.17 and host 172.29.6.18","primary_probe_senders_pure_mac_method_enabled":"false","primary_probe_senders_eth_mode":"0","primary_probe_senders_eth_address_list":"","primary_probe_senders_ipv4_mode":"249","primary_probe_senders_ipv4_address_list":"","primary_probe_senders_ipv6_mode":"0","primary_probe_senders_ipv6_address_list":"","secondary_probe_senders_pure_mac_method_enabled":"false","secondary_probe_senders_eth_mode":"0","secondary_probe_senders_eth_address_list":"","secondary_probe_senders_ipv4_mode":"249","secondary_probe_senders_ipv4_address_list":"","secondary_probe_senders_ipv6_mode":"0","secondary_probe_senders_ipv6_address_list":"" }'
GET	{apiRoot}/measurement/stop ?QSMeasId={:qsMeasId}	Stop measurement. Example: curl -k -X GET https://192.168.9.120:8080 /measurement/stop?QSMeasId=1
GET	{apiRoot}/AverageResult?qmId={:qmId}&limit={:numResultslimit}&sort={:direction}&after={:aftertime}&before={:beforeTime}&page={:page}	Get measurement results per measurement ID. Example: curl -k -X GET https://192.168.9.120:8080 /AverageResult?qmId=1&limit=10&sort=desc&after=1728358529&before=1728362129&page=2

The Cumucore slice management API and the network resource optimization (AI/ML) API are described in their respective sub-sections below in this document.

5.1.3 Workflows

During the execution of an experiment, the NNA communicates with several sub-components, namely Cumucore, AI/ML, and Qosium. This interaction with the sub-components is illustrated in Figure 9.

When the experiment is being run, the NNA queries the recent, real-time per slice QoS measurement data from Qosium. This data consists of throughput, jitter, delay (latency), and packet loss ratio. The obtained data is then forwarded to the AI/ML component, which utilizes a novel algorithm to optimize the slice bandwidth allocation. The AI/ML component returns the optimal slice allocation to the NNA, which in turn adjusts the slice parameters accordingly by using the Cumucore slice management API (described in detail in Chapter 8). This sequence continues until the experiment is finished.

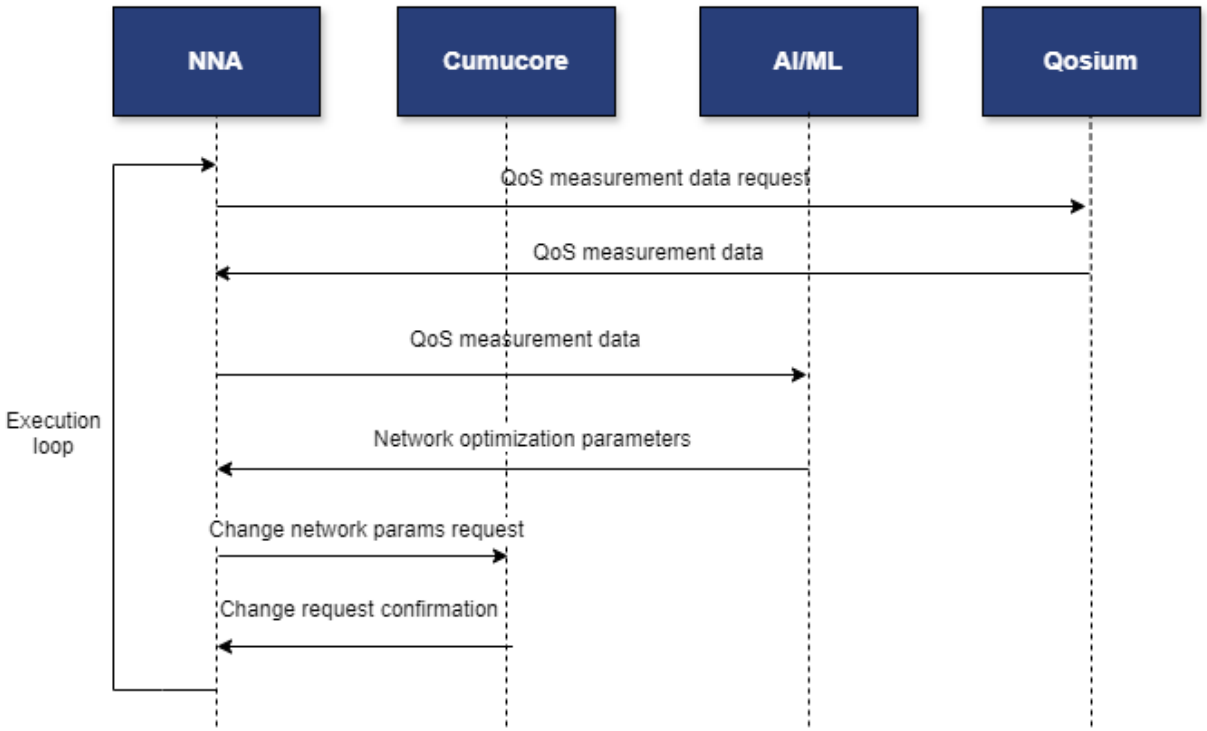

Figure 9: NNA interaction with sub-components

Before an experiment is run, the orchestrator NNA communicates with OSM through its Northbound APIs and triggers a request to create a VNF. Then, OSM looks for the required Virtual Network Function Descriptor (VNFD) to determine what kind of resources are needed for the VNF. After that, the OSM sends a request to OpenStack to create the VMs to support the VNF, OpenStack checks the request and deploys the necessary VMs. OpenStack responds to OSM with details like the VM ID and Internet Protocol (IP) address and then OSM saves the information about the created VMs. The OSM processes the deployment of the necessary applications on the newly created VMs and notifies the NNA that the required VMs and applications are ready.

Thus, OSM continuously monitors and updates the status of the VMs and applications during their lifecycle. When the VMs are no longer needed (i.e. the experiment has ended), NNA asks OSM to delete them. Then OSM sends a request to OpenStack to delete the VMs and OpenStack confirms the deletion. The flow diagram among NNA, OSM and OpenStack is given in Figure 10.

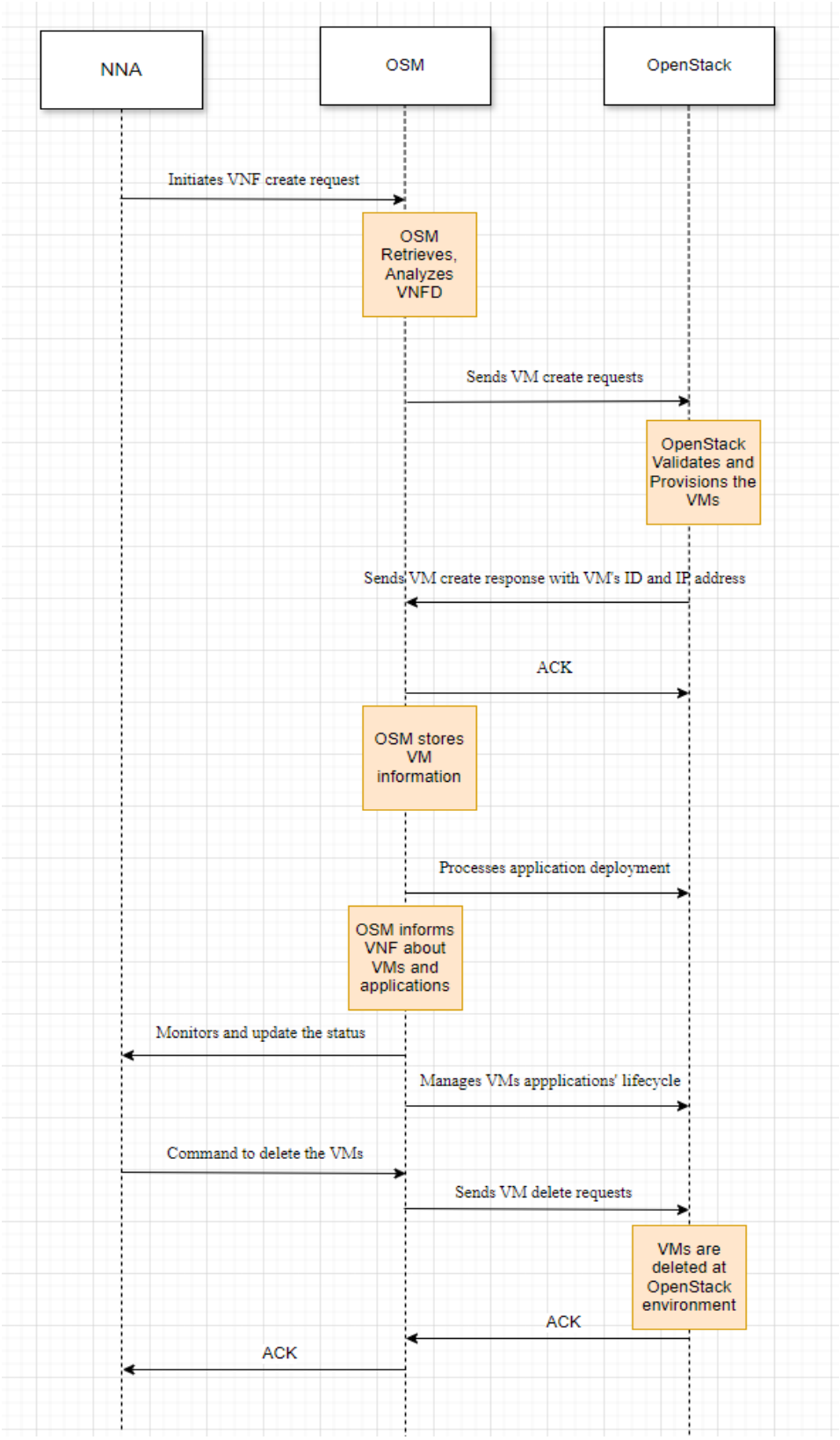

Figure 10: The flow diagram of the NNA, OSM and OpenStack

5.2 RESOURCE OPTIMIZATION

The resource optimization solution is designed to optimize resource allocation in an E2E network slicing environment within the 5GTN facility. This system uses AI to dynamically manage and allocate resources across multiple network slices, ensuring optimal performance and efficiency under varying network conditions.

In the context of 5G and beyond, network slicing allows for the creation of multiple virtual networks on top of a shared physical infrastructure. Each slice can be tailored to meet specific service requirements, such as high throughput for video streaming or low latency for mission-critical applications. The 5GTN facility provides a cutting-edge testbed for implementing and evaluating such advanced networking concepts.

5.2.1 Functional Description

Our AI-Assisted Network Slicing system aims to address the complex challenge of real-time resource optimization across these diverse slices. By utilizing reinforcement learning techniques and real-time network data, the system can adapt to changing conditions and service demands, maximizing overall network performance while meeting the unique requirements of each slice.

The primary objectives of the system are to:

- Maximize throughput across all slices.
- Minimize performance penalties such as latency, jitter, and packet loss.
- Ensure Quality of Service (QoS) requirements are met for each slice.
- Adapt to dynamic network conditions in real-time.

To achieve these goals, the system integrates various components, including real-time monitoring tools, an AI-driven decision-making engine, and a flexible network slicing environment. This section provides the improved design and functionality of this system, detailing its components, design goals, and operational workflow.

System components:

1. Qosium [8] (an underlying component of NNA):
    - Role: Qosium serves as the primary data source for network performance indicators in the 5GTN E2E environment. It continuously monitors the network, providing real-time metrics for each slice.
    - QoS Metrics: KPIs include:
        - Throughput (Mbps): Measures the amount of data successfully transmitted per unit time.
        - Latency (ms): Quantifies the time taken for data to travel from source to destination.
        - Jitter (ms): Represents the variability in packet arrival times.
        - Packet Loss (%): Indicates the percentage of packets lost during transmission.

- Integration: Qosium is tightly integrated with the 5GTN infrastructure, allowing for comprehensive monitoring across the E2E network slicing environment.

2. E2E Network Slicing Environment:
    - Structure: Leverages the 5GTN facility to create and manage multiple network slices across the entire network path, from radio access to the core network.
    - Dynamic States: The environment includes real-time signals that represent the current state of the network:
    - Slice Management: Allows for the creation, modification, and deletion of slices based on service requirements and AI system decisions.
3. Policy Network (AI Agent):
    - Core Technology: Utilizes a state-of-the-art reinforcement learning model REINFORCE, a Monte Carlo variant of a policy gradient algorithm.
    - Decision Making: Employs a Dirichlet distribution to sample optimal resource allocation between slices. The REINFORCE algorithm trains a neural network to predict the alpha parameter of the Dirichlet distribution. All samples from this distribution correspond to a full split of available resources among N contestants.
    - Input Processing: Receives dynamic state signals from the E2E environment and Qosium metrics as input.
    - Learning Mechanism: Continuously learns through interactions with the environment, optimizing its decision-making process over time. As the gradient of the real system operation is not available, the chosen approach makes use of the E2E reward in order to reinforce good policies.
    - Adaptation: Capable of adjusting to the unique characteristics and demands of the 5GTN E2E slicing environment.
4. Reward Function:
    - Objective: Drives the AI agent to maximize overall network efficiency by balancing throughput maximization and penalty minimization.
    - Mathematical Formulation:

      $$\text{Reward} = (\text{Allocation}_{\text{slice1}} \times \text{Throughput}_{\text{slice1}}) + (\text{Allocation}_{\text{slice2}} \times \text{Throughput}_{\text{slice2}}) - (\text{Latency}_{\text{slice1}} + \text{Latency}_{\text{slice2}}) - (\text{Jitter}_{\text{slice1}} + \text{Jitter}_{\text{slice2}}) - (\text{PacketLoss}_{\text{slice1}} + \text{PacketLoss}_{\text{slice2}})$$

    - Customization: The reward function can be fine-tuned to prioritize specific performance aspects based on 5GTN research objectives or test scenarios.

Design goals:

1. Optimize Resource Allocation: The core goal of the system is to allocate resources dynamically between network slices based on current network conditions, as observed by Qosium. This ensures that each slice receives enough resources to meet its QoS requirements while minimizing performance degradation.

2. Adaptive Learning: The AI agent adapts over time by learning the best resource allocation strategy based on rewards. The use of a reinforcement learning approach allows the system to improve over time, becoming more efficient with every episode.

3. Dynamic State Representation: The system now uses dynamically changing states as inputs, which provide real-time feedback about the health of the network. The agent must adjust its resource allocations based on real-time observations such as signal strength, connection breaks, and packet delays.

4. Validation and Predictability: The adaptation of the reinforcement learning algorithm to the network slicing problem has been validated through a predictable network simulator to evaluate the implementation correctness. After this step, the algorithm can be integrated with the real system for further experimentation/training. The design ensures predictability in how the AI agent responds to different network conditions, which is key for ensuring reliable performance.

5.2.2 API Methods

The AI-assisted network slice management algorithm is designed to dynamically allocate resources to different network slices within a 5G or beyond 5G network. This allocation is based on real-time Key Performance Indicators (KPIs) and network state data. The primary goal is to optimize the performance of each slice in terms of throughput, latency, jitter, and packet loss while minimizing penalties associated with these KPIs. The algorithm interacts with the network environment, which can receive real-time data through API integrations like Flask

- <REST API basic URL>/update _status
  - Use: Send per slice network QoS parameters to AI/ML
  - Method: POST
  - Request params: none
  - Request content: Content-Type: application/json
    - [
      {“jitter”: 12.23, “throughput”: 12.56, “delay”: 0.45, “packetLoss”: 0.03},
      {“jitter”: 12.23, “throughput”: 12.56, “delay”: 0.45, “packetLoss”: 0.03}
      ]
  - Possible responses:
    - 200 – OK
- <REST API basic URL>/resource_allocation
  - Use: Get info how bandwidth should be allocated among slices
  - Method: GET
  - Request params: none
  - Request content: none
  - Possible responses:
    - 200 – OK: {“slice-1”: 0.123, “slice-2”: 0.877}

5.2.3 Workflows

System workflow:

The workflow of the AI-based resource optimization is presented in Figure 11.

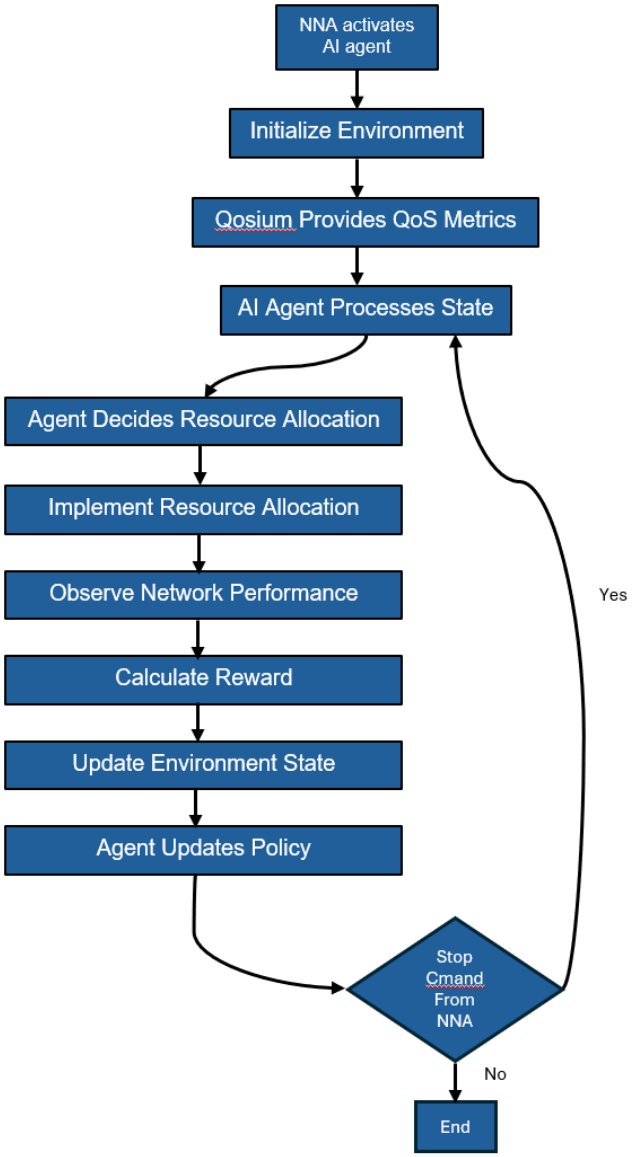

Figure 11: The workflow of the AI-based resource optimization

The workflow progresses as follows:

1. **NNA Activates AI Agent**
   NNA, also called Network Management System, activates the AI agent. This represents the point at which the network identifies the need for optimization or resource reallocation and engages the AI agent to make decisions based on current network conditions. The AI agent is the reinforcement learning model designed to optimize network performance.

2. **Initialize Environment**

At this stage, the environment in which the AI agent will operate is initialized. This environment includes network parameters, KPIs, and network state information (e.g., throughput, latency, jitter, signal strength). The environment will simulate or interact with the live network, allowing the AI agent to assess the current state of the network.

3. **Qosium Provides QoS Metrics**
   Qosium is a tool or system that provides metrics, which are performance indicators such as Throughput, Latency, Jitter, and Packet loss. These QoS metrics are critical for the AI agent to understand the performance of different network slices or services.

4. **AI Agent Processes State**
   The AI agent processes the network state by taking into account both the QoS metrics provided by Qosium and the general network environment (signal strength, packet counts, etc.). Based on this input, the AI agent assesses the overall state of the network and prepares to decide on resource allocation.

5. **Agent Decides Resource Allocation**
   The AI agent, based on the processed network state, decides how to allocate resources to different network slices or services. The resource allocation decision aims to optimize network performance based on the QoS metrics. For example, the agent may allocate more bandwidth to services experiencing high latency or packet loss.

6. **Implement Resource Allocation**
   Once the AI agent has made a decision, it sends the resource allocation instructions back to the network to be implemented. The network management system carries out the resource changes based on the AI agent's recommendation, dynamically adjusting resources between the two network slices.

7. **Observe Network Performance**
   After the resource allocation is implemented, the network performance is observed to assess how effective the changes were. New QoS metrics are gathered, which help evaluate the impact of the agent's decisions.

8. **Calculate Reward**
   The AI agent receives feedback in the form of a reward based on how well the new resource allocation performed. The reward is calculated based on the improvement (or degradation) in the QoS metrics. For instance, reduced latency and packet loss could result in a higher reward.

9. **Update Environment State**
   The environment is updated to reflect the new network state after the resource allocation changes have been implemented. The AI agent will now work with the updated network conditions to further optimize its decisions.

10. **Agent Updates Policy**
    The AI agent uses the feedback (reward) to update its policy. The policy is the agent's internal model for making future decisions. By adjusting its policy, the agent learns from past actions and improves its decision-making ability over time.

11. **Stop Command From NNA**
    The optimization continues for as long as the NNA provides the network state and the QoS KPIs, otherwise the process stops. The NNA determines whether the process stops or continues.

12. **End**
    The workflow ends once the agent determines that no further optimization is necessary, or the system reaches a predefined stopping criterion.

Testing and Validation:

Figure 12 depicts the ongoing testing and validation status of the system in a simulated environment. The figure shows early results which were recorded during initial experiments, which are still ongoing. The loss over all the episodes is expected to drop over time as the AI agent learns through the environment to optimize our KPIs, by maximizing throughput and imposing penalties on higher latency jitter, and packet loss. Figure 12 shows how the AI fairly allocates the resources to both slices as clearly defined in the reward function, and this is validated as the agent is expected to allocate 60% to slice 1 and 40% to slice 2. This was achieved while the reward moved in an upward trend. The AI agent is capable of effectively and dynamically allocating resources based on the needs which are expected to be dynamic.

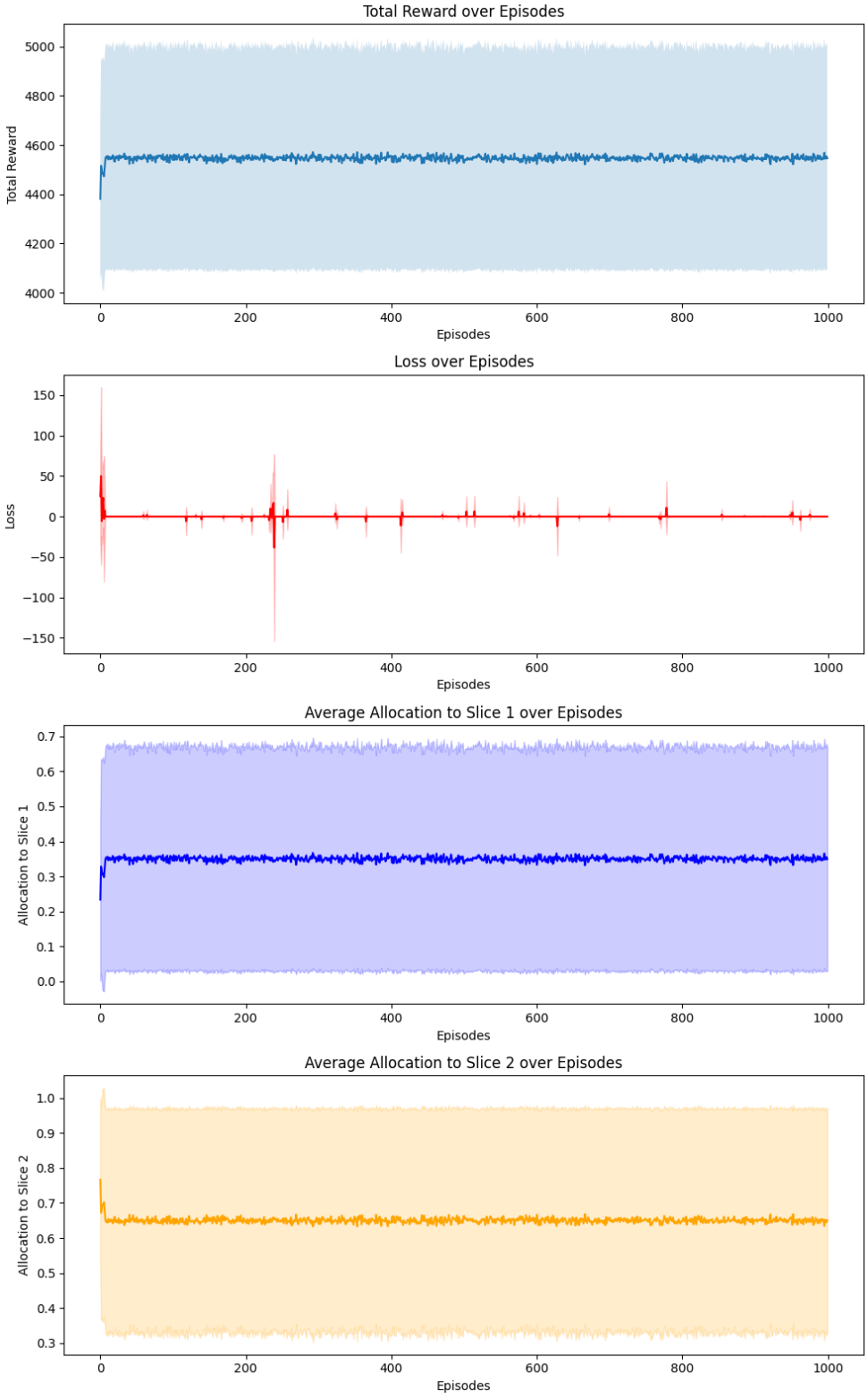

Figure 12: Resource optimization performance in a simulated environment

6 EDGE NORTHBOUND INTERFACE (NBI) API

The primary MEC Orchestrator (IEAP) in 6G-XR exposes an NBI API for application lifecycle management (LCM), including artifact management, application instance and application provider resource management aligned with ETSI MEC specification [9]. Details on this LCM API are provided in section 6.1.

The NBI edge MEC includes also other APIs that contribute to the ease of access to network capabilities for application providers and which are aligned and contributing to the CAMARA Linux Foundation project [10] . These are the *Quality on Demand*, *Edge Discovery* and *Traffic Influence* APIs which are documented in sections 6.2, 6.3 , and 6.4 respectively.

6.1 LIFECYCLE MANAGEMENT (LCM) API

6.1.1 Functional description

As defined by ETSI MEC [9], the LCM API is in charge of managing the complete lifecycle of the applications, covering:

- Application onboarding (descriptor + artifacts)
- Application instantiation
- Instance monitoring
- Instance updates
- Application instance termination

This API implements the analogous functionalities of the CAMARA MEC Experience Management and Exposure API (part of Edge-Cloud group).

6.1.2 API methods

At the Madrid – 5Tonic testbed the MEC orchestrator implements a Capgemini proprietary solution (IEAP). The following methods are provided:

- POST /lcm: Instantiates the application in a zone. Returns an instance id that will confirm the successful launching.
- GET /lcm: Obtains the runtime status of the application instance that was returned from the POST body.
- DELETE /lcm: Terminates the application instance that was returned from the POST body.

6.1.3 Workflows

The LCM workflow in the IEAP MEC orchestrator is depicted in Figure 13. The Holo Orchestrator sends the service definition to IEAP's MEO, which will onboard the app for a specific list of zones. Artifacts will later be attached to the application and then the app will be launched.

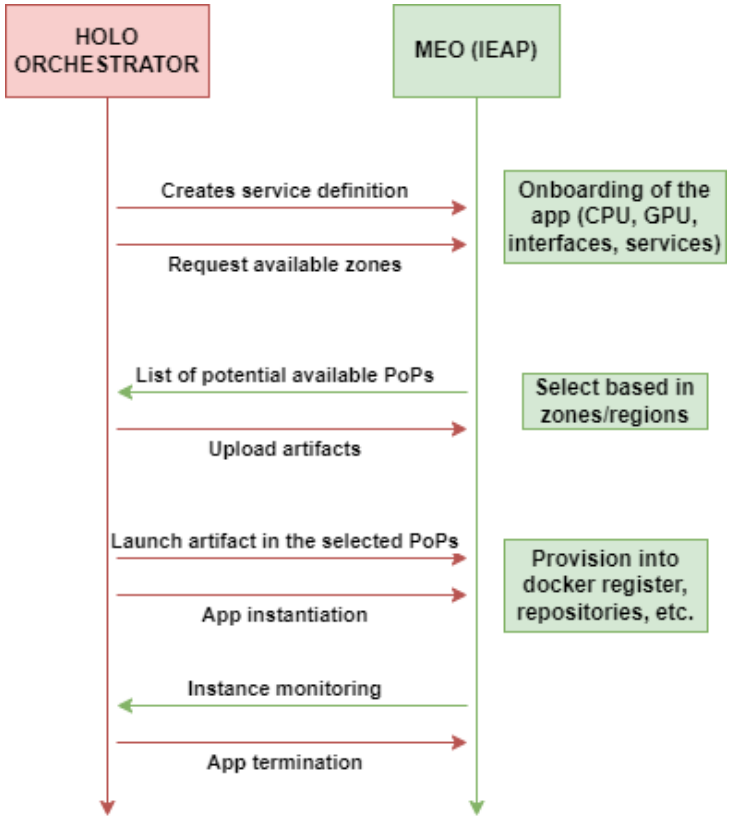

Figure 13: MEC orchestrator Application Lifecycle Management API workflow

6.2 QUALITY ON DEMAND (QOD) API

6.2.1 Functional Description

The Quality on demand (QoD) API provides the capability of setting the quality for a flow within an access network connection and gets a notification in case the network cannot fulfil it. Different levels can be defined according to the quality class indicators (QCIs) in the network, for instance:

- QOS_E: prioritization of low delay (low throughput in certain limit)
- QOS_S: prioritization of throughput up to a limit (low)
- QOS_M: prioritization of throughput up to a limit (medium)
- QOS_L: prioritization of throughput up to a limit (high, or no limit)

This API is aligned with CAMARA LF QoD API. This API will be used as part of Use Case 1.

6.2.2 API Methods

The QoD API provides the following methods:

- POST /qod/v0/sessions: Create QoS Session to manage latency/throughput priorities.
- GET /qod/v0/sessions/{sessionId}: Querying for QoS session resource information details.

- DELETE /qod/v0/sessions/{sessionId}: Release resources related to QoS session.
- POST /qod/v0/sessions/{sessionId}/extend: Extend the overall duration of an active QoS session. If this operation is executed successfully, the new duration of the target session will be the original duration plus the additionally requested duration.
- GET /qod/v0/qos-profiles: Returns all QoS Profiles that match the given criteria. If no criteria is given, all QoS Profiles are returned.
- GET /qod/v0/qos-profiles/{name}: Returns a QoS Profile that matches the given name.

6.2.3 Workflows

The QoD API general workflow is represented in Figure 14. The NEF makes a notification representing network congestion, then the Holo Orchestrator will request and create a QoS session with the network information, which will be forwarded to the UE.

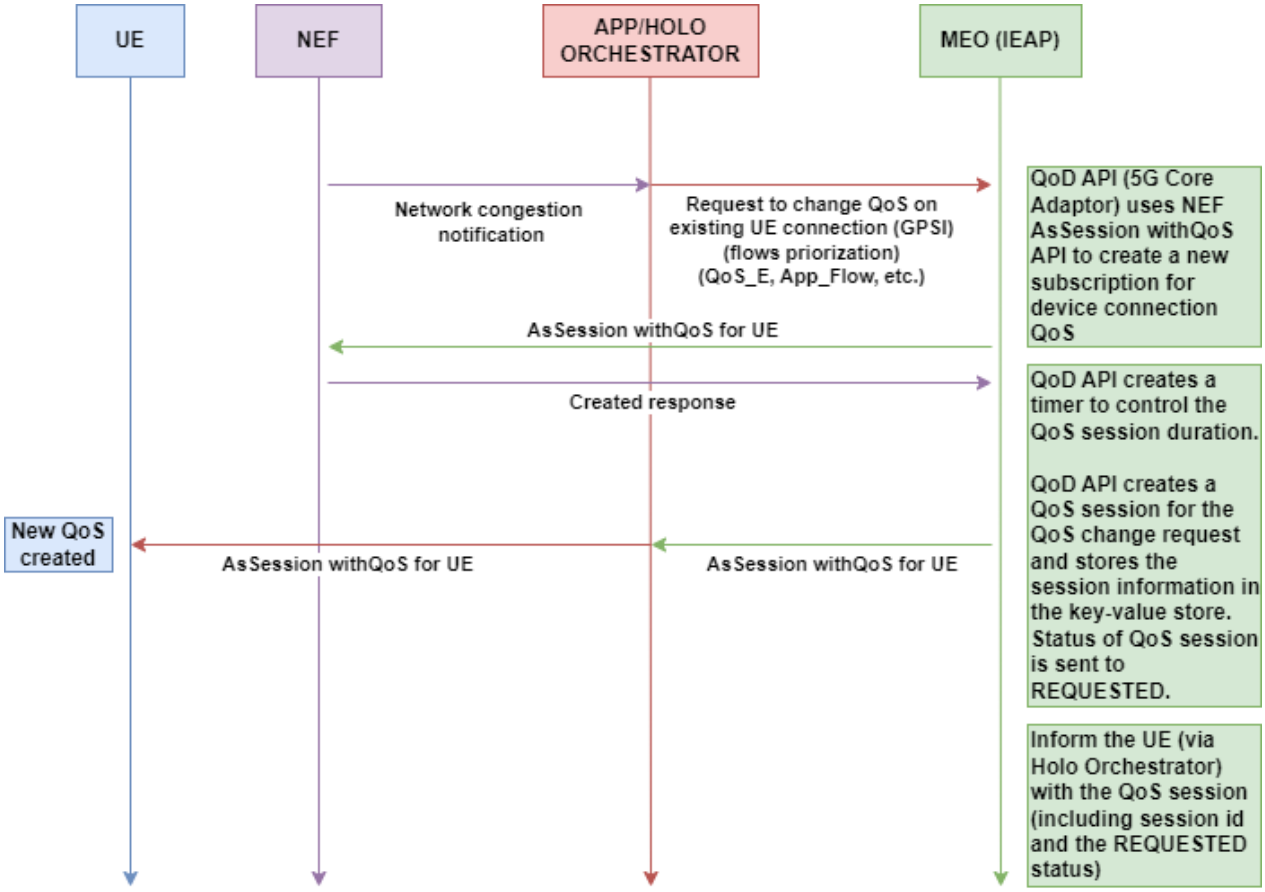

Figure 14: Quality on Demand API workflow

6.3 EDGE DISCOVERY API

6.3.1 Functional Description

The edge discovery API enables to retrieve a list of available and suitable edge platforms according to the requirements set for each application and as closest as possible to the User Equipment (UE) location. The closest Edge can be obtained based on the IP address, phone number (*MSISDN - Mobile Station International Subscriber Directory Number*, in 3GPP terminology), or Network Access Identifier (*GPSI – Generic Public Subscription Identifier*, in 3GPP terminology). For that purpose, the MEC Orchestrator will be querying the NEF regarding the Tracking Area Code (TAC).

This API is aligned with CAMARA LF Simple Edge Discovery API (Part of Edge-Cloud group). This API will be used as part of Use Case 2.

6.3.2 API Methods

The Edge Discovery API provides the following methods:

- GET /mec-platforms: Returns the names of the available Edge Cloud Zones which is the closest to the user device identified in the request.

6.3.3 Workflows

The Edge Discovery API general workflow is represented in Figure 15. The Edge Discovery API will receive a request based on the IP/GPSI to get the available Points of Presence, which will be forwarded to the NEF for event handling, then this information will be forwarded back to the Holo Orchestrator to choose which PoP will be used for app instantiation.

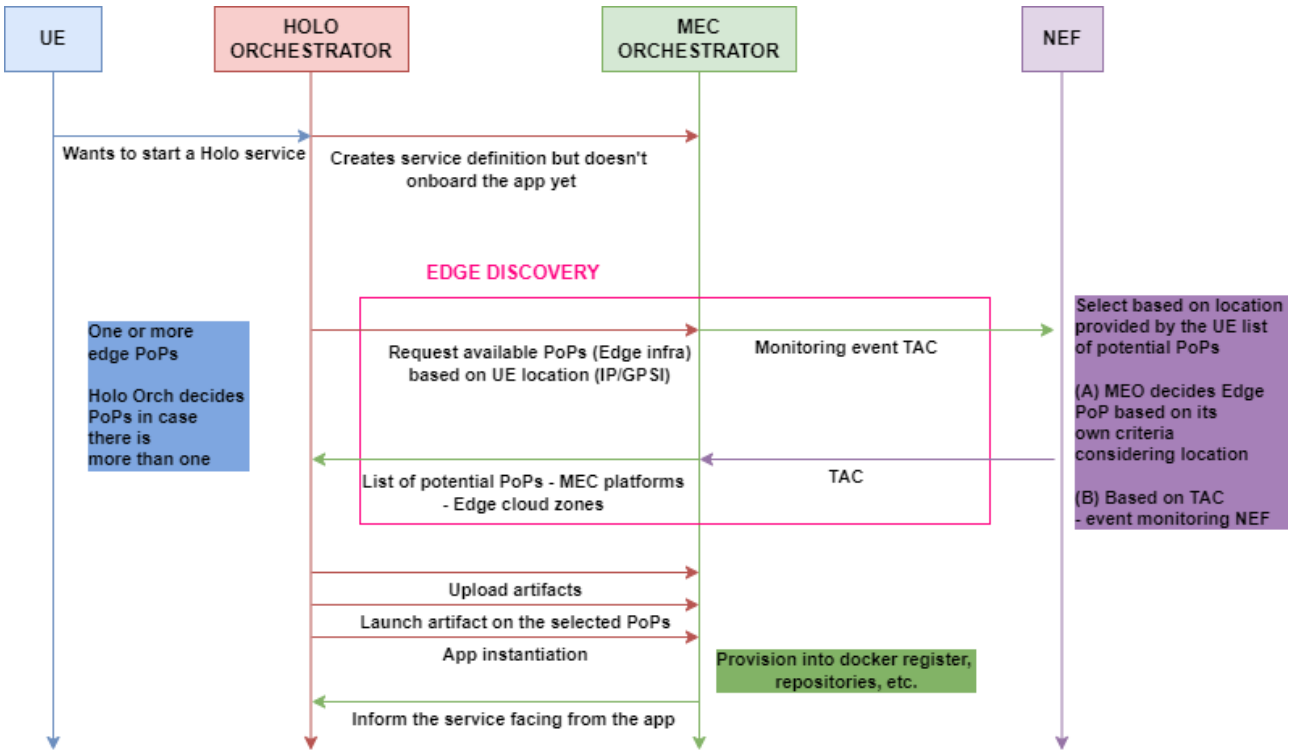

Figure 15: Edge Discovery API workflow

6.4 TRAFFIC INFLUENCE API

6.4.1 Functional Description

The Traffic Influence (TI) API provides the capability for the application providers to request the minimal latency in a specific geographical area to enhance the quality of experience. This is done by leveraging on local instances of the application deployed at the Edge, influencing the traffic routing from the UE towards the edge instance of the application. The edge platform for optimal routing (minimum delay) can be obtained based on IP address, phone number (*MSISDN*, in 3GPP terminology), or Network Access Identifier (*GPSI)*. For that purpose the MEC Orchestrator will be interacting with the Service Parameter API of the NEF regarding the UE Route Selection Policy (URSP).

As alignment with CAMARA is targeted, the final design of the TI API (part edge-cloud group) is expected by December'24. This API will be used as part of Use Case 2.

6.4.2 API Methods

This API provides the following methods (work in progress, the final implementation may vary):

- GET /traffic-influences: Reads all of the active TI resources owned by the same API Consumer.
- GET /traffic-influences/{trafficInfluenceID}: Returns a specific TI resources owned by the same API Consumer.
- POST /traffic-influences: Takes as input an object containing the intents from the API Consumer and creates a TI resource accordingly. The trafficInfluenceID parameter, that is part of the object, must not be valorized when creating a new resource. For this reason, the trafficInfluenceID parameter must be avoided in the object, anyway it will be ignored by the API Producer. It is automatically generated by the system and returned in the response.
- PATCH /traffic-influences/{trafficInfluenceID}: Updates a specific TI resource, identified by the trafficInfluenceID value.
- DELETE /traffic-influences/{trafficInfluenceID}: Invoked by the API Consumer to stop influencing the traffic, deleting a TI resource previously created.

6.4.3 Workflows

The TI API general workflow is represented in Figure 16.

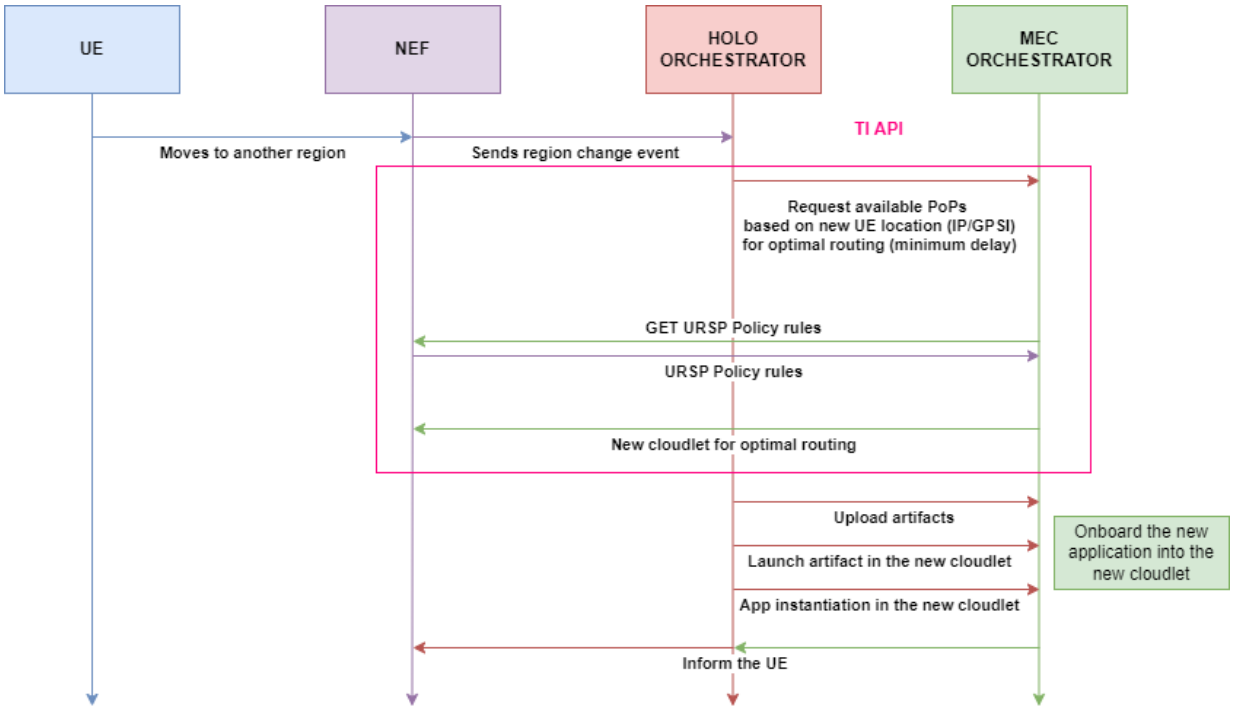

Figure 16: Traffic Influence API Workflow

It should be noted that the interaction with the NEF to influence the traffic by means of the URSP is still to be defined and will be detailed in the deliverable D2.3 which is due by May'2025.

7 FEDERATION

In 6G-XR, and specifically within UC2, federation plays a crucial role. It enables an operator to access a broader set of resources by establishing agreements with other infrastructure providers. To establish federation, both compute domains (operator domains) must have a MEC federator (MEF) component. This component interfaces with the other domain and interacts with the local edge orchestrator. The following sections will detail the various components, exposed APIs, and workflows involved.

It should be noted that in In the North Node, no separate East-Westbound federation has been implemented. VTT 5GTN and the University of Oulu 5GTN IP networks have already been integrated together before the 6G-XR started. Both networks are fully accessible from the other network. Therefore, there was no need to implement East-Westbound federation between the test facilities.

7.1 FUNCTIONAL DESCRIPTION

6G-XR has adopted the specifications proposed by the GSMA Operator Platform Group (OPG) for federation. These specifications aim to standardize an API for the East-Westbound interface [11]. At the time of writing, the project adheres to release 4 of these specifications, aligning with the proposals of ETSI.

For federation to be established, the EWBI interface requires two MEC Federator components, one on each side. Figure 17 illustrates the high-level architecture of the federation between two operators.

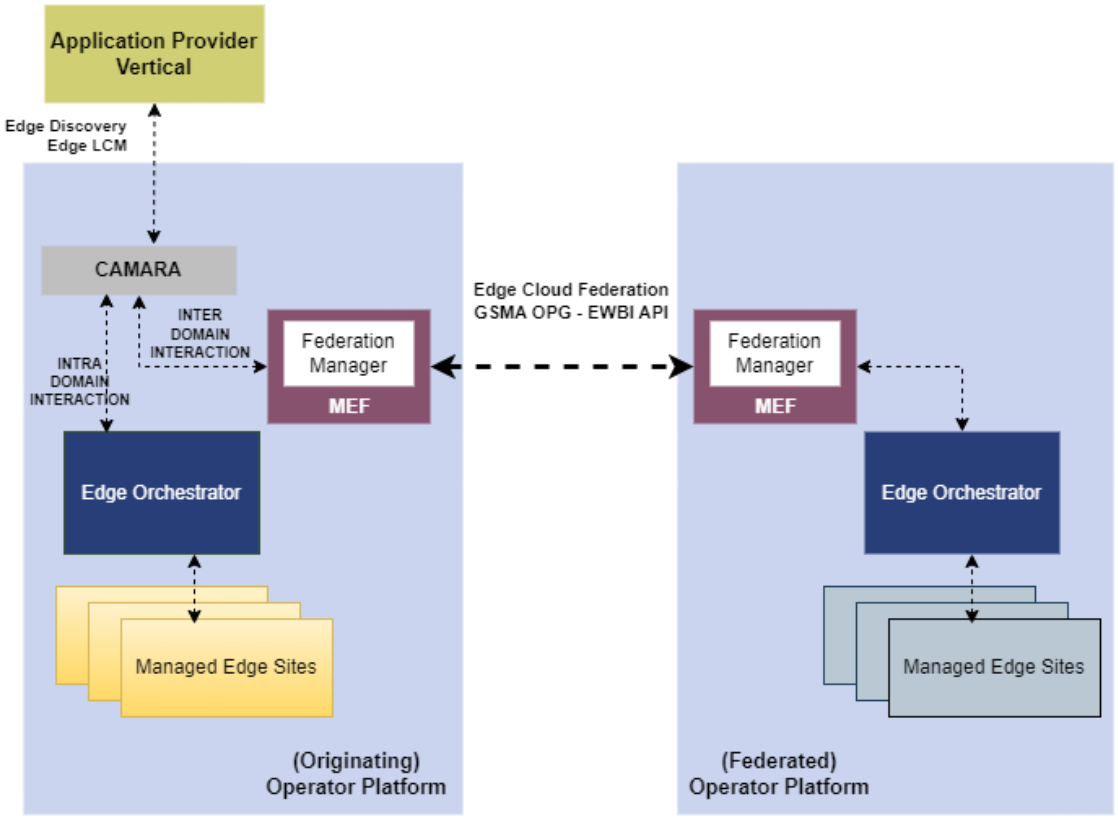

Figure 17: High level architecture for edge federation

It is important to note that the federation relationship between two operators is uni-directional. A federation creation request is initiated by one operator to another, the partner operator. This request results in the exposure of the partner operator's edge cloud resources and network capabilities to the requesting operator. If both operators wish to expose their resources to each other, they must each initiate a federation creation request.

7.2 FEDERATION EXTERNAL APIS

GSMA OPG EWBI has defined a comprehensive list of methods to cover a myriad of scenarios. These methods can be grouped according to their functionalities into the following main categories:

1. Federation Management
2. Availability Zones Information Synchronization
3. Artifacts Management
4. Application Onboarding
5. Application LCM (Lifecycle Management)

The following subsections detail these functionalities.

7.2.1 Federation Management

Federation Management focuses on establishing and maintaining federation relationships between two domains. The key methods include:

- CreateFederation: Creates a directed federation relationship with a partner operator platform (OP).
- GetFederationDetails: Retrieves details about the federation relationship with the partner OP, including information about the zones offered by the partner, partner OP network codes, edge discovery, and LCM service.
- DeleteFederationDetails: Removes an existing federation with the partner OP.

13. **API methods**:

    - POST /partner: Creates a one-direction federation with a partner OP.
    - GET /{federationContextId}/partner: Retrieves details about the federation context with the partner OP.
    - PATCH /{federationContextId}/partner: Updates the parameters associated with the existing federation.
    - DELETE /{federationContextId}/partner: Removes an existing federation with the partner OP.

14. **Federation Management Workflow**:

    Following picture Figure 18 illustrates the workflow for the establishment of a Federation between two operators.

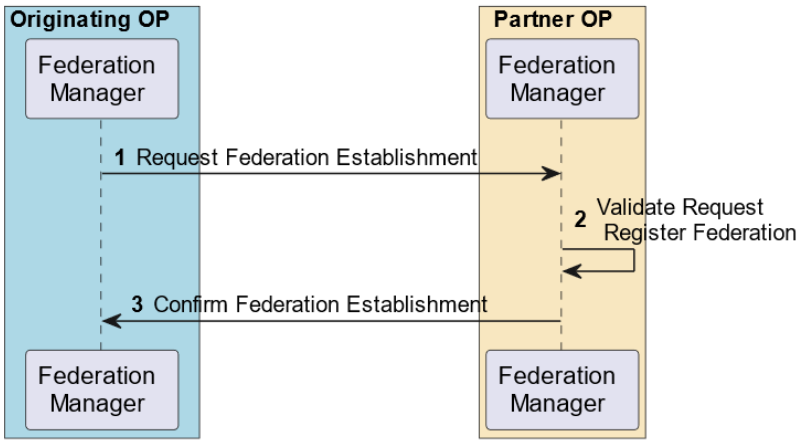

Figure 18: Federation Establishment

7.2.2 Availability Zones Information Synchronization

This category manages the resources of partner OP zones and provides status updates. It includes methods for synchronizing zone information:

- ZoneSubscribe: Notifies the partner OP that the originating OP is willing to access specified zones, prompting the partner OP to reserve compute and network resources.

- ZoneUnsubscribe: Informs the partner OP that the originating OP will no longer access a specified zone.
- GetZoneData: Retrieves details about the computation and network resources reserved by the partner OP for a specific zone.

1. **API methods**:

    - POST /{federationContextId}/zones: Informs the partner OP of the originating OP interest in specified zones.
    - DELETE /{federationContextId}/zones/{zoneId}: Indicates that the originating OP will no longer access a specified zone.
    - GET /{federationContextId}/zones/{zoneId}: Retrieves details about the resources reserved by the partner OP for the zone.

2. **Availability Zone Info Synchronization Workflow**:

The following image Figure 19 illustrates the workflow used by the two operators to exchange information about the availability zones. This information will be required at the time of application deployment.

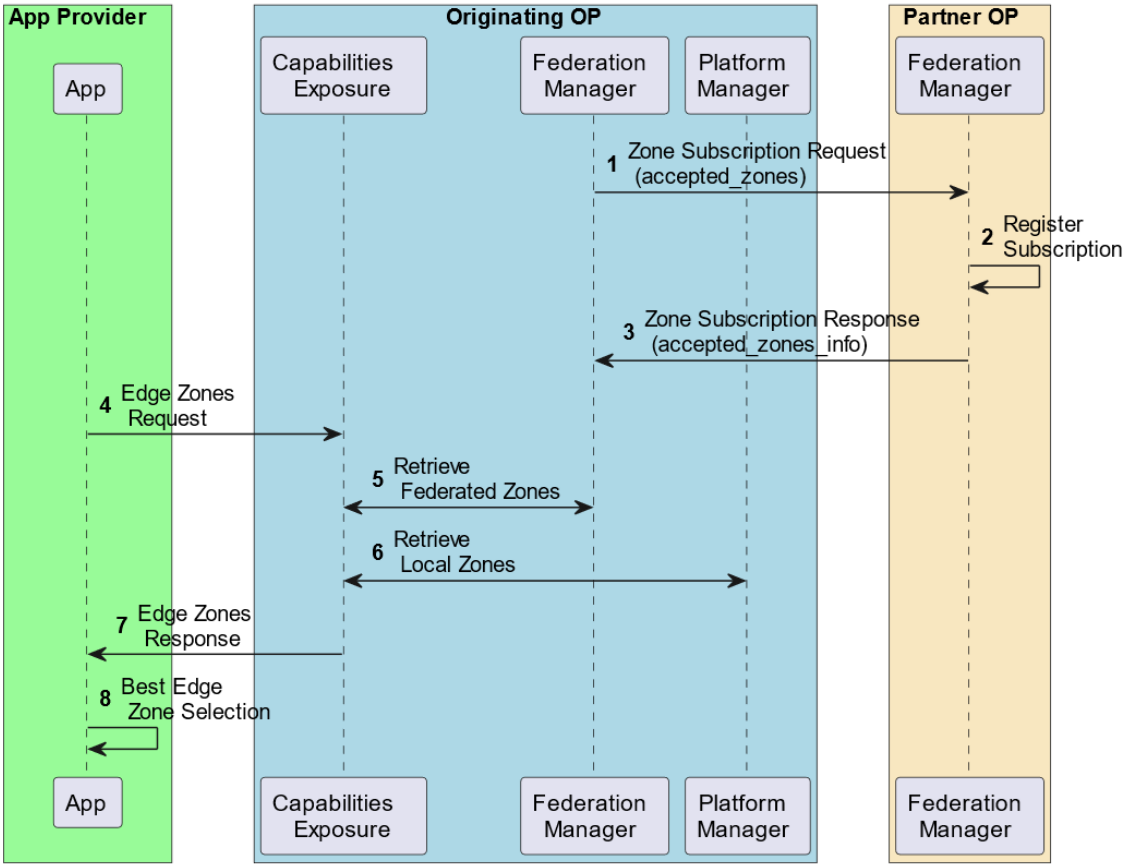

Figure 19: Availability Zone Exposure workflow

7.2.3 Artifacts Management

The Artifacts Management handles the uploading, removal, retrieval, and updating of application descriptors, charts, and packages over EWBI towards a partner OP:

- UploadArtefact: Uploads an application artifact to the partner OP.
- RemoveArtefact: Removes an artifact from the partner OP.
- GetArtefact: Retrieves details about an artifact from the partner OP.
- UploadFile: Uploads application binaries to the partner OP.
- RemoveFile: Removes application binaries from the partner OP.
- ViewFile: Retrieves details about binaries associated with an application from the partner OP.

1. **API methods**:

    - POST /{federationContextId}/artefact: Uploads an application artifact (a zip file containing scripts or packaging files like Terraform or Helm).

- GET /{federationContextId}/artefact/{artefactId}: Retrieves details about an artifact.
- DELETE /{federationContextId}/artefact/{artefactId}: Removes an artifact.
- POST /{federationContextId}/files: Uploads an image file.
- DELETE /{federationContextId}/files/{fileId}: Removes an image file.
- GET /{federationContextId}/files/{fileId}: Retrieves info about the image file.

2. **Workflow for Artefact Management**:

The next image Figure 20 illustrates the required workflow to upload an artefact file, from the OP to the partner OP, to later be able to deploy that image.

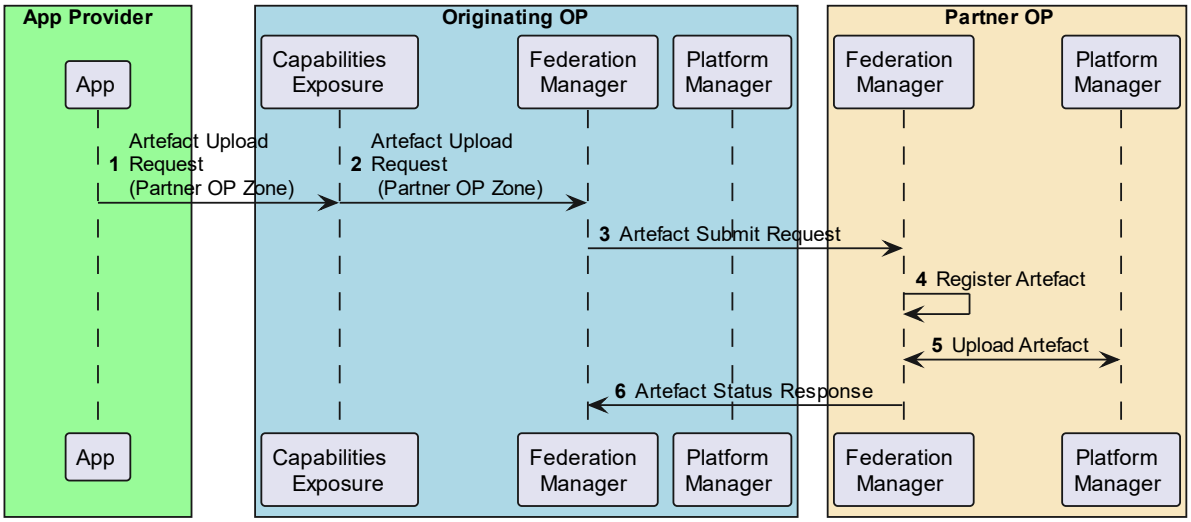

Figure 20: Artefact Management

7.2.4 Application Onboarding

This set of methods allows to register, retrieve, update, and remove applications over E/WBI towards a partner OP.

- OnboardApplication: Submits application details to a partner OP. Based on the details provided, partner OP shall do bookkeeping, resource validation and other pre-deployment operations.
- UpdateApplication: Updates partner OP about changes in application compute resource requirements, QOS Profile, associated descriptor or change in associated components.
- DeboardApplication: Removes an application from partner OP.
- ViewApplication: Retrieves application details from partner OP.
- OnboardExistingAppNewZones: Make an application available on new additional zones.
- LockUnlockApplicationZone: Forbid or permit instantiation of application on a zone.

1. **API methods:**

- POST /{federationContextId}/application/onboarding: Submits application details to a partner OP for bookkeeping, resource validation, and other pre-deployment operations.
- DELETE /{federationContextId}/application/onboarding/app/{appId}: Deboards the application from all zones and deletes the app.
- PATCH /{federationContextId}/application/onboarding/app/{appId}: Updates the partner OP about changes in application compute resource requirements, QoS profile, associated descriptor, or change in associated components.
- GET /{federationContextId}/application/onboarding/app/{appId}: Retrieves application details from the partner OP.
- DELETE /{federationContextId}/application/onboarding/app/{appId}/zone/{zoneId}: Deboards an application from specific partner OP zones.
- POST /{federationContextId}/application/onboarding/app/{appId}/additionalZones: Onboards an existing application to a new zone within the partner OP.
- POST /{federationContextId}/application/onboarding/app/{appId}/zoneForbid: Forbids or allows application instantiation on a partner zone.

2. **Workflow for Application Onboarding Management**:

The following image Figure 21 illustrates, the way that from the artefact and application is created, which can later be instantiated. This step is required to be able to deploy an artefact.

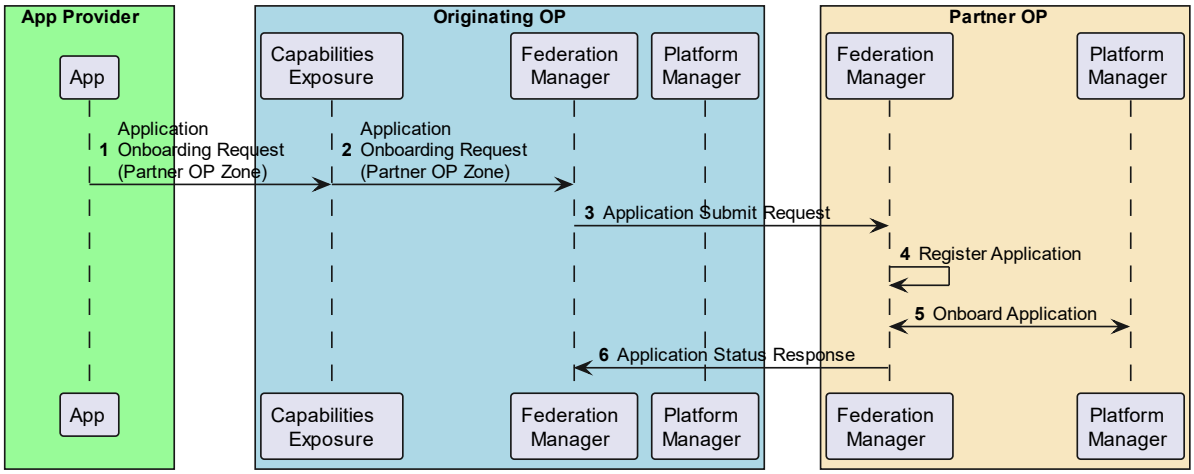

Figure 21: Application Onboarding Workflow

7.2.5 Application Instance LCM (Lifecycle Management)

Lastly Application Instance LCM category groups the methods for creating, updating, retrieving, and terminating application instances over EWBI towards a partner OP:

- InstallApp: Instantiates an application on a partner OP zone.

- GetAppInstanceDetails: Retrieves an application instance details from partner OP.
- RemoveApp: Terminates - Terminate an application instance on a partner OP zone.
- GetAllAppInstances: Retrieves details about all instances of the application running on partner OP zones.

1. **API Methods**:
    - POST /{federationContextId}/application/lcm: Instantiates an application on a partner OP zone.
    - GET/{federationContextId}/application/lcm/app/{appId}/instance/{appInstanceId}/zone/{zoneId}: Retrieves application instance details from the partner OP.
    - DELETE/{federationContextId}/application/lcm/app/{appId}/instance/{appInstanceId}/zone/{zoneId}: Terminates an application instance on a partner OP zone.
    - GET/{federationContextId}/application/lcm/app/{appId}/appProvider/{appProviderId}: Retrieves all application instances of the partner OP.

2. **Application Deployment Workflow**

The following workflow Figure 22 shows, the process to instantiate an application. It assumes that the previous steps have been already done.

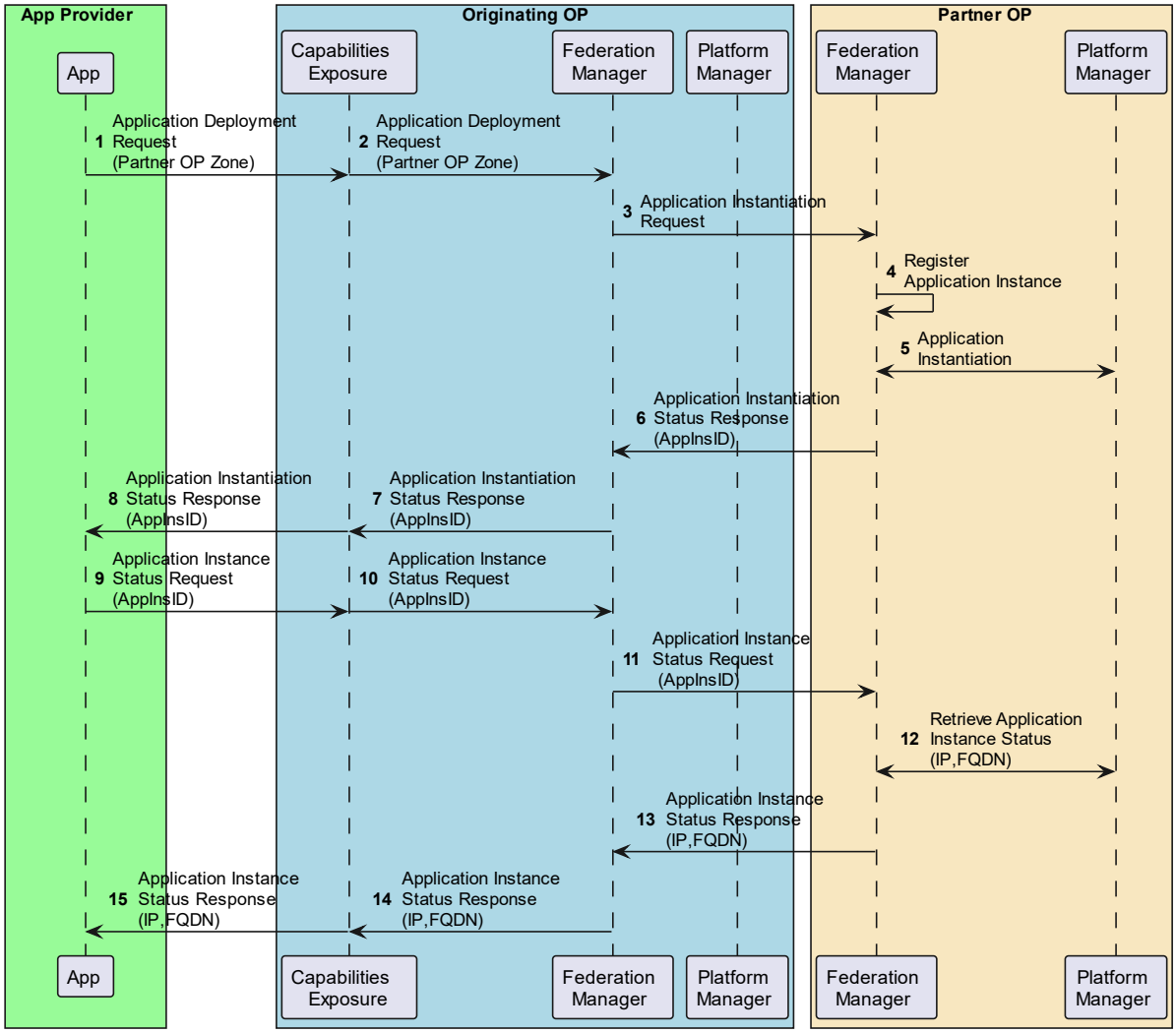

Figure 22: Application Deployment Workflow

8 NEF APIS FOR SLICING

This chapter describes the APIs developed on the NEF in the South Node 5Tonic lab. This component implements a series of APIs to expose network data such as metrics or user location, and network capabilities like slicing or QoS on demand, to third-parties (3PP) Application Functions (AFs).

Since the submission of deliverable D2.1 [2], the development of these features and APIs has progressed. The new NEF component implements an extension of 3GPP NEF APIs, to fit the specific needs of the project UCs:

- Data Collection API – Fetch KPIs: This API is an extension of 3GPP Analytics Exposure API described in D2.1.
- Service Parameter API – List, create or update the slices associated with a given user: This API is an extension of 3GPP Service Parameter API described in D2.1.
- UE Location API – Get UE location: This API is an extension of 3GPP Monitoring Event API described in D2.1.
- QoS Session API (invoked by CAMARA Quality-On-Demand API) – List, create or update the associated QoS to an application and user: This API is an extension of 3GPP AF Session with QoS API described in D2.1.

8.1 DATA COLLECTION API

8.1.1 Functional Description

The NEF Data Collection API allows the AF to retrieve analytics information, such as relevant KPIs per cell and per UE, which are collected periodically from network functions and network probes, and stored in an InfluxDB. In UC1 *Resolution Adaptation or Quality on Demand*, the Congestion Detection Function (CDF) performs a congestion detection and alert mechanism, based on information requested periodically to the network, like metrics of the holographic application user and the cell where the user is located.

The KPIs per cell supported by Data Collection API are listed in Table 5.

Table 5: Traffic KPI RAN data per cell

KPI	Description
RanDownlinkThroughput	Downlink throughput in kbps per cell id
RanUplinkThroughput	Uplink throughput in kbps per cell id
ActiveUsersDL	Number of active users in DL per cell id
ActiveUsersUL	Number of active users in UL per cell id
HarqDlAck16Qam	Total number of successful downlink HARQ transmissions using 16-QAM modulation

HarqDlAck64Qam	Total number of successful downlink HARQ transmissions using 64-QAM modulation
HarqDlAck256Qam	Total number of successful downlink HARQ transmissions using 256-QAM modulation
HarqDlAckQpsk	Total number of successful downlink HARQ transmissions using QPSK modulation
HarqUlAck16Qam	Total number of successful HARQ transmissions in uplink using 16-QAM modulation
HarqUlAck64Qam	Total number of successful HARQ transmissions in uplink using 64-QAM modulation
HarqUlAck256Qam	Total number of successful HARQ transmissions in uplink using 256-QAM uplink
HarqUlAckQpsk	Total number of successful HARQ transmissions in uplink using QPSK modulation
PdschTable1McsDistr	Distribution of MCS used for PDSCH transmissions where MCS index table 1 is applied and UE instances are enabled with up to 64-QAM
PdschTable2McsDistr	Distribution of MCS used for PDSCH transmissions where MCS index table 2 is applied and UE instances are enabled with up to 256-QAM
PdschTable3McsDistr	Distribution of MCS used for PDSCH transmissions where MCS index Table 3 is applied and UE instances are enabled
PuschTable1McsDistr	Distribution of MCS used for PUSCH transmissions where MCS index table 1 is applied and UE instances are enabled with up to 64-QAM
PuschTable2McsDistr	Distribution of MCS used for PUSCH transmissions where MCS index table 2 is applied and UE instances are enabled with up to 256-QAM
PuschTable3McsDistr	Distribution of MCS used for PUSCH transmissions where MCS index table 3 is applied and UE instances are enabled
UtilizationDistributionDL	Distribution of utilization for downlink resource block symbols averaged over 1 second per cell id
UtilizationDistributionUL	Distribution of utilization for uplink resource block symbols averaged over 1 second per cell id

The KPIs per UE supported by Data Collection API are listed in Table 6.

Table 6: Traffic KPI data per UE

KPI	Description
UeDownlinkThroughput	Downlink throughput in kbps per UE
UeUplinkThroughput	Uplink throughput in kbps per UE

8.1.2 API Methods

NEF Data Collection API methods:

- POST /analyticsexposure/{npnId}/fetch: Retrieves analytics data such as the location information (given a subscription), the network performance (given a subscription) or the radio metrics of a given cell.

8.1.3 Workflows

The workflow to fetch KPIs is shown in Figure 23.

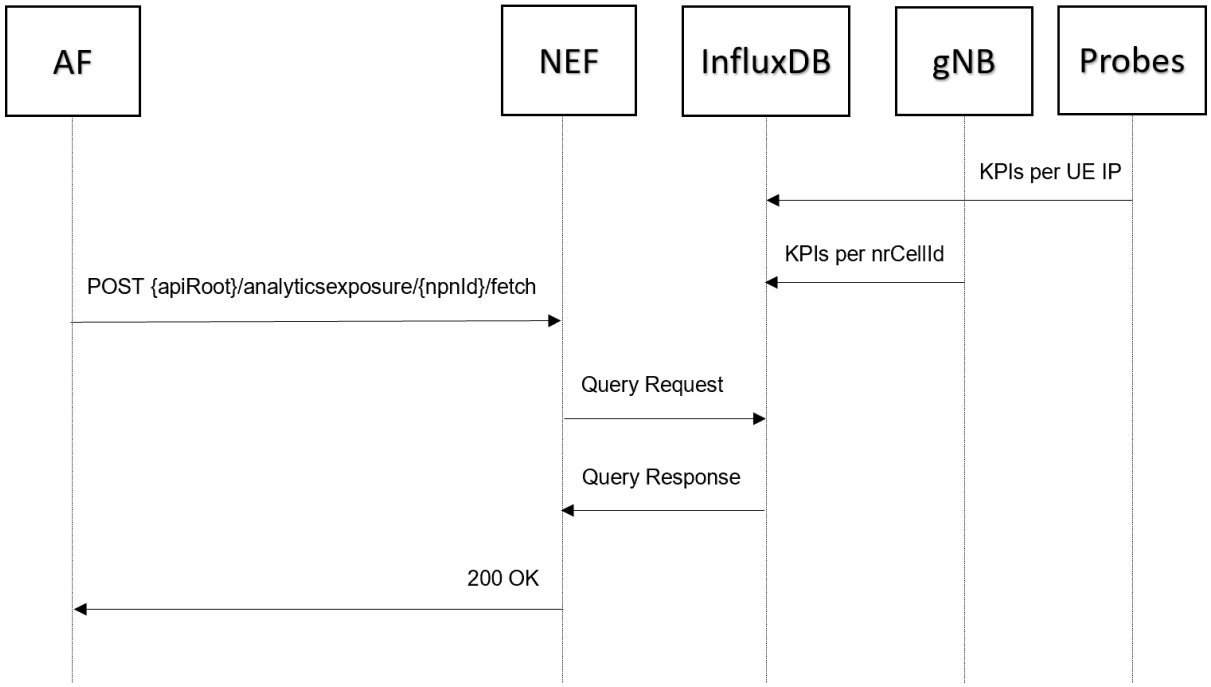

Figure 23: Data Collection API workflow

First, the AF sends a Hypertext Transfer Protocol (HTTP) POST *analyticsexposure* message to NEF requesting relevant KPIs per user and/or per cell, providing the UE IP and the New Radio (NR) cell identifier. Next, Southbound in the network, the NEF obtains the metrics information querying the InfluxDB where the metrics, obtained from the different network elements like gNB and network probes, are stored. Finally, the NEF sends a 200 response to the AF to acknowledge the request, including the requested KPIs for the provided UE IP and/or NR cell.

8.2 SERVICE PARAMETER API

8.2.1 Functional Description

The NEF Service Parameter API allows the AF to request the use of the most appropriate slice, using the UPF and Data Network Name (DNN) which are closer to the user. This is achieved by means of the UE Routing Selection Policy (URSP) feature. Thus, the AF will request a policy subscription (URSP rules) to be delivered to a UE, in order to use a certain Single Network Slice Selection Assistance Information (S-NSSAI) and DNN for its own application data flows.

Additionally, the Service Parameter API allows the AF to retrieve the list of slices available in a specific facility, and to assign slices from this list to a specific user. When setting up an experiment via *Trial Controller*, described in D4.1 [12], the experimenter selects on the corresponding node web portal the

initial slice to be used in the experiment among the available slices in each facility (for South Node: 5Tonic in Madrid or i2CAT in Barcelona). The South Node Adapter (SNA) achieves this invoking the NEF Service Parameter API to retrieve the available slices from the network, and to assign the slice selected by the experimenter to each UE.

In UC2 *Routing to the Best Edge*, the MEC Orchestrator (IEAP) assists the Holo Orchestrator (HO) when performing an app re-instantiation upon a change in UE location within the same Mobile Edge Orchestrator (MEO) or to a different MEO, by requesting the NEF to select a slice that uses the UPF and DNN closer to the user, to be used by the application flow.

8.2.2 API Methods

NEF Service Parameter API methods:

- GET /npn/{npnId}/slices: Retrieves the slice list IDs of a Non-Public Network (NPN) identified by npnId.
- GET /subscriptions/{subscriptionId}/slices: Retrieves the list of slices of a subscriber identified by subscriptionId.
- POST /subscriptions/{subscriptionId}/slices: Updates, given a subscriber identified by subscriptionId, the slices associated to the subscriber.

8.2.3 Workflows

- The workflow to get the list of slices in a specific facility is depicted in Figure 24.

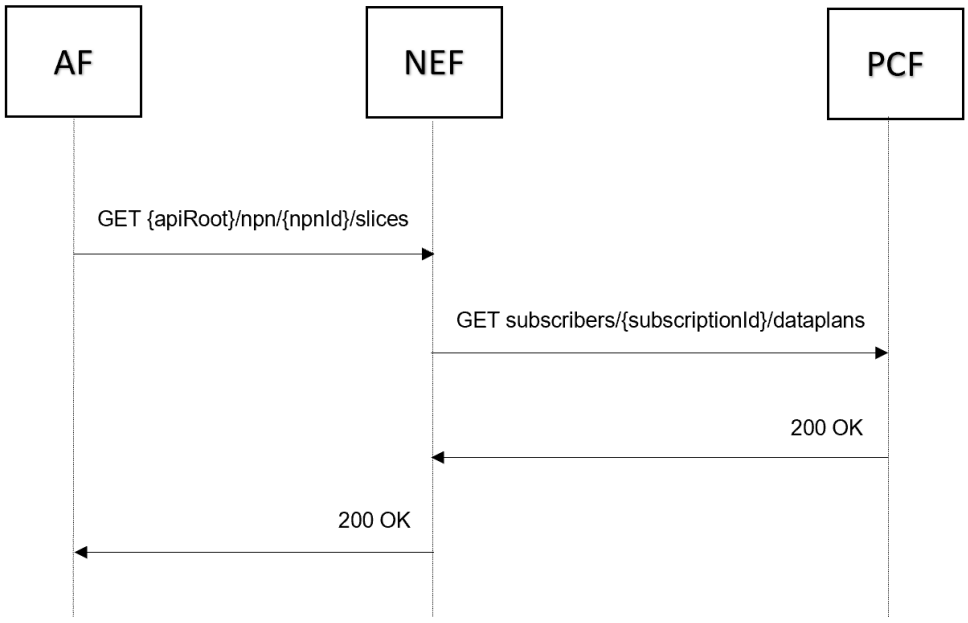

Figure 24: Service Parameter API workflow for "Slices list retrieval"

First, the AF initiates a NEF Service Parameter API request (GET /npn/{paid}/slices) for the retrieval of all the slices that can be selected in a specific facility. Then, Southbound in the network, NEF contacts Policy Control Function (PCF) to obtain the slices identifiers. Next, PCF returns HTTP response code 200

with a list of slices ids to NEF. Finally, NEF returns HTTP response code 200 with a list of slices ids to the AF.

- The workflow to select a specific slice for a UE is shown in Figure 25.

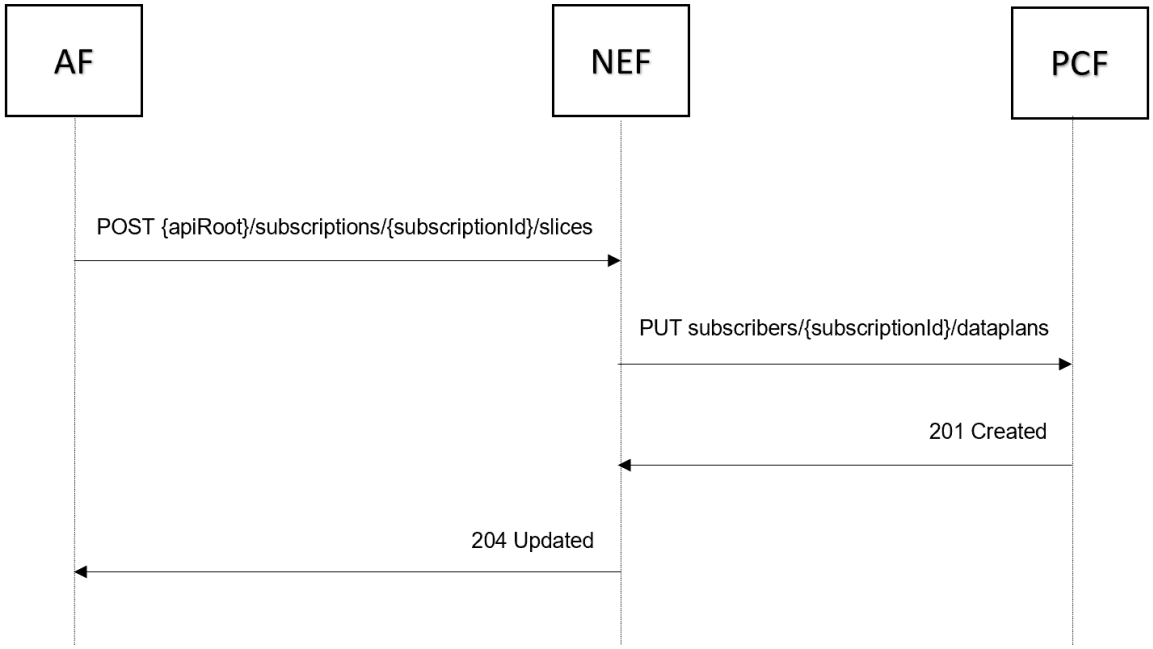

Figure 25: Service Parameter API workflow for "Slice selection"

First, the AF sends a NEF Service Parameter API request (POST /subscriptions/{subscriptionId}/slices) to select a specific slice from the available slices for a user, providing GPSI and the selected slice identifier. Then, Southbound in the network, NEF contacts PCF to provision a policy subscription, with the URSP rules indicating to use a certain S-NSSAI and DNN corresponding to the selected slice, to be assigned to the specified GPSI. Consequently, PCF initiates the 3GPP standard UE Policy delivery procedure, as specified in 3GPP TS 23.502 clause 4.2.4.3 [13]. This procedure workflow is depicted in Figure 26 below. Next, PCF acknowledges the provisioning with a 201 response. Finally, NEF responds with a 204 response, acknowledging the update of the slices associated to the subscriber, to the AF.

- The workflow for UE Policy delivery is shown in Figure 26. It has been extracted from 3GPP documentation. Please refer to 3GPP TS 23.502 clause 4.2.4.3 [13] for further details on this workflow.

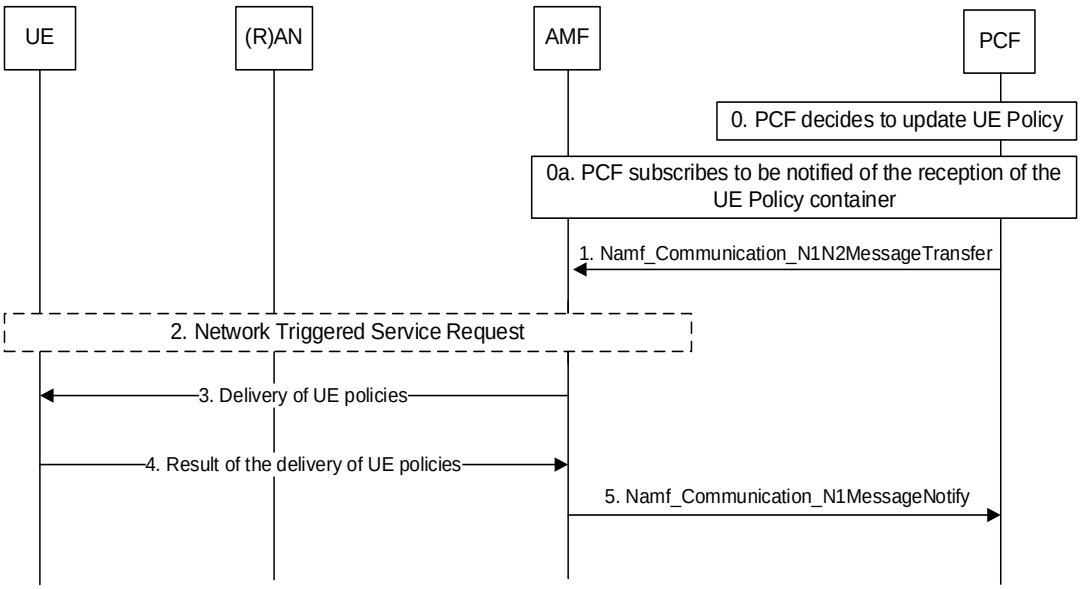

Figure 26: UE Configuration Update procedure for transparent UE Policy delivery

8.3 UE LOCATION API

8.3.1 Functional Description

The NEF UE Location API allows the AF to get information from the network about the location of the user, providing an identifier for the UE. Upon its request the NEF obtains this information from the Access and Mobility Management Function (AMF) or Session Management Function (SMF), which require the GPSI or Subscription Permanent Identifier (SUPI) as UE identifier. For this reason, if the GPSI or SUPI are unknown for the AF, the API also offers the AF the possibility to obtain the GPSI or SUPI from the UE IP and DNN.

In UC1 *Resolution Adaptation or Quality on Demand*, the CDF needs to know in which cell the user is located, to monitor the congestion upon this cell. If the UE moves to a different cell, the CDF will fetch the KPIs used for congestion detection in the new cell.

In UC2 *Routing to the Best Edge*, the Holo Orchestrator (HO) performs an initial app instantiation based on UE location provided by the MEC Orchestrator (IEAP), who obtains it from the NEF. Additionally, the HO, based on the UE location information it gets periodically from the NEF, performs an app re-selection without change of MEO when it detects that the UE has moved to other region within same MEO, or an app re-selection with change of MEO, when it detects that the UE has moved to another region within the federated MEO.

8.3.2 API Methods

NEF UE Location API methods:

- POST /analyticsexposure/{npnId}/fetch: Retrieves analytics data such as the location information (given a subscription), the network performance (given a subscription) or the radio metrics of a given cell.

8.3.3 Workflows

The workflow to fetch the location of a specific UE is shown in Figure 27.

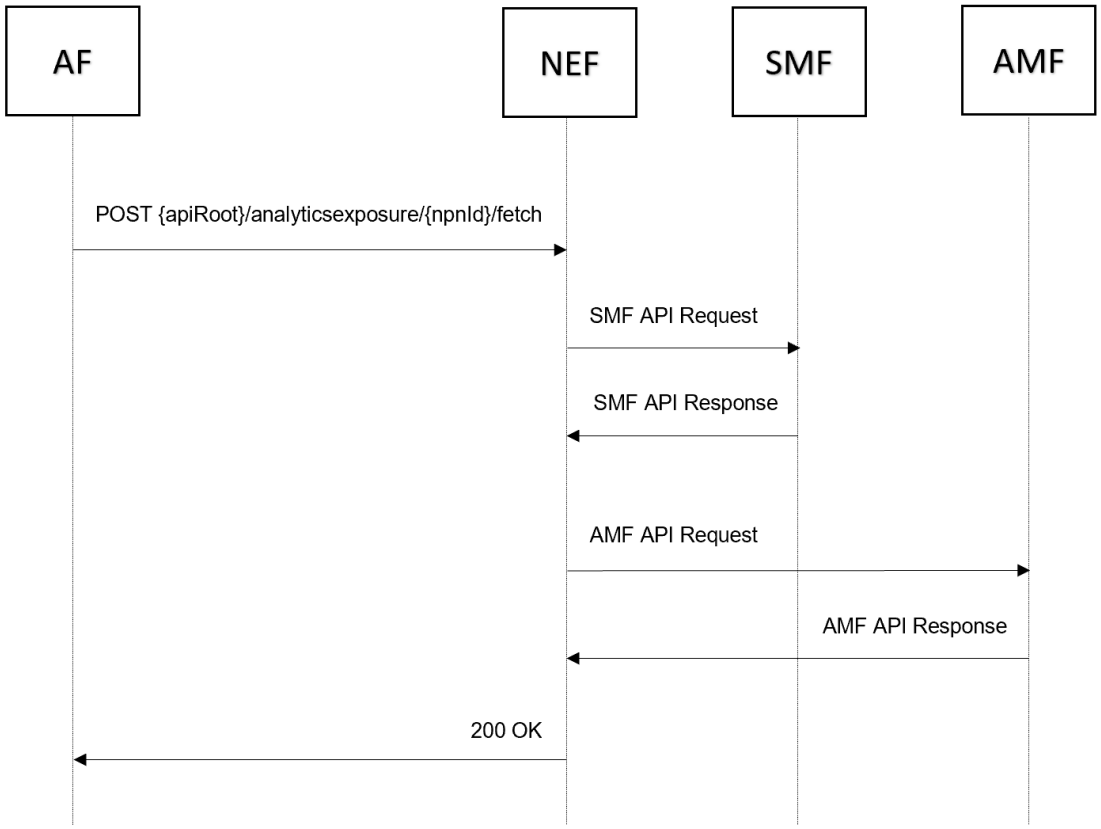

Figure 27: UE Location API workflow

First, the AF sends an HTTP POST *analyticsexposure* message to NEF requesting a specific UE location information, providing a UE identifier (GPSI or SUPI). When these identifiers are not known by the AF, it has to provide the UE IP and the DNN in the request body. Next, Southbound in the network, NEF obtains the UE location information from SMF or AMF, and the UE identifiers from SMF. Finally, NEF sends a 200 response to the AF to acknowledge the request, including the UE IP, GPSI, SUPI and NR cell where the UE is located.

8.4 QOS SESSION API

8.4.1 Functional Description

CAMARA QoD offers the application developers the capability to request for stable latency (reduced jitter) or throughput for some specified application data flows between application clients (within a user device) and application servers (backend services). The developer has a pre-defined set of QoS profiles (made available by the service provider, e.g., QOS_E, to map latency and throughput requirements, see section 6.2.1) which they could choose from, depending on their latency or throughput requirements. The usage of the QoD API is based on QoS profile classes and parameters which define App-Flows. Based on the API, QoS session resources can be created, queried, and deleted. Once an offered QoS profile class is requested, application users get a prioritized service with stable latency or throughput even in the case of congestion.

The NEF QoS Session API allows the AF to influence the QoS of one ongoing Protocol Data Unit (PDU) session of a UE, providing specific QoS requirements to the Unified Data Repository (UDR), which propagates them to the corresponding SMF, which initiates the PDU session modification. For instance, an AF can request a guaranteed bandwidth during a specific data communication session of the application users located in a cell where congestion has been reported, to secure the network resources to their users.

In UC 1 *Resolution Adaptation or Quality on Demand*, for the second case (QoD), once congestion is detected in a cell where a UE is running the XR application, the HO will do an On-demand Quality of Service Session request invoking the CAMARA QoD API, which in turn calls NEF QoS Session API to interact with the 5G network, aiming to prioritize XR flows, providing specific QoS requirements to the network, which initiates the PDU session modification.

8.4.2 API Methods

CAMARA QoD API methods:

- POST /sessions: Create QoS Session to manage latency/throughput priorities.
- GET /sessions/{sessionId}: Querying for QoS session resource information details.
- DELETE /sessions/{sessionId}: Release resources related to QoS session.
- POST /sessions/{sessionId}/extend: Extend the overall duration of an active QoS session.
- GET /qos-profiles: Returns all QoS Profiles that match the given criteria. If no criteria is given, all QoS Profiles are returned.
- GET /qos-profiles/{name}: Returns a QoS Profile that matches the given name.

NEF QoS Session API methods:

- GET /subscriptions/{subscriptionId}/qos: Retrieves the QoS associated to a specific subscription identified by subscriptionId.
- PUT /subscriptions/{subscriptionId}/qos: Creates or updates the QoS associated to a subscription identified by subscriptionId.

8.4.3 Workflows

The workflow to change the QoS of an existing device connection is shown in Figure 28.

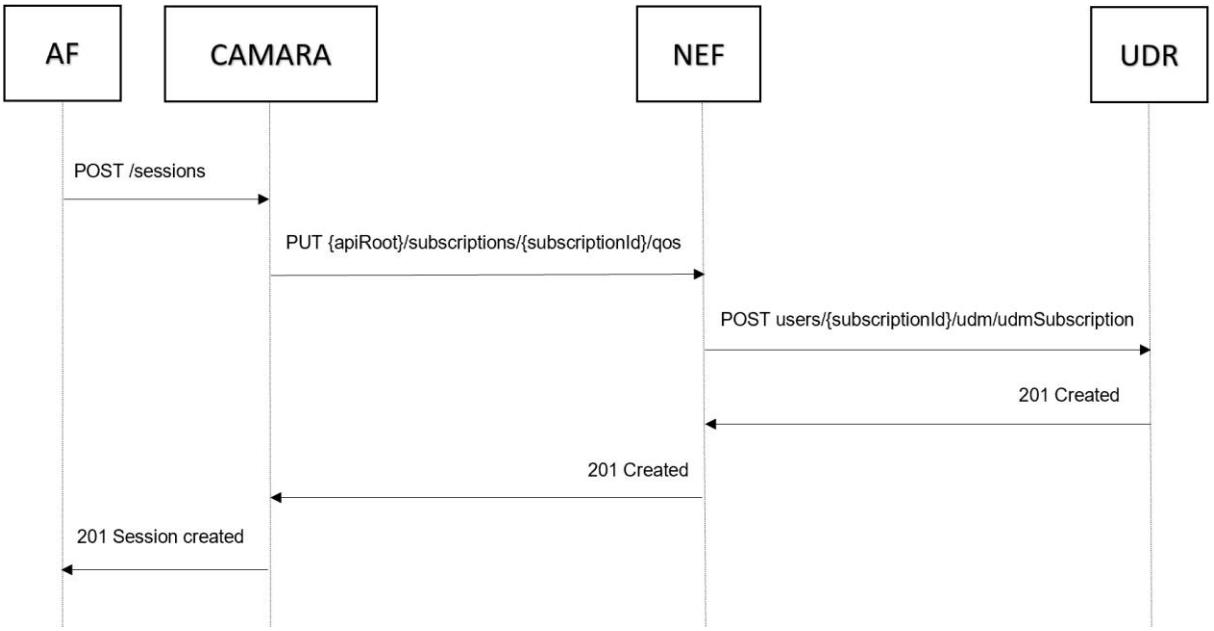

Figure 28: QoS Session API workflow

First, the AF initiates a CAMARA QoD API request (POST /sessions) to request QoS change of an existing device connection, specifying the desired QoS Profile (latency or throughput requirements of the application mapped to relevant QoS profile class) and other parameters which identify the precise application data flow the developer wants to prioritize and have stable latency or throughput for, like the identifier of the device (IPv4 address, IPv6 address, Phone number, or Network Access Identifier) and the identifier of the application server (IPv4 and/or IPv6 address). Then, the CAMARA QoD calls NEF by means of QoS Session API (PUT /subscriptions/{subscriptionId}/qos) to modify device connection QoS, providing the GPSI and the requested QoS Profile. Next, Southbound in the network, the NEF makes a request to UDR, to use the received QoS Profile, that is mapped to a set of QoS parameters, for the requested user data flow. Then, the UDR propagates the request to the SMF, which initiates the PDU session modification, as specified in 3GPP TS 23.502 clause 4.3.3.2 [13], starting from *2a. N4 Session Establishment/Notification* step in the worklow depicted in Figure 29. Next, the UDR acknowledges NEF request with a 201 response. Then, the NEF returns HTTP response code 201 with the information of the created subscription to CAMARA and the CAMARA QoD creates a QoS session for the QoS change request and stores the session information. The status of the QoS session is set to REQUESTED, which indicates that QoS change has been requested by creating a session. Finally, the CAMARA QoD returns HTTP response code 201 with the information of the created QoS session (including session ID and the REQUESTED status) to the AF.

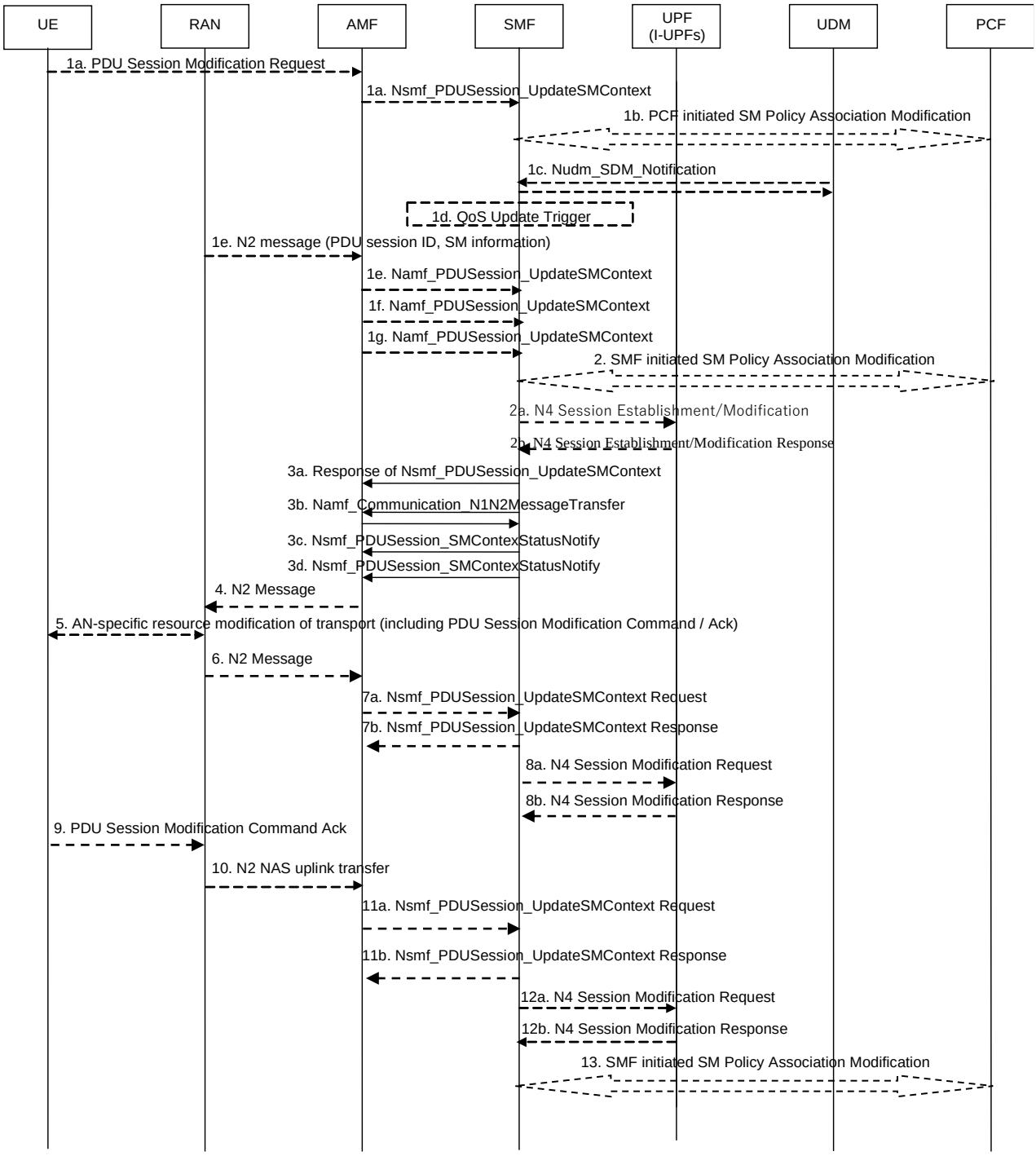

Figure 29: UE or network requested PDU Session Modification

9 CUMUCORE SLICE MANAGEMENT

This chapter describes network slice creation and management in the North Node facility (5GTN) using the Cumucore product [14]. As we are currently studying and determining the feasibility of Cumucore API in our implementation, we have not yet established exact workflows in our setup. Therefore, no flow diagrams on Cumucore integration are presented in this section. Flow diagrams explaining out how to integrate Cumucore in our setup will be presented in deliverable D2.3.

9.1 SLICE CREATION API

Cumucore has a Network Slicing REST API and a Graphical User Interface (GUI) that can be used to manually manage network slicing. Using the Network Slicing API the user can define slice sizes, quality parameters and traffic rules for each slice including priorities and pre-emption rules.

As described in D2.1 [2], the Cumucore slice creation API will be used in the 5GTN facility for the slice management by the NNA. The Cumucore Slice Creation REST API is described in detail in [14].

At the beginning of an experiment, the NNA creates requested slices (maximum of 2 slices) using the Cumucore slice creation API, and when the experiment is finished, the API is used to tear down the created slices. The slice creation API has been integrated into the current NNA implementation. However, the current implementation of Cumucore slice creation API cannot dynamically manage the core network slice parameters (i.e., latency, max Upload (UL) throughput per slice, UL threshold per slice, max throughput Download (DL) per slice, and DL threshold). Therefore, a different approach is used in managing slice parameters (latency and throughput) dynamically. The approach is described in the following section.

9.2 DYNAMIC SLICE MANAGEMENT

As the Cumucore slice creation API is not able to handle dynamic network parameter management, a UPF-based solution is used instead. In this approach, we use several pre-configured UPFs with different throughput capacities. Our setup is depicted in Figure 30. Whenever a slice needs higher throughput, a UPF with higher capacity is dynamically assigned to the slice. Likewise, when we increase throughput of a slice, in our 2-slice configuration, the throughput of the second slice must be reduced. This is done by assigning a UPF with lower throughput to the second slice.

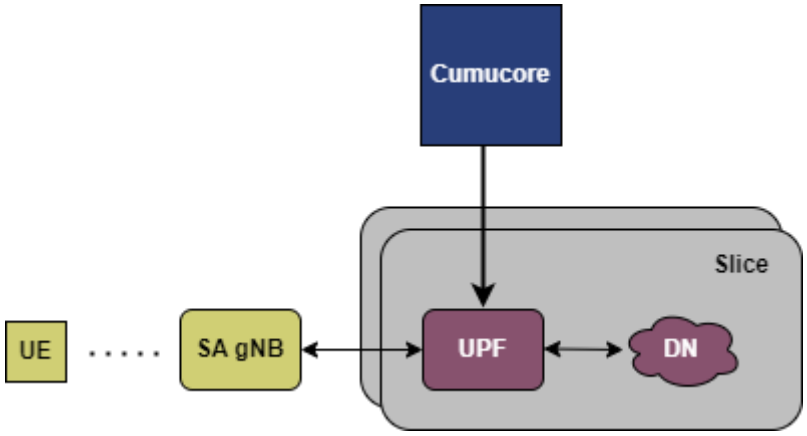

Figure 30: UPF-based slice management with Cumucore

Cumucore provides an API for changing UPF configuration, i.e. assigning it to a slice. The API endpoint is described in Table 7.

Table 7: UPF configuration API

Method	Path	Description
PUT	{apiRoot}//api/v1.0/cnc-config/upf-profile-update/{instance_id}	Update UPF configuration using the instance ID as a query parameter. Example: curl -k -X PUT https://192.168.9.120:3000/api/v1.0/cnc-config/upf-profile-update/fdc923f0-8602-40fd-a043-403fb97e9e85 -d '{"nfInstanceId":"fdc923f0-8602-40fd-a043-403fb97e9e85","nfInstanceName":"sandvik","nfType":"UPF","nfStatus":"REGISTERED","ipv4Addresses":["192.168.9.124"],"upfInfo":{"sNssaiUpfInfoList":[{"sNssai":{"sst":1,"sd":"000001"},"dnnUpfInfoList":[{"dnn":"ims"}]}],"interfaceUpfInfoList":[{"interfaceType":"N3","ipv4EndpointAddresses":["10.128.1.10"]}]}}'

10 OPEN-SOURCE 5G SOLUTION FOR SUSTAINABILITY FRAMEWORK: OAIBOX

As described in D2.1 [2], OAIBOX [15] will be an enabler in 5GTN facility for the energy measurement framework. The energy measurement framework enables real-time power consumption measurements of the OAIBOX (gNB+Core) and Universal Software Radio Peripheral (USRP) by changing the RAN configurations.

The enabler is developed and integrated in the North Node 5GTN facility. It facilitates the sustainability framework in the following ways:

- **Energy Measurements**: It measures energy consumption for OAIBOX MAX (gNB, core) and USRP. By changing RAN configurations, the energy consumption in different setups can be analyzed.
- **Restricting 5G NR Bandwidth**: Bandwidth can be adjusted between 20 MHz, 40MHz, 60 MHz, and 100 MHz. Reducing bandwidth can help optimize energy consumption and system performance.
- **Selecting Time-Division Duplex (TDD) Frame Structure**: Adjusting the balance between UL and DL slots impacts the energy efficiency of the system. Increasing UL slots can help conserve energy in certain scenarios.
- **Modulation Constellation Restrictions**: Modifying Modulation and Coding Scheme (MCS) for UL and DL allows for control over data rates and energy consumption. Options include Auto, QPSK, 16QAM, 64QAM, and 256QAM.
- **DL MIMO Mode Selection**: Enabling spatial multiplexing through different Multiple-Input Multiple Output (MIMO) configurations enhances data throughput. The system supports SISO (Single Input Single Output) (1x1), MIMO (2x1), and MIMO (2x2) configurations for varying energy efficiency and performance.

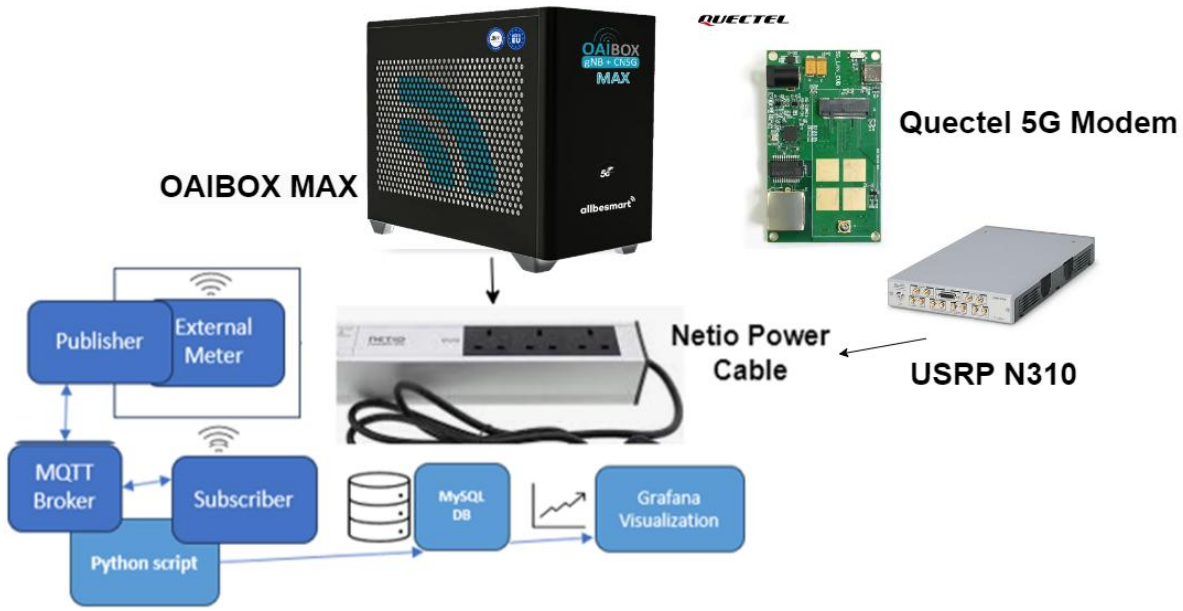

Figure 31: The OAIBOX integration with the energy measurement platform

The central controller is used to integrate the OAIBOX power measurements within the energy measurement framework using the external power meter (NetioPowerBox [5]) as shown inFigure 31.

Open source 5G (OAIBOX) power profiling is based on different RAN parameters for analysis the power saving measures. Using real time telemetry data (gNB logs/s), we can perform a systematic analysis to understand the relationship between network performance metrics and energy usage. The experimentation results will be reported in future deliverables.

11 CONTROL PLANE INNOVATIONS

In a 5G Network, while the media plane is responsible for transporting the actual information across the network, the control plane determines how the connection is managed, including orchestrating authentication, mobility, and resource allocation. It enables real-time decisions on routing and quality of service, ensuring low latency and efficiency, which is vital for real-time streaming applications.

11.1 FUNCTIONAL DESCRIPTION

Since the previous deliverable D2.1 [2], a stable connection has been successfully established between the 5Tonic environment and Ericsson's Cloud IP Multimedia Subsystem (IMS), which hosts the Data Channel Server (DCS). This connection is critical as it enables the seamless flow of signalling and data traffic between the 5Tonic testbed and the core IMS services, which are crucial for managing voice, video, and messaging services across the 5G network. The integration with Azure's cloud infrastructure ensures that the IMS services are scalable, resilient, and capable of supporting the high-performance demands of 5G applications.

The next step has been establishing a connection between the Ericsson Cloud IMS, which is integrated with DCS on Azure, and the Matsuko servers, also hosted on Azure. This connection is essential for delivering Matsuko's advanced services over the 5Tonic 5G network because it will allow the IMS to efficiently manage and route the signalling and media streams necessary for Matsuko's services, ensuring that they are delivered with the low latency and high quality expected in a 5G environment.

Interfaces from IMS Functions to the DC Application server are key, especially in B2B scenarios with Business Voice Architectures (interfaces MDC2 and DC4). These interfaces have not yet been defined in 3GPP Release 18 [16]. Figure 32 shows the up-to-date architecture required to implement UC3. The steps taken in the communication between the different network elements are described in the following subsections.

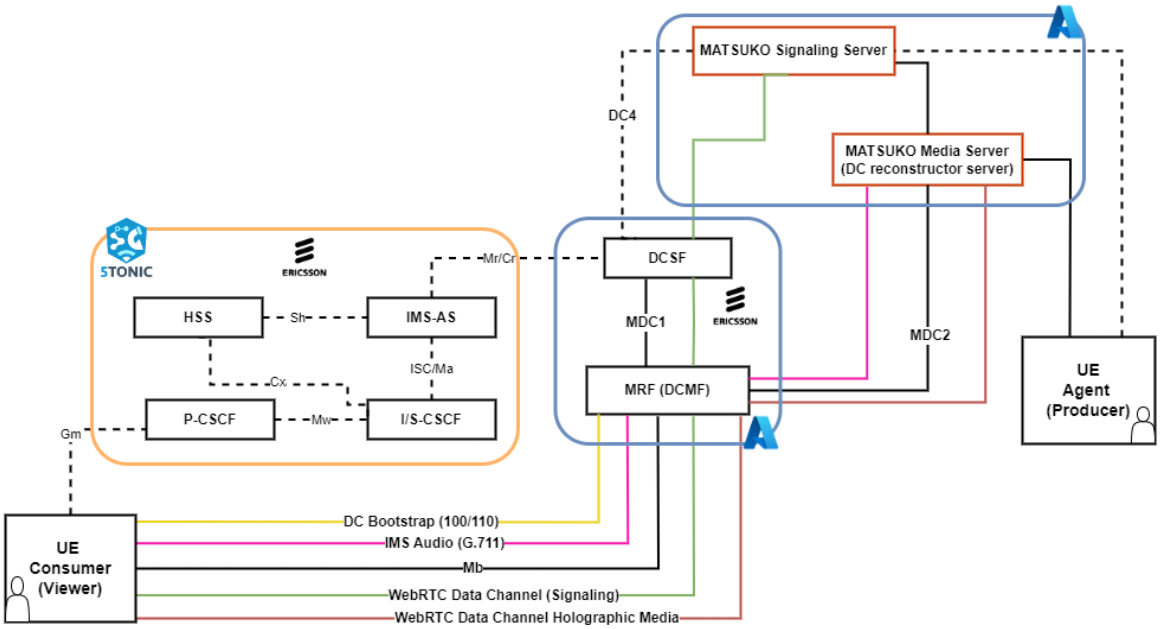

Figure 32: Updated UC3 general IMSDC architecture

11.2 COMMUNICATION DESCRIPTION

The following steps describe the message exchange between network elements required to establish holographic communication:

1. **DCS Registration in the IMS Network (Mr/Cr interface)**

   The DCS registers in the IMS network. The DCS identifies itself in the network with a Tel Uniform Resource Identifier (URI), which is essentially an address that other devices can use to find and communicate with it. For communication to be established, the hologram receiver device (a UE, such as a mobile phone or tablet) needs to "call" this URI to connect with the DCS. Generally, once the user has established the call with the DCS, the dialler of the mobile phone will display the DCS menu interface, allowing interaction and selection between the different stored services (see Figure 33).
   For the correct operation of UC3UC and to demonstrate the holographic call service, this system will be patched so the service automatically launches the Matsuko application, enabling the hologram of the producer to be displayed.

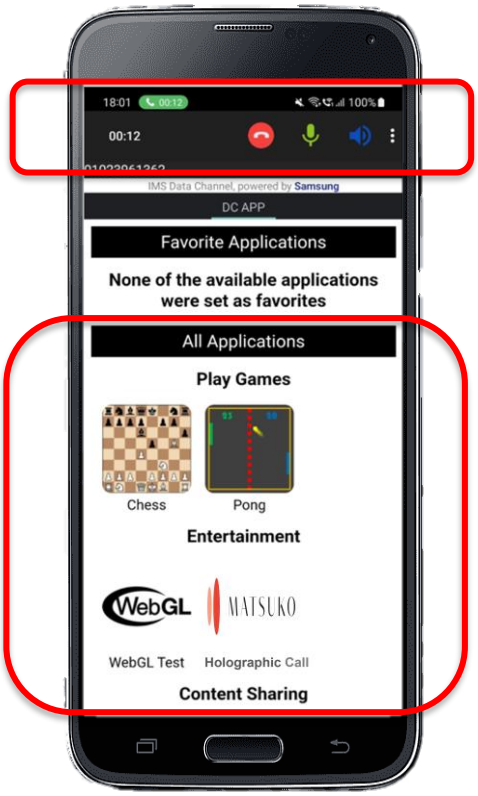

Figure 33: "Menu" view in DCS user interface

2. Signalling Messages (DC4 interface alternative)

When the UE initiates a call to the DCS, the latter establishes a WebSocket connection with a signalling server. The WebSocket enables real-time communication between a client (in this case, the DCS) and a server, which is crucial for applications that require low latency. The DCS sends important information to the signalling server via Uniform Resource Locator (URL) parameters.

Parameters:

- u - user ID
- r - room name/meeting ID
- l - GPS location
- o - MCC MNC of the operator

The address and port the signalling server are listening on are part of the DCS configuration. For this early phase of the proof of concept (PoC), the room ID, GPS location, and operator MCC/MNC are hard-coded. The user ID will be the holographic receiver's SIP/TEL URI. Upon establishing the WebSocket connection towards the signalling server, the DCS will send the required configuration parameters.

3. **Configuration Exchange**

   Once the connection with the signalling server is established, the DCS sends the necessary configuration parameters to enable communication. This includes details like Interactive Connectivity Establishment (ICE) servers, which help devices find the best route for transmitting multimedia data (such as holograms) over the network.

4. **Peer ID Exchange**

   The UE responds with a "peer ID," a unique identifier for that hologram session. This ID is used to link the data streams between different devices and servers.

   sending message (i.e.): *{"msg": {"peerId":"xxxxxxxx"}}*

5. **SDP Offer**

   The DCS sends an "SDP offer" to the signalling server. The Session Description Protocol (SDP) describes the parameters of the multimedia session, like the type of media to be transmitted (audio, video, data, etc.). At this point, the SDP offer only contains the basic configuration without details on how the data will be sent.

6. **ICE Candidate Exchange**

   Next, both the DCS and the UE start exchanging "ICE candidates." These candidates are potential routes through which data can flow. The candidates can be internal or external IP addresses, and the goal is to find the most efficient route for data transmission.

   received message (i.e.): *{"msg": {"peerId":"xxx","iceCandidate":"{\"candidate\": \"candidate: xxx x udp xxxx xx.xxx.x.x xxxxx typ xx generation xx ufrag xxxxx network-cost xx\"\"sdpMLineIndex\":x,\"sdpMid\":\"x\"}"}}*

7. **SDP Response from the UE**

   The UE sends its own SDP response, either accepting or modifying the DCS initial offer. After this, the ICE candidate exchange continues until both sides agree on the best route for communication.

8. **Data Channel Setup (MDC2 interface alternative)**

   The DCS and the reconstructor server (responsible for processing holograms in real-time) set up data channels. The DCS between the JavaScript application running on the mobile phone and the MRF has one more stream than the data channel between the MRF and the reconstruction server. The MRF redirects the first stream towards the DCSF and then to the signalling server. This redirection does not alter the exchanged messages and occurs after the successful SDP signalling. The MRF redirects the stream and upwards towards the reconstruction server in both directions.

11.3 WORKFLOW

This entire process illustrated in Figure 34 ensures that the UEs and the DCS can communicate efficiently, sending and receiving holograms in real-time with minimal latency. It's a complex procedure that involves network components designed to ensure that data is delivered quickly and reliably, which is essential for advanced applications like holography.

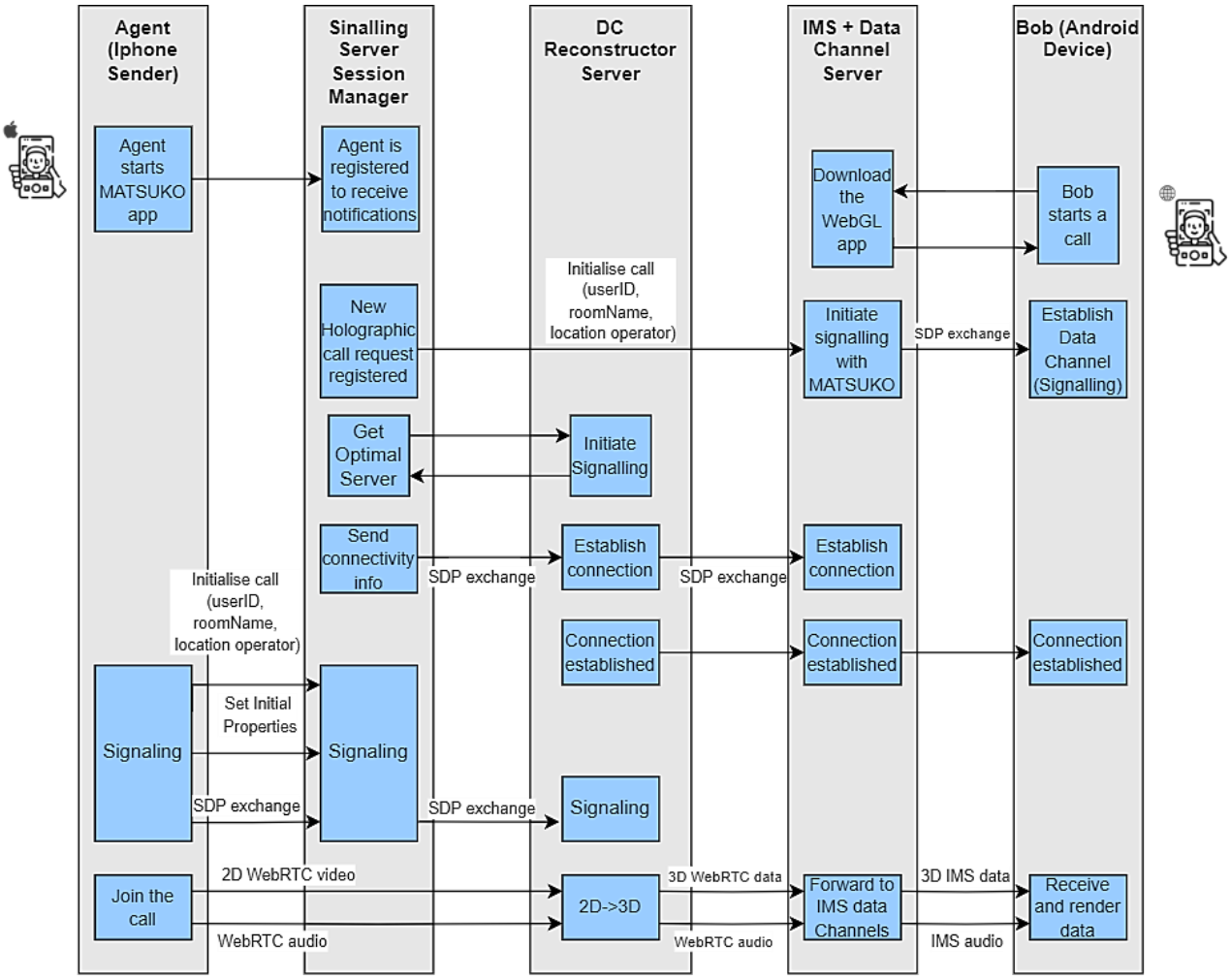

Figure 34: High level IMS DC network elements workflow

12 SUMMARY

This deliverable provides a comprehensive overview of the implementation of the core network enablers required for the validation of the UCs identified in D1.1 [1]. The deliverable highlights any deviations or modifications that have occurred since the initial planning phase in D2.1 [2] and reflecting on how the enablers are evolving during the development phase.

In addition to the enabler implementation, this deliverable presents an update on the infrastructure compared to the higher-level descriptions provided in D2.1. The project partners have detailed any new components added to the two test facilities in the South Node, located in Barcelona and Madrid, as well as updates to the infrastructure at the North Node. These updates provide the reader with an up-to-date view of the system architecture, reporting on the status of the experimental platform in its most recent configuration.

Alongside the technical descriptions of the enablers, the deliverable outlines the methods available through the respective APIs, which interactions facilitate interaction with these enablers. Accompanying these descriptions are detailed flow diagrams, offering a clear visualization of the operational workflows, providing a well-structured guide on how the enablers function, how they should be utilized in practice.

Most enablers have been developed in isolation, ensuring compliance with the relevant standards adopted by the project partners. However, as future work, these enablers will undergo integration, allowing them to interoperate and collectively provide the necessary validation for the use cases. This integration phase will be documented in detail in D2.3, where the combined performance and system-level functionality will be assessed.

13 REFERENCES

[1] "D1.1 Requirements and Use Case Specifications, 6G-XR project, June 2023.," [Online]. Available: https://www.6g-xr.eu/wp-content/uploads/sites/96/2023/10/D1.1-Requirements-and-use-case-specifications-V1.0.pdf..

[2] "D2.1: Orchestration, AI techniques, End-to-end slicing and Signalling for the core enablers – design, 6G-XR project, March 2024.," [Online]. Available: https://www.6g-xr.eu/wp-content/uploads/sites/96/2024/03/D2.1-Orchestration-AI-techniques-End-to-end-slicing-and-signalling-for-the-core-enablers-design-V1.0-2.pdf..

[3] "D1.3: Test infrastructure specification," [Online]. Available: https://www.6g-xr.eu/wp-content/uploads/sites/96/2024/09/D1.3-Test-infrastructure-specification-V1.0-1.pdf.

[4] "D5.1: Description of sustainability experimentation framework, September 2024.," [Online]. Available: https://www.6g-xr.eu/wp-content/uploads/sites/96/2024/09/D5.1-Description-of-sustainability-experimentation-framework.pdf.

[5] "Netio PowerBOX 4Kx," [Online]. Available: https://www.netio-products.com/en/device/powerbox-4kx.

[6] "5G!Drones, Unmanned Aerial Vehicle Vertical Applications' Trials everaging Advanced 5G Facilities, EU H2020 ICT-19-2019 Project, https://5gdrones.eu," [Online]. Available: https://5gdrones.eu.

[7] Kaitotek, "REST API Documentation (Qosium Storage)," [Online]. Available: https://www.kaitotek.com/resources/documentation/storage/user-interface#tabs-information.

[8] "Qosium," [Online]. Available: https://www.kaitotek.com/qosium.

[9] "ETSI MEC.," [Online]. Available: https://www.etsi.org/technologies/multi-access-edge-computing.

[10] "CAMARA," [Online]. Available: https://camaraproject.org/.

[11] "GSMA Official Document OPG.04 - East-Westbound Interface APIs, Version 3.0, 26 July 2023," [Online]. Available: https://www.gsma.com/futurenetworks/wp-content/uploads/2023/07/OPG.04-v3.0-GSMA-Operator-Platform-Group-East-Westbound-Interface-APIs-Version-3.0.pdf.

[12] "D4.1: State-of-the-art analysis and initial design of beyond 5G RAN, core, and open-source networks, disruptive RAN technologies and trial controller," [Online]. Available: https://www.6g-xr.eu/wp-content/uploads/sites/96/2024/09/D4.1-State-of-the-art-analysis-and-initial-design-of-beyond-5G-RAN-core-and-open-source-networks-disruptive-RAN-technologies-and-trial-controller.pdf.

[13] "3GPP Technical Specification 23.502 17.0 , "Procedures for 5G system", 3GPP TS 23.502 description," [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/17.05.00_60/ts_123502v170500p.pdf.

[14] "Cumucore 5G NC Specifications, Cumucore, 2023, "Cumucore_5G_NC_Product_Specifications_2023_March.docx.pdf"," [Online]. Available: https://cumucore.com/mobile-private-network/.

[15] "OAIBOX, Open source 5G/6G network in a box.," [Online]. Available: https://www.oaibox.com/.

[16] 3GPP, "3GPP documentation Release 18," [Online]. Available: https://www.3gpp.org/specifications-technologies/releases/release-18.