FUDGE-5G

FUlly DisinteGrated private nEtworks for 5G verticals

Deliverable 4.2

Final Technical Validation of 5G Components with Vertical Trials

Version 1.0

Work Package 4

Main authors	Luís Cordeiro, João Fernandes, André Gomes (ONE)
Distribution	Public
Date delivered	06/03/2023 (M31)

Partners

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Nº 957242.

Disclaimer

Project details

Project title:	FUlly DisinteGrated private nEtworks for 5G verticals
Acronym:	FUDGE-5G
Start date:	September 2020
Duration:	30 months
Call:	ICT-42-2020 Innovation Action

For more information

Project Coordinator
Prof. David Gomez-Barquero
Universitat Politecnica de Valencia
iTEAM Research Institute
Camino de Vera s/n
46022 Valencia
Spain

http://fudge-5g.eu
info@fudge-5g.eu

Acknowledgement

FUDGE-5G has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement Nº 957242. The European Union has no responsibility for the content of this document.

Abstract

This document describes the efforts towards the validation of 5G components with vertical trials. In D4.1 was described a common validation methodology, followed by a description of the trial's validation for each use case during the first phase of the project. In D4.2, following a different approach, the document goes further into detailing the validation of each 5G component.

Contributions and reviewers

Contributors

Partner	Authors
ONE	Luís Cordeiro, André Gomes, João Fernandes
UPV	Carlos Barjau, Borja Iñesta, Josep Ribes
THA	Franck Scholler
5CMM	Manuel Fuentes, Alberto Beltrán, David Martín-Sacristán
FHG	Pousali Chakraborty, Hemant Zope, Marius-Iulian Corici
IDE	Sebastian Robitzsch, Mohamad Kenan Al-Hares
O2M	Peter Sanders
ATH	Daniele Munaretto, Nicola di Pietro
CMC	Mika Skarp
UBI	Thanos Xirofotos

Reviewers

Reviewer	Partner

Executive Summary

FUDGE-5G aims to perform field trials for the validation of its platform in five vertical use cases, which should target a Technology Readiness Level (TRL) of 7 or above, i.e., system prototype demonstrations in an operational environment. After reporting the interim technical validation efforts in D4.1, this deliverable (D4.2) aims to report on the final technical validation efforts of FUDGE-5G components, both in the field and lab environments.

Each use case organized trials involving prominent stakeholders and set up the components to obtain tangible results. Hence, for each trial, the methodology involved the collection of KPIs from all the 5G components while obtaining relevant feedback from the stakeholders, which combined provide both means for the technical validation of the 5G infrastructure and to conduct a gap analysis between the 5G measured performance and the technical requirements and KPIs stemming from the vertical use cases (use case validation).

The contents of this document are driven towards the validation of each 5G component, while D4.3 will give further detail on the validation of the platform in a trial context.

Abbreviations

5G	5th Generation of mobile communications
5GC	5G Core
AF	Application Function
AMF	Access and mobility Management Function
API	Application Programming Interface
AUSF	Authentication Server Function
E2E	End to End
gPTP	generic Precision Time Protocol
LAN	Local Area Network
NEF	Network Exposure Function
NF	Network Function
NOW	Network on Wheels
NPN	Non-Public Network
NSA	Non-Standalone
PLMN	Public Land Mobile Network
PNI-NPN	Public Network Integrated-NPN
PTT	Push To Talk
PPDR	Public Protection and Disaster Relief
RAN	Radio Access Network
SBC	Session Border Controller
SCP	Service Communication Proxy
SEPP	Security Edge Protection Proxy

SFVO	Service Function Virtualization Orchestrator
SMF	Session Management Function
SA	Stand-Alone
SH	Service Host
TRL	Technology Readiness Level
TSi	Ingress Timestamping
TSN	Time Sensitive Networking
UC	Use Case
UDM	Unified Data Management
UE	User Equipment
UPF	User Plane Function
VM	Virtual Machine

Table of contents

List of Figures

List of Tables

1. Introduction

FUDGE-5G is split into 5 vertical Use Cases (UCs), which aim to validate the technology innovations brought by the FUDGE-5G platform. These use cases follow a realization methodology, which is depicted in Figure 1.

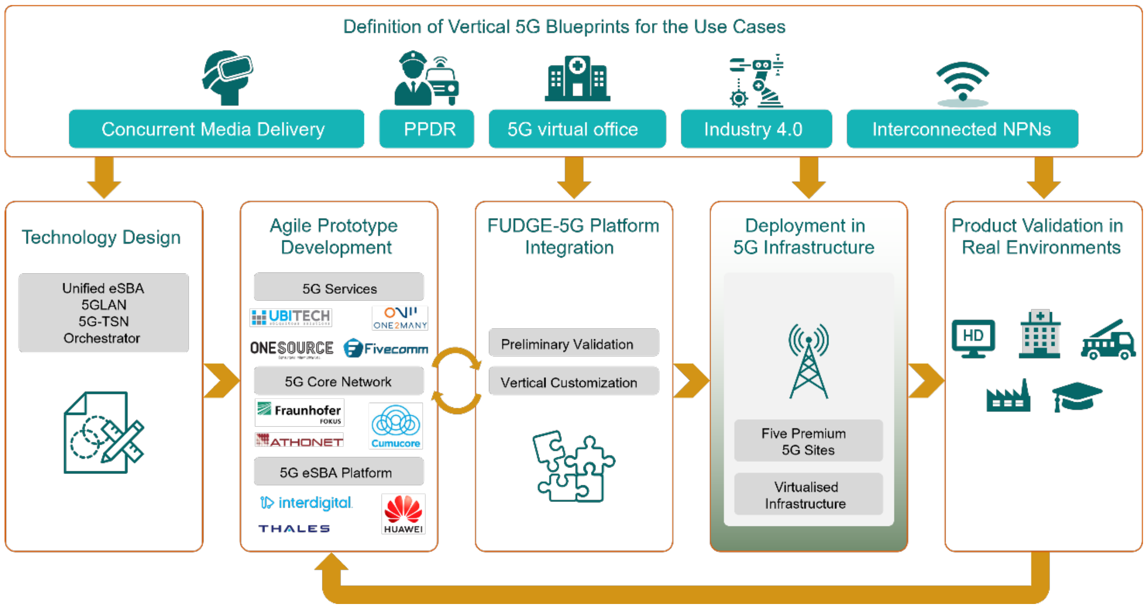


Figure 1: FUDGE-5G use case validation methodology

This deliverable focuses on the technical validation of each 5G component in trial and lab contexts. These are divided into five different sections:

- FUDGE-5G Platform
- 5G Cores and integrated microservice based Network Functions
- 5G Devices
- FUDGE-5G Innovations

2. FUDGE-5G Platform

The technical validation of the FUDGE-5G platform components, introduced in this section, focuses on providing a numerical assessment of the routing and orchestration functionality of the eSBA platform. Figure 2 illustrates the infrastructure and platform topology of the staging testbed, which is used for integration and validation purposes.

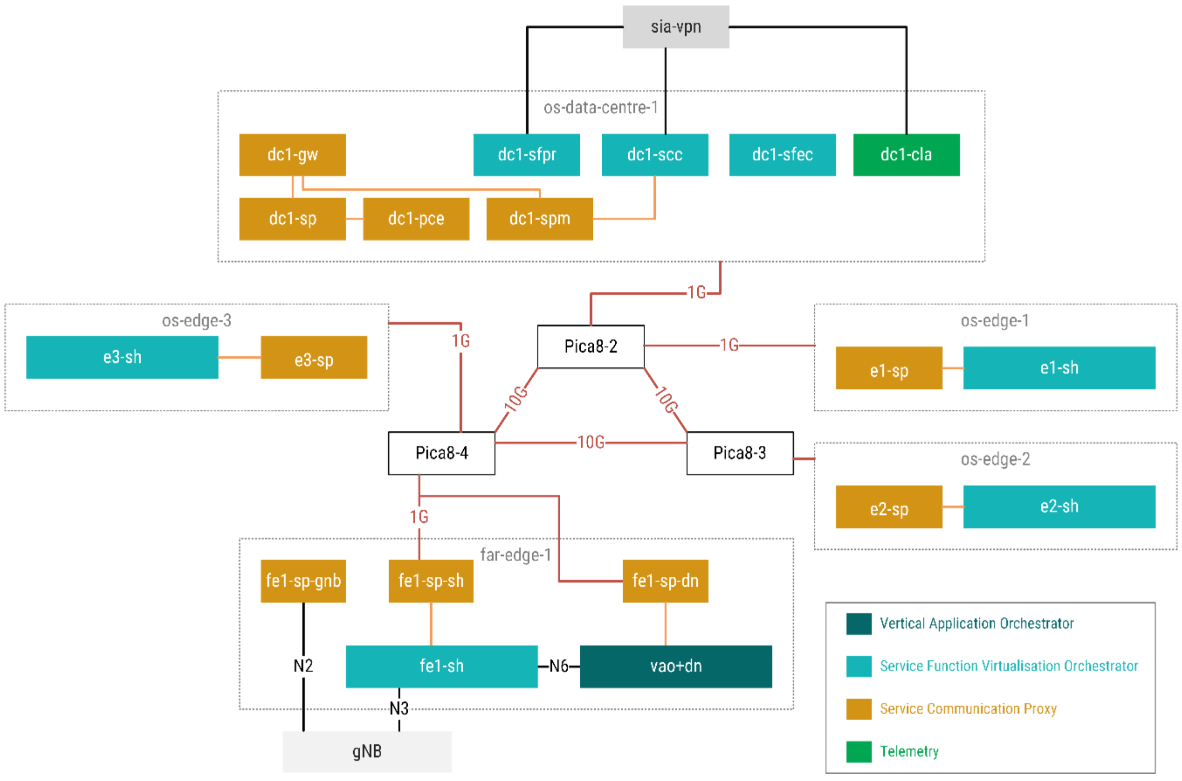


Figure 2: Topology of Staging Testbed with FUDGE-5G Platform Components

The dotted grey boxes represent OpenStack-managed compute hosts, except for the node “far-edge-1” that represents a LXC host machine. The compute hosts are interconnected by hardware-based Pica8 SDN switches that offer 1G ports towards the compute hosts and 10G interfaces among each other. This results in a star-like topology with a triangular shape at the core.

Each coloured box in the figure stands for an instance of a platform component. For OpenStack-based coloured boxes, these are Virtual Machines (VMs). For the ones on the “far-edge-1” compute host, these are LXC containers. The grey box at the bottom is a commercial off-the-shelf gNB from Amarisoft. At the top of the figure, the grey box “sia-vpn” offers secure remote access for experimenters (FUDGE-5G partners) to orchestrate their services.

The next two subsections will provide an analytical assessment of the platform’s routing and orchestration performances in the staging testbed.

2.1. Orchestration

The assessment of the routing component of the FUDGE-5G platform, the Service Communication Proxy (SCP), is going to focus on latency and throughput. As mentioned in the introduction, the SCP leverages and SDN-underlay in the infrastructure composed of hardware SDN switches and utilises Open-vSwitch inside the Service Proxies (SPs) where IP traffic is terminated and translated into NBR-internal semantics (i.e. Information-Centric Networking). As a result, when querying the SDN controller (Floodlight in this case), the topology is reported, as illustrated in Figure 3. Floodlight natively reports latencies between switches which are provided as integer numbers in the figure below.

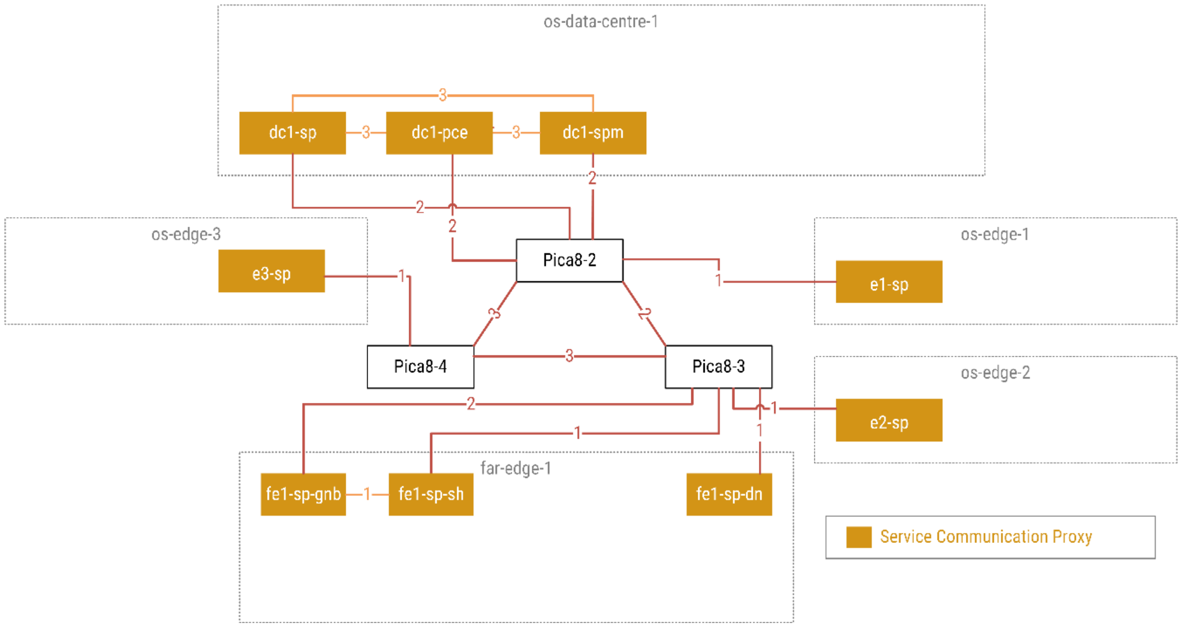


Figure 3: Latencies Among Service Communication Proxy Platform Components in Milliseconds

For the end-to-end latency analysis conducted further below, it is important to assess how much latency is created by the SDN underlay and how much latency is created by the SCP. In average, the latency numbers do not exceed 3ms and there is a number of observations possible from the figure above, i.e.:

- The OpenStack compute hosts os-edge-* are installed in the same server rack as the SDN hardware switches (pica8-*). The latency between the hardware switches and the SCP VMs OpenStack (*-sp) is always 1ms.
- The os-data-centre-1 OpenStack compute host is located in the lab environment on another floor at InterDigital premises in London, resulting in larger cable lengths and an added latency resulting in 2ms between the SDN switches and the OpenStack compute hosts.
- The latency between OpenStack VMs in os-data-centre-1 are rather high at 3ms caused by OpenStack's virtual networks.

The end-to-end latency analysis is going to assess the connections from any given Service Host (SH) towards the GW. Table 1 provides the resulting latency numbers as a sum of the links between SCP-components, as reported by Floodlight.

Table 1: Inter-SCP Component Latencies, as Reported by the SDN Controller Floodlight

Service Host	SCP Topology	E2E Latency
e1-sh	e1-sp > pica8-2 > dc1-sp	3
e2-sh	e2-sp > pica8-3 > pica8-2 > dc1-sp	5
e3-sh	e3-sp > pica8-4 > pica8-2 > dc1-sp	6
fe1-sp-sh	fe1-sp-sh > pica8-3 > pica8-2 > dc1-sp	5

The end-to-end latency analysis used ICMP Echo Request/Response messages from each Service Host (SH) towards the GW. While SHs are located on the OpenStack compute hosts os-edge-1, os-edge-2, os-edge-2 and the bare metal one far-edge-1 (see Figure 2), the GW was the dc1-gw VM on the OpenStack compute host os-data-centre-1. The results of the ICMP tests are provided in Figure 4 and have the ICMP endpoints and direction as labels of all five tests.

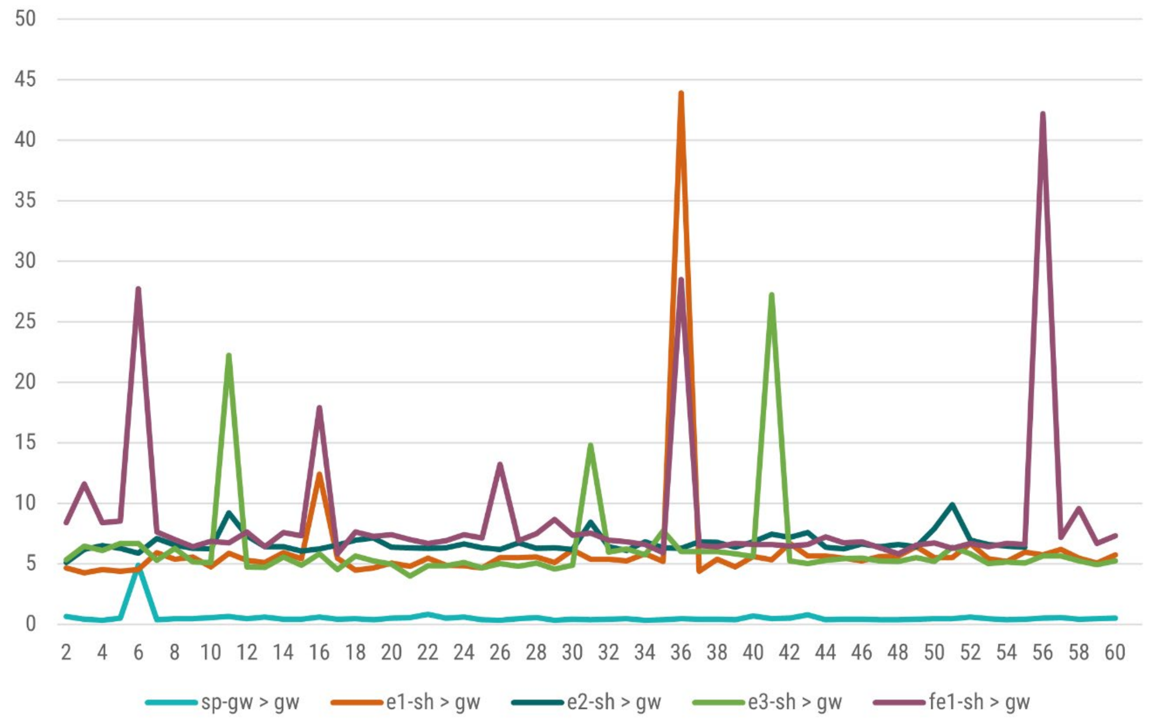


Figure 4: Latency Analysis of the Platform's Service Communication Proxy Across Locations

As can be observed, the baseline measurement between the SCP and the GW (using a standard IP-based virtual network in OpenStack) stays expectedly low and solid at below 1ms. The other measurements that traverse the SCP and the SDN underlay stay close to the latency numbers reported by the SDN controller, allowing to conclude the SCP does not add more latency to the communication. Having said this, there is occasional outliers in the measurements pointing at an artifact of the implementation of the SCP, i.e. a user space application that performs packet manipulation resulting in the need to configure a generic Linux kernel to permanently allow such activity in user space. It has been observed that by default all user space application (in a generic Linux operating system) are treated as best effort and specific mechanisms are implemented inside the Linux kernel prohibiting user space applications to receive CPU “airtime” whenever they need. The reason for this is to allow a better experience to users when running their programs, as no other application would be allowed to allocate 100% of a CPU. As a result, the SCP components have been configured to be operated in a round robin fashion by the kernel where all SCP user space processes will receive their equal CPU times in a round robin fashion. As logging (journald) was added to this process allowing the SCP to log at info level to catch any unexpected behaviour when testing Enterprise Services, the measured outliers may be a result of that.

Table 2: End-to-End Latency Measurements for the Service Communication Proxy

Connection	Mean [ms]	Min [ms]	Max [ms]	StdDev [ms]
sp-gw > gw	0.5	0.3	4.9	0.57
e1-sh > gw	6.1	4.3	43.9	5.04
e2-sh > gw	6.9	5.1	9.9	2.23
e3-sh > gw	7.2	4.0	27.2	8.46

Next was a throughput analysis using the same links as for the latency measurements. Figure 5 provides the results of the TCP measurements obtained using iperf3. To put the resulting numbers into perspective, the first focus should go the standard IP-based communication between two OpenStack VMs, i.e. sp-gw > gw. Knowing about the physical capabilities of this virtual network (i.e. theoretically the fastest two physical CPU can operate allowing two iperf user space application to exchange TCP-based data), the resulting mean of 475 Mbps seems rather low. This can be explained by the fast that checksum offloading had to be disabled, as the SCP operates in user space and the underlying virtualised NIC would often get the checksum wrong for both UDP and TCP. Turning off checksum offloading essentially tells the Linux kernel to calculate it instead of the NIC doing it. As checksum offloading is a key feature to boost performance of communication of COTS hardware, the effect of turning it off can be observed in the obtained data. Note that the sp-gw > gw throughput numbers do not involve any SCP user space process and is standard iperf on two VMs.

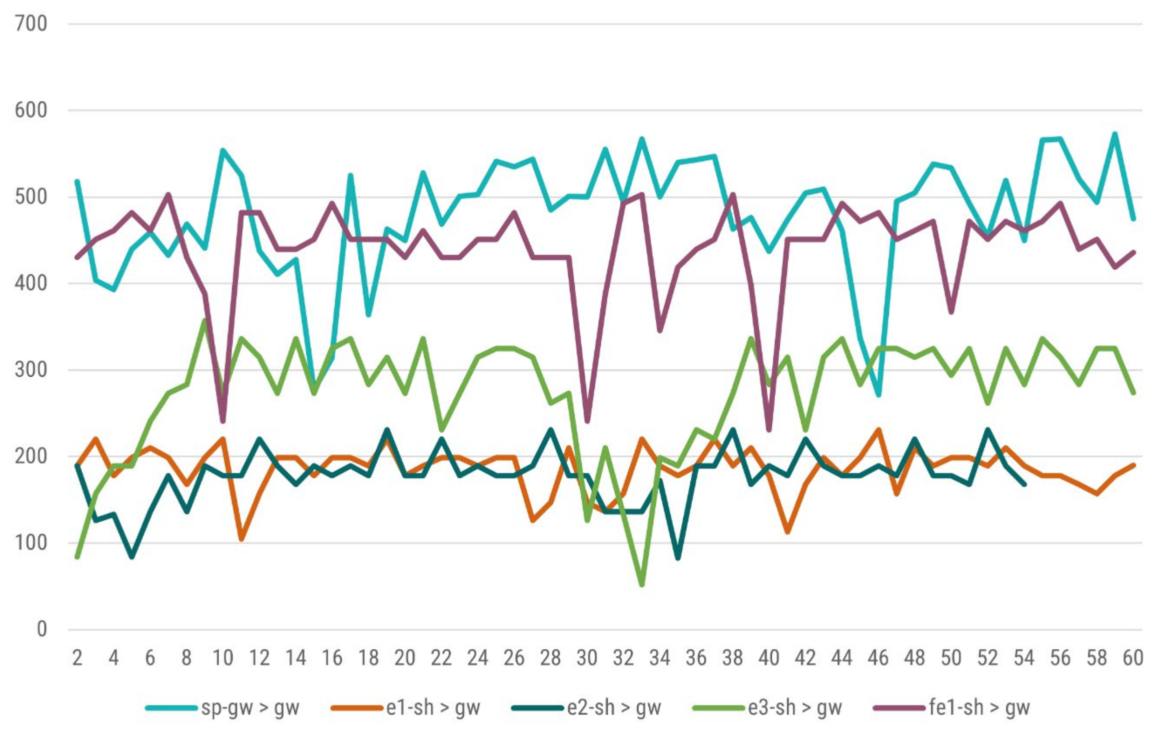


Figure 5: TCP Throughput Analysis of the Platform's Service Communication Proxy Across Locations

The other TCP throughput numbers in Table 3 allow the conclusion that with checksum offloading disabled and 1/3 of the time logging is taking CPU time in the round-robin-based execution of SCP actions, the throughput numbers are reasonably well positioned against the standard IP-based communication between sp-gw > gw. Most notably, the measurements for fe1-sh > gw demonstrates how well the SCP performs compared to standard IP communication, by being 38Mbps short on average. This can be explained by no fully virtualised Linux kernel in place on the fe1-sp-sh SCP component; instead a Linux Container is being used which has direct access to the Linux kernel on bare metal (checksum offloading still disabled). Table 3 provides the detailed numeric numbers to what has been illustrated in Figure 5.

Table 3: End-to-End TCP Throughput Measurements for the Service Communication Proxy

Connection	Mean [Mbps]	Min [Mbps]	Max [Mbps]	StdDev [Mbps]
sp-gw > gw	475	271	573	71.72
e1-sh > gw	190	105	231	27.59
e2-sh > gw	275	52	357	67.00

e3-sh > gw	182	83	231	31.78
fe1-sh > gw	437	231	503	61.87

2.2. Validation Results

This section presents a numerical assessment of the Service Function Virtualisation Orchestrator (SFVO). In detail, the time to orchestrate a Service Chain is measured based on the number of Service Functions that form the entire chain. As described in D1.3 [1], the Service Function Virtualisation Orchestrator (SFVO) offers the ability to orchestrate LXC, KVM and Docker containers across a set of Service Hosts (SHs) (aka locations). For the experiments conducted, LXC-based Service Functions were prepared with a Debian 10 base image and the bare minimum of additional packages installed, as per the packaging environment provided to Enterprise Service developers.

The resulting LXC-based SF was then orchestrated into one, two and three Service Hosts and the time was measured to issue the orchestration request up to the point the SFVO reported that the Service Chain had been successfully provisioned. The resource descriptor for this experiment is provided below. The orchestration over n locations was conducted three times each and whiskers boxplot representation of the results was chosen to illustrate the mean, variance and non-existing outliers.

The obtained numbers allow to conclude that the orchestration of a single LXC-based SF into a single Service Host takes 1min:20s, a single SFs into two SHs 1min:48s, and a single SF into three locations 1min:57s. Even though the SFVO orchestrates in parallel across locations, there is some serialised procedures in that implementation, resulting in additional time required if a new SH is added to the resource descriptor.

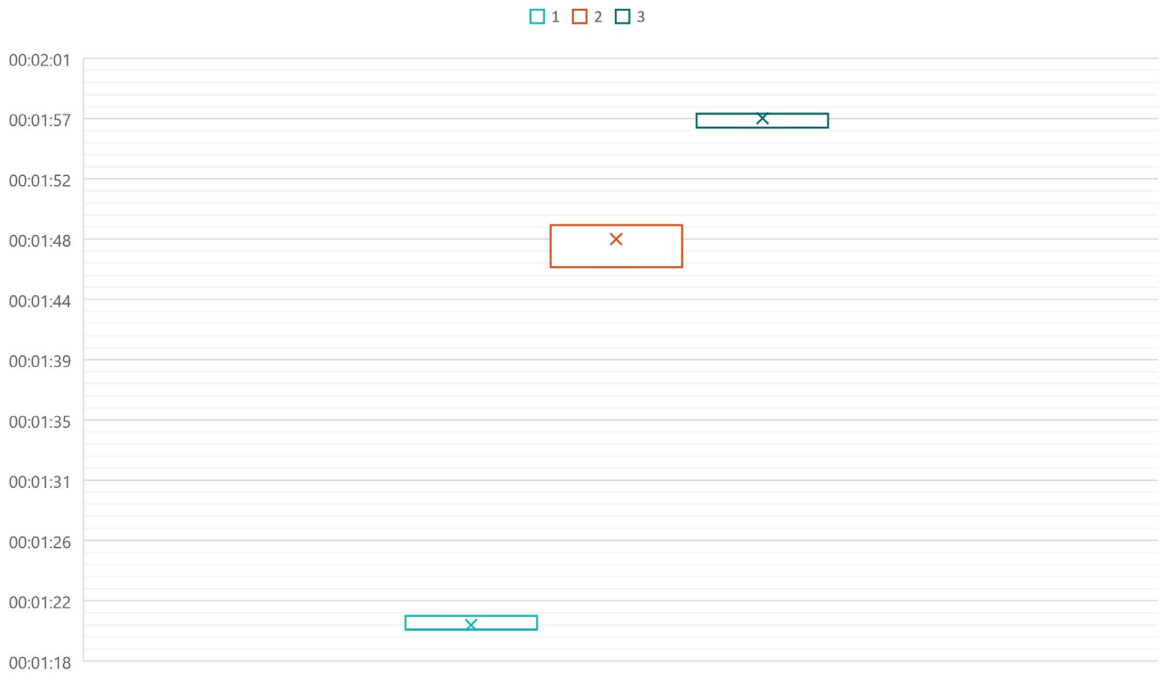


Figure 6: Orchestration Times over Number of Service Functions (LXC based)

```
meta:
  definition_version: "1.0"
  service_chain: "ide"

service_functions:
  - name: "sf-test"
    identifiers: ["sf-test.test"]
    sfp_url: "http://sfpr:8080/sfp/download/sebastian/test.lxc.tar.gz"
    instance_manager: "lxc"
    compute: 1
    memory: 250
    storage: 500

provisioning:
  - service_function: sf-test
    service_host: e1-sh
    state: "connected"
    instances: 1
  - service_function: sf-test
    service_host: e2-sh
    state: "connected"
    instances: 1
  - service_function: sf-test
    service_host: e3-sh
    state: "connected"
    instances: 1
```

3. 5G Cores

The following subsections describe the validation of the three 5G Cores integrated in the multiple testbeds and trial sites of the FUDGE-5G project.

3.1. ATH

Athonet provides its 5G Core network (5GC) technology for the i) PPDR and ii) Media Showroom use cases of FUDGE-5G. The flavour of the solution deployed in the two use cases slightly differs as follows:

- PPDR use case: full 5GC installed in ruggedized servers for mission critical usage, installed in the project's Network on Wheel (NoW), and especially adapted to on-demand standalone private network deployments in PPDR scenarios.
- Media showroom use case: 5G Control Plane (CP) functionalities deployed in the public cloud (namely, AWS) and integrated with the User Plane Function (UPF) of the consortium partner Cumucore, installed locally at their premises, in a hybrid deployment with UPF at the edge that realistically represents media-related scenarios. More details can be found in D3.2.

Concerning the 5GC deployed for the PPDR use case, Athonet has continuously supported Telenor and the related stakeholders, such as the Norwegian Defence, on enhancing the robustness and the performance of the solutions provided during the project lifetime. The strict requirements of PPDR scenarios imply that not only the 5GC software (SW) is programmed to include self-healing and resilient capabilities, but also the HW hosting the SW needs to be lightweight, robust, and resistant. So, one of the successfully achieved objectives of the work carried out in this context, was to test and evaluate the effectiveness of the ruggedized hardware (HW) selected for this peculiar 5GC instantiation scenario, with direct validation feedback from stakeholders that regularly operate in the PPDR field. Over time, it was decided to bring to this use case more than one 5GC solution, installed over different HW (as described below), but all provided solutions were chosen to meet the robustness and resiliency requirements. The experience that Athonet has gained over the years in the public safety and mission critical arena has definitely been key in deciding what combination of HW and SW to bring to the consortium specifically for this use case.

We can summarize the list of solutions (provided sequentially) and tested on field (also in public events and trials) as follows:

1. 5GC installed in a small ruggedized server with 1Gbps interfaces;
2. 5GC installed in the Snowcone product by AWS for edge deployments;
3. 5GC installed in a 10Gbps-interface ruggedized server.

All the above solutions were installed and integrated in the Network on the Wheel (NoW) and tested successfully by the partner and use case champion, Telenor, and the involved stakeholders (i.e. Norwegian Defence, NATO, AWS, etc.).

A further key validation test for Athonet's 5GC concerned the effective support and execution of mission critical applications over a real-life private 5G network for PPDR use cases. Therefore, Athonet provided to the project a mission critical application (MCX, as defined in the 3GPP standard [2] from one of its third parties to allow the PPDR stakeholders to test on field the set of capabilities that such application can bring to the first responders and special forces, such as instant messaging, man-down, group chat, voice and video calls, image sharing (as can be seen in Figure 7, where video call between a group is in progress). Such 3GPP application requires specific integrations with the 5G core network as per the standards, i.e., via the N5 interface from the Application Function (AF), i.e., the MCX application, to the 5GC's Policy Control Function (PCF), which only few solutions in the market can support nowadays. This capability is crucial and differs from common over-the-top applications, as it allows the end users to have their own traffic prioritized and with guaranteed quality over best-effort traffic in the 5G mobile network. The validation of the policy control operations via Athonet's exposed N5 interface was successful, and it constitutes a relevant step in the path towards the full inclusion of the same feature in Athonet's commercial deployments.

For what concerns the Media Showroom use case, Athonet provided the CP network functions of its 5GC in a cloud instance in the public cloud of AWS and integrated with the UPF provided by Cumucore and installed locally at their premises, at the edge compared to the CP, close to the Radio Access Network (RAN). The integration steps are described in D3.2 [3].

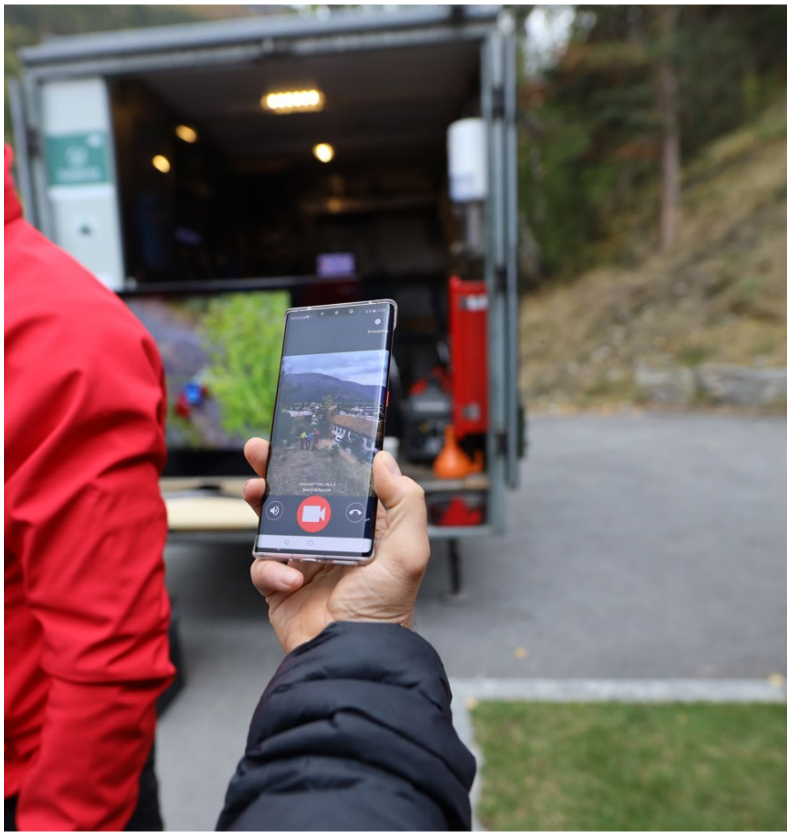


Figure 7: Group call via NEMERGENT PTT app

3.2. CMC

5G Core has a user management service where allowed users of the network are defined. User definition has also information about services users are allowed to use. For example, access to the network slices is predefined. This communication happens over N1 interface as shown in diagram below.

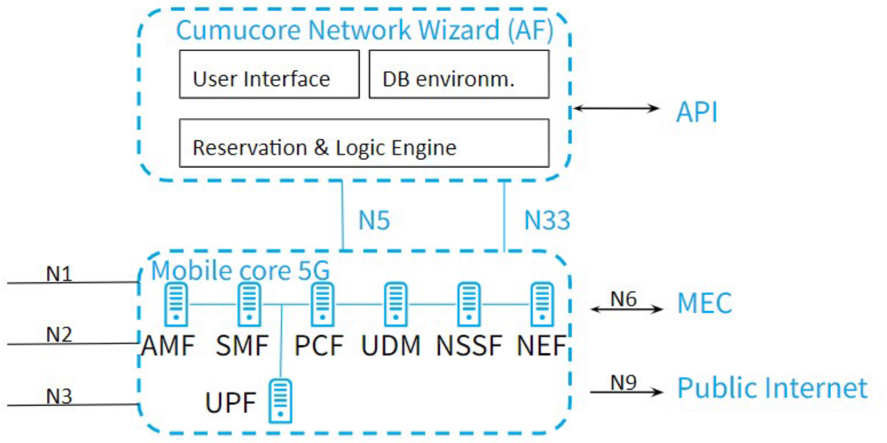


Figure 8 Cumucore architecture

During the attachment process 5G Core will ensure that the user has been granted an access right to use the network. In the end of the attachment process active data flow between User through Network to Internet is established using Interfaces N3 and N9

Interface N2 is used for signalling between User, Radio Access Network and Core Network during the active data flow. User Equipment delivers for example measurement results over N2 interface to AMF and receives for example radio channel power settings over this interface.

Mobile core network is organized into two network slices in the other words two separate virtual networks. One network slice is used for communication between UE and Internet and the second slice is for audio traffic between UE and Local audio processing server. Mobile Network core slices have different capacity and latency capabilities. Network slice has its own UPF and can also have other own network functions if needed. Inside the network slice there are user profiles that can be network slice specific. User profiles are defined under Mobile Network Core. User profiles are attached to data flows to differentiate Quality of Service levels. User profile includes:

- Connection type: Guaranteed bit rate / Nonguaranteed bitrate
- Max capacity downlink / uplink
- Pre-emption capability & vulnerability
- Quality of Service class
- Traffic Priority classifier

Network Slicing Manager is responsible for creating data flows per request. Network Slice Manager ensures that network slices can meet the promised KPI for all active and reserved upcoming data flows. Specific data flows can be requested outside of the network. In this use case specific data flows are for microphones and in-ear monitors. There is an open API to receive data flow requests. Data flow request includes flowing parameters:

- Source and destination IP address
- Requested profile
- Downlink and Uplink bitrate
- Time that data flow is requested (start & end time)

Network Slicing Manager has information of all active data flows, how much resources they are consuming and the cell they are in. Network Slicing manager has information about users' real time radio environment and location through NEF function. Based on this information Network Slicing manager will decide if requested data flow can be served by the network slice without causing any KPI violation to existing or upcoming reserved data flows.

New data flow requests will be sent over NG5 interface to PCF that will set up a data flow and send requests to Radio network over N2 interface. After all settings in the Radio and Core networks are done, the user is informed over the N1 interface to start using new data flow for a specific application. Audio processing will happen locally next to the 5G Core. The interface between Network Core and Audio processing computer is N6.

Network slicing manager has an user interface where the operator can see every application's resource utilization level. This information can be used to improve overall efficiency of the network.

During FUDGE project Cumucore core has been added 5GLAN functionality with TSN features in Industry4.0 use case. Cumucore has also added cell broadcast capability as demonstrated in in the public safety use case.

3.3. FHG

Fraunhofer FOKUS Open5GCore testbed is used for deploying private networks in two use cases within FUDGE- Hospital use case and Interconnected NPNs use case. In the Hospital use case the 5G core is responsible to forward the monitored data to vertical application whereas in Interconnected NPNs, the roaming UEs need to be authenticated and provided with both local and home network services by the 5G core network. Depending on the requirements, 5G core components were selected and deployed for these two use cases.

In the Hospital use case, the 5G Core was deployed on bare metal HPE DL110 telco server. Here all the 5G core components were deployed on the host machine without using network namespaces or any virtualization technique to achieve higher performance. The high level 5G core architecture is illustrated here using Figure 9. The AMF and UPF both are attaching directly to eth0 interface to receive N2 and N3 connection from RAN (NOKIA in this case). The OneSource Mobi-trust vertical application was also hosted in the same server and the data traffic is routed via the N6 interface to the application.

To validate the use case, from the 5G core it was checked if the registered active devices have successfully established data bearers or not. Tests were performed to validate the throughput of the data path. During the trial, while sending data to the application bandwidth was monitored and captured from the core network, which will be reported in D4.3 [4].

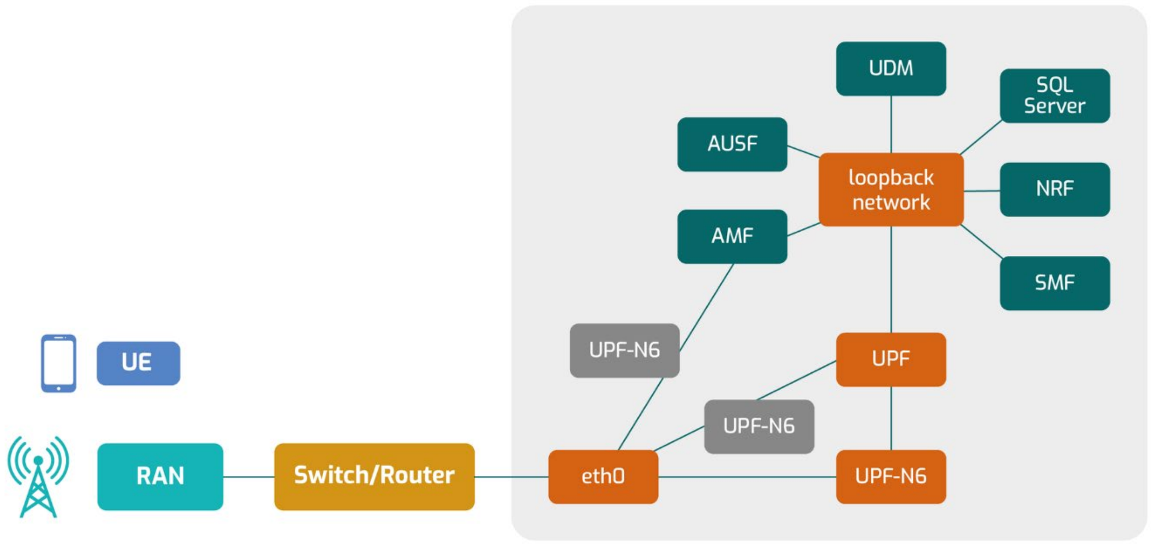


Figure 9: 5G Core deployed for 5G Virtual Office use case

In the Interconnected NPNs use case, the main requirement is to showcase roaming between private networks as mentioned earlier in this section. To illustrate that, 5G small-scale private networks were deployed in three geographically distant locations Berlin, Valencia and Oslo with 5G core components along with a newly developed component for this use case SBC (in more detail in section 5.7). The high-level architecture is shown in Figure 10. In each of the locations a lightweight containerized deployment was followed to run the 5G core components. The 5G core running in the three locations were configure with different PLMNs. Wireguard was used to interconnect the three nodes in order to have secure control plane message exchange between the domains and to forward the data traffic securely.

To validate the features of the use case, emulated UEs and Benchmarking Tool within the Open5GCore platform were used. They were configured with different PLMNs to test the home and visited subscriber scenarios. Different 5G procedures like registration, PDU session creation and deregistration were initiated for both home and visited subscribers. The duration of the procedures was collected and compared to check the overhead introduced in roaming. Also, the overhead of message exchange through SBC was captured. The data path capacity was also tested in cases, where data traffic is offloaded via local data network to the data network (local breakout) and where data traffic is offloaded via the home network to the data network (home-routed). All these captured values and their explanations will be reported in deliverable D4.3 [4].

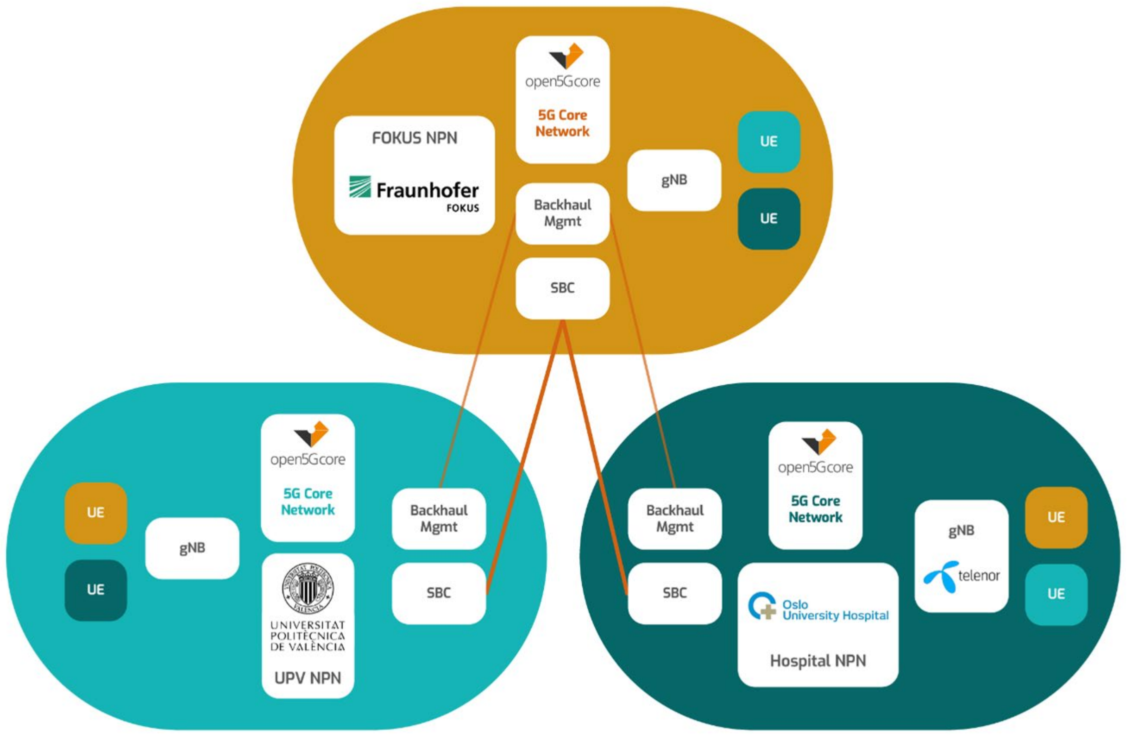


Figure 10: 5G Core deployed at three locations (Berlin, Valencia, Oslo) and interconnected for Interconnected NPNs use case

During the tenure of FUDGE-5G project as part of the use case Interconnected NPNs the Open5GCore testbed has been extended with new features. A new component SBC with the functionalities of SCP and SEPP, has been developed and integrated with the FOKUS 5G core for routing secured messages between private networks. Home-routed roaming feature (following 3GPP specification [5]) was also added in the 5G core to provide the roaming users access to their home services.

3.4. Microservices/NFs

One of the innovative frameworks proposed by FUDGE-5G is that 5G Core Networks (5GCs) can be orchestrated as cloud-native services. To enable this, 5GC Network Functions (NFs) need to be programmed as modular flexible and scalable pieces of software that fit within cloud-native architectures. In this perspective, FUDGE-5G’s Task 2.4, “Disintegration of Network Functions as Microservices,” was devoted to the study of how such functions can be efficiently decomposed into *microservices*, which by definition makes them suitable for cloud-native environments and “orchestrable” within end-to-end Enterprise Services, according to the terminology defined by the project (cf. D1.2 [6], D2.1 [7]). The technical work of T2.4 is reported in D2.4 [8]. In particular, in Section 3.2.4 therein, we analysed a possible decomposition into microservices of the following heterogenous set of 5GC NFs:

- Unified Data Management (UDM),
- Authentication Server Function (AUSF),

- User Plane Function (UPF),
- Policy Control Function (PCF),
- Session Management Functions (SMF),
- Network Exposure Function (NEF),
- Cell Broadcast Centre Function (CBCF).

Such NFs were chosen for their specific role within the 5GC and their relevance as enablers of advanced 5G functionalities. Such work does not contradict the architectural specifications of the 5GC [9] at an inter-NF level but goes beyond the state of the art in the internal design of the NFs, fostering the adoption of microservices in their development. In particular, in D2.4 [8] we advocated the need for a non-trivial redesign of the 5GC into a microservice-based architecture to address the fact that it is not possible to program the existing standardized 5GC NFs as collections of independent microservices by simply mapping each of the distinct standardized services offered by the NFs with a single corresponding microservice.

D2.4 [8] already contained some validation analyses, derived after the actual development in real proofs of concept of FUDGE-5G's re-designed NFs. In the following, we report further complementary results and considerations on NEF, CBCF, AUSF, and PCF, obtained within the activities of WP3 and of the final months of WP2, after the submission of D2.4 [8].

3.4.1. NEF

The validation tests of NEF can be split into two main objectives. First, the goal of ensuring a 3GPP compliant implementation and integration of NEF with the remaining NFs of the 5GC. Second, to showcase some of the advantages of the microservice based architecture by means of analytical results.

In D3.2 [3], results of the integration tests with the Fraunhofer FOKUS 5G Core were described. With those results, it was possible to confirm a successful implementation and integration with 5GC. Further traces and logs can be found in the Annex A.

Regarding the validation of the microservice based architecture, it consisted of stress tests by sending multiple requests for QoS policy changes (for PCF, through NEF) with two different setups. In the first setup, detailed in Table 4, all the microservices were deployed on the same machine, while the AF sending the requests and the PCF were in different machines.

Table 4: NEF test setup 1 details

VNF	Components	Resources
NEF	Gateway Service Npcf Service + Database	2 vCPU @ 2 GHz 2GB of RAM

PCF	PCF Service	2 vCPU @ 2 GHz 2GB of RAM
AF	100 processes	2 vCPU @ 2 GHz 2GB of RAM

The second setup used the same components, but now with NEF's microservices split into different machines (1:1 mapping). Table 5 further details that setup.

Table 5: NEF test setup 2 details

VNF	Components	Resources
NEF	Gateway Service	2 vCPU @ 2 GHz 2GB of RAM
	Npcf Service + Database	2 vCPU @ 2 GHz 2GB of RAM
PCF	PCF Service	2 vCPU @ 2 GHz 2GB of RAM
AF	100 processes	2 vCPU @ 2 GHz 2GB of RAM

Figure 11 describes the base setup, giving further context on the composition of the NEF and communication between the multiple components involved.

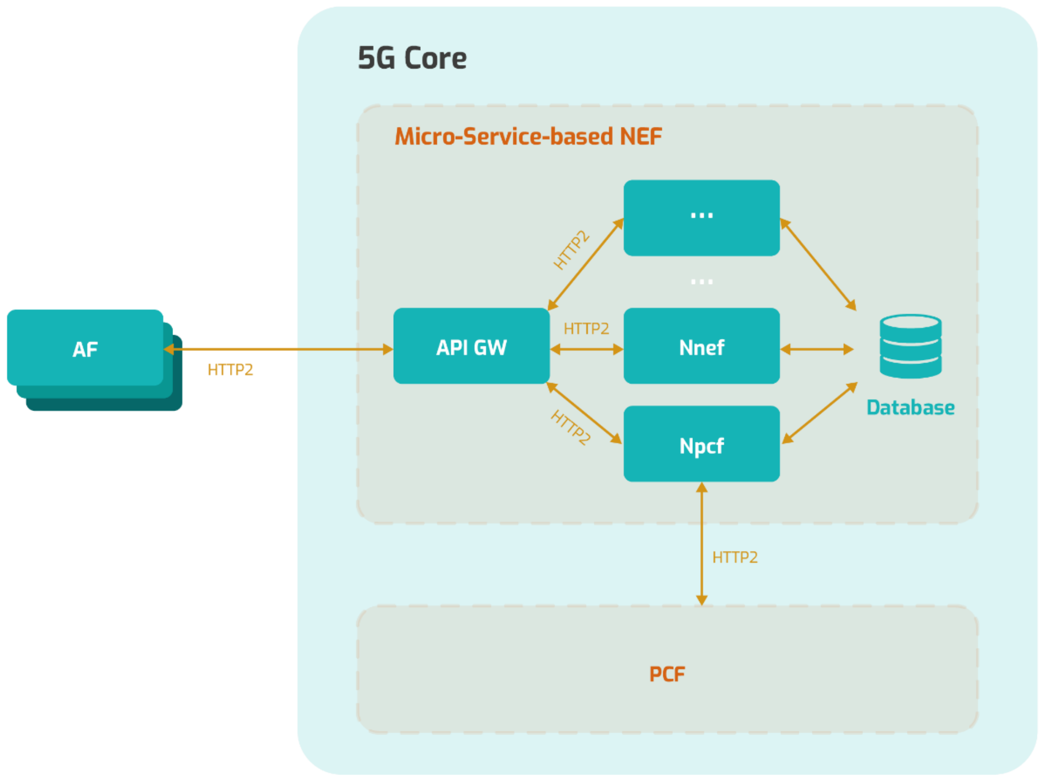


Figure 11: NEF base test setup

To obtain results from the setups, multiple test runs were executed. Each test run executed for five minutes, with every 100 instances of the AF sending one request per second. For each run, the response time for each request, and the current CPU and RAM of the machines was registered. In Table 6, one can observe the results from tests using both the setups.

Table 6: Microservice NEF tests results

Measurement	Setup 1	Setup 2
Total requests	11695	16005
Complete requests	11695	16003
Average CPU usage	99%	82%
Average RAM usage	50%	38%
Average requests per second	38	53

Comparing the results, it is verifiable a noticeable improvement in the number of requests processed with the disintegrated NEF setup. This improvement can be attributed to the fact that less processing power is required from each machine, allowing for more requests to be processed. In the first setup, the CPU of the NEF machine was at maximum usage, where in the second setup, there was still a substantial amount of CPU usage available. This supports one of the key advantages of a decomposed microservice architecture, where the usage of stateless micro services enabled easy scalability and recovery in case of failures.

Even though the objective of the tests was accomplished, it is noticeable a big decrease of performance in terms of requests processed in a second when comparing with results reported in D2.4 [8]. However, those early tests were performed using an older version of NEF with a much simpler code base (older 3GPP standard) and still using HTTP 1.1 as the base protocol. Our findings demonstrated that HTTP 2.0 is heavier and brings some decreases in overall performance. To further confirm that fact, further test runs were performed using the same HTTP 1.1 libraries from the D2.4 [8] tests. Table 7, shows the results with that change using the same setup described in Table 4 (setup 1).

Table 7: Microservice NEF test results (HTTP 1.1 only)

Measurement	Result
Complete requests	24254
Average CPU usage	95%
Average RAM usage	50%
Average requests per second	81

3.4.2. CBCF

For the integration testing the CBCF was deployed on the same AWS node as Cumucore's AMF. The CBCF only uses the services of the AMF in this 5Gcore network.

TCP traces were made to measure the time it takes from posting the message to the REST microservice and output ADAPTER 5G accomplishing the WRITE operation to the AMF. See Figure 11. The write/replace message contained only a single cell, which would be consistent with the deployment in the NOW. The processing time in the microservices-based CBCF is 250-260 ms.

The traces are taken in the REST container and in the ADAPTER 5G container. Because they run on the same host the clock is synchronized – it is the same clock. The traces show (see Figure 12):

1. from 172.30.0.1 (host) to 172.30.0.7 (rest) ingest port 7000 – message post
2. from 172.30.0.9 (adapter5g) to 172.30.2.1 (AMF) port 4002 – write/replace

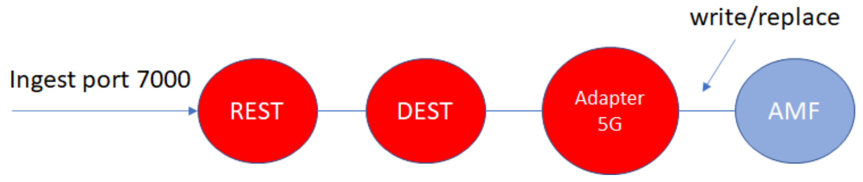
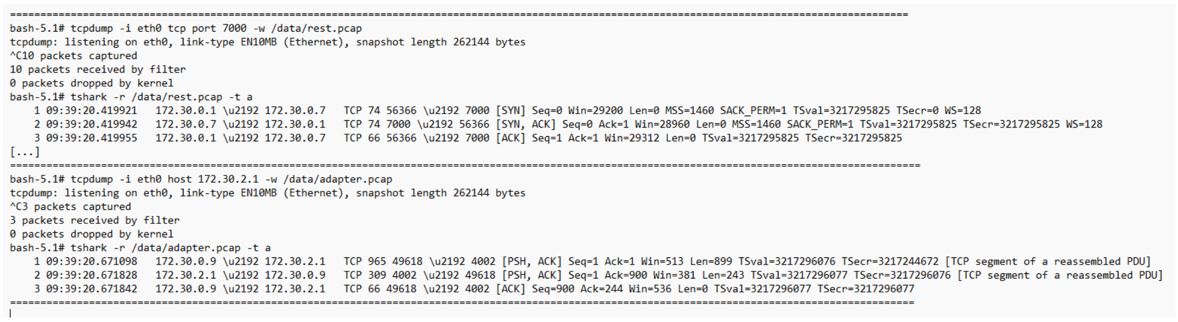


Figure 12: CBCF representation

The O2M monolithic CBC is about 10 times faster. An initial analysis shows that the use of Kafka could be the reason for the high latency. Further analyses will be performed to verify this.

Lab tests were also performed in Cumucore's lab. See section 3.2.

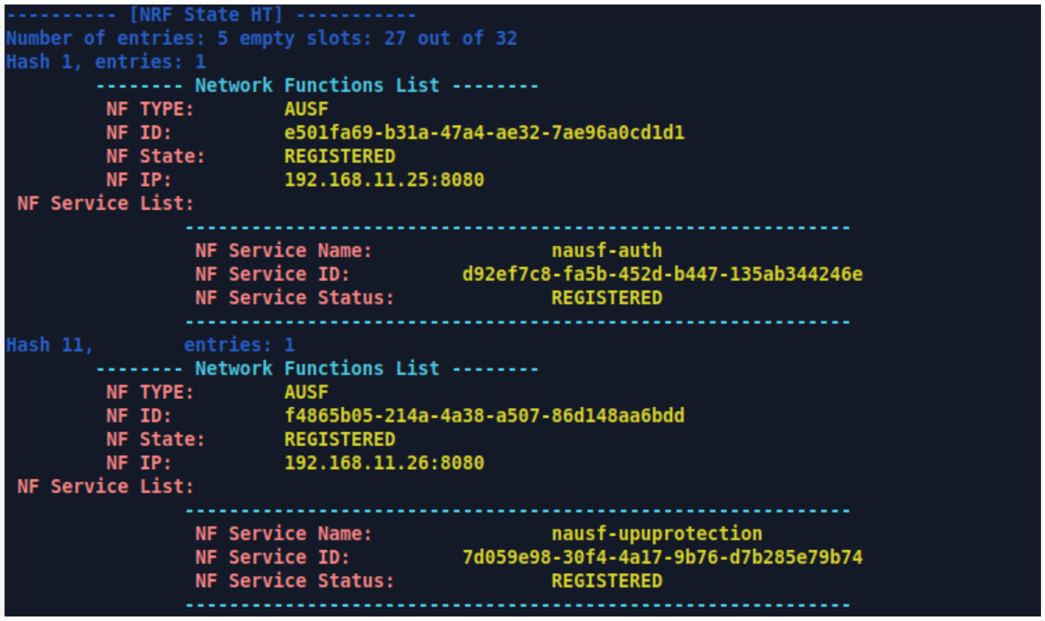


```
==========================================================================================================================================================
bash-5.1# tcpdump -i eth0 tcp port 7000 -w /data/rest.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C10 packets captured
10 packets received by filter
0 packets dropped by kernel
bash-5.1# tshark -r /data/rest.pcap -t a
    1 09:39:20.419921   172.30.0.1 \u2192 172.30.0.7   TCP 74 56366 \u2192 7000 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3217295825 TSecr=0 WS=128
    2 09:39:20.419942   172.30.0.7 \u2192 172.30.0.1   TCP 74 7000 \u2192 56366 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3217295825 TSecr=3217295825 WS=128
    3 09:39:20.419955   172.30.0.1 \u2192 172.30.0.7   TCP 66 56366 \u2192 7000 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3217295825 TSecr=3217295825
[...]
============================================================================================================================================================
bash-5.1# tcpdump -i eth0 host 172.30.2.1 -w /data/adapter.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C3 packets captured
3 packets received by filter
0 packets dropped by kernel
bash-5.1# tshark -r /data/adapter.pcap -t a
    1 09:39:20.671098   172.30.0.9 \u2192 172.30.2.1   TCP 965 49618 \u2192 4002 [PSH, ACK] Seq=1 Ack=1 Win=513 Len=899 TSval=3217296076 TSecr=3217244672 [TCP segment of a reassembled PDU]
    2 09:39:20.671828   172.30.2.1 \u2192 172.30.0.9   TCP 309 4002 \u2192 49618 [PSH, ACK] Seq=1 Ack=900 Win=381 Len=243 TSval=3217296077 TSecr=3217296076 [TCP segment of a reassembled PDU]
    3 09:39:20.671842   172.30.0.9 \u2192 172.30.2.1   TCP 66 49618 \u2192 4002 [ACK] Seq=900 Ack=244 Win=536 Len=0 TSval=3217296077 TSecr=3217296077
===========================================================================================================================================================
|
```

Figure 13: Traces from CBCF alert message delivery

3.4.3. AUSF

The AUSF was breakdown for different services and was deployed as different instances using Kubernetes containers. Each of the AUSF instances registers with its services to NRF and the consumer selects the instance depending on the service needs. As shown in Figure 13, two instances of AUSF is registered and AUSF instance serving 'nausf-auth' service takes part in the authorization process. So, AMF as a consumer sends authentication request to the first AUSF here from the image. Multiple AUSF can also be serving the same services resulting into load balancing in the system.



```
---------- [NRF State HT] -----------
Number of entries: 5 empty slots: 27 out of 32
Hash 1, entries: 1
        -------- Network Functions List --------
         NF TYPE:        AUSF
         NF ID:          e501fa69-b31a-47a4-ae32-7ae96a0cd1d1
         NF State:       REGISTERED
         NF IP:          192.168.11.25:8080
 NF Service List:
                ------------------------------------------------------------
                 NF Service Name:               nausf-auth
                 NF Service ID:         d92ef7c8-fa5b-452d-b447-135ab344246e
                 NF Service Status:             REGISTERED
                ------------------------------------------------------------
Hash 11,        entries: 1
        -------- Network Functions List --------
         NF TYPE:        AUSF
         NF ID:          f4865b05-214a-4a38-a507-86d148aa6bdd
         NF State:       REGISTERED
         NF IP:          192.168.11.26:8080
 NF Service List:
                ------------------------------------------------------------
                 NF Service Name:               nausf-upuprotection
                 NF Service ID:         7d059e98-30f4-4a17-9b76-d7b285e79b74
                 NF Service Status:             REGISTERED
                ------------------------------------------------------------
```

Figure 14: List of AUSFs registered at NRF

4. 5G Devices

This section presents the results that served us to validate the 5G devices (5G modem) developed by Fivecomm in the context of FUDGE-5G. Such devices were specifically designed to satisfy the needs envisaged at the beginning of the project for the use case demonstration of the **Industry 4.0 vertical**. However, such devices have been used in other demonstrations and tests during the project, e.g. they have been used in London to validate the IDE platform, among others.

Since such prototypes were developed for the Industry 4.0, it is in such test-bed were they were also validated. The validation has been based on three main 5G network-related KPIs:

i. 5G RTT latency (between two 5G devices).
ii. One-way latency (5G device to 5G core).
iii. Throughput
iv. Radio signal quality

As mentioned, the 5G modem is a hardware-based device that provides sub-6 GHz connectivity. Frequency bands such as n77 and n78, used in the ABB lab are supported (among many others). The 5G modem also supports both non-standalone (NSA) and standalone (SA) configurations, although just the last one was used in the trial. The following picture shows the final form factor of the 5G devices employed at the end of the project.

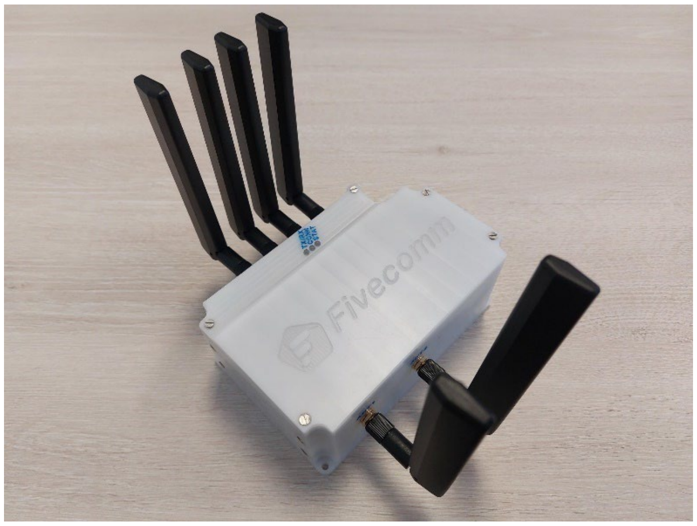


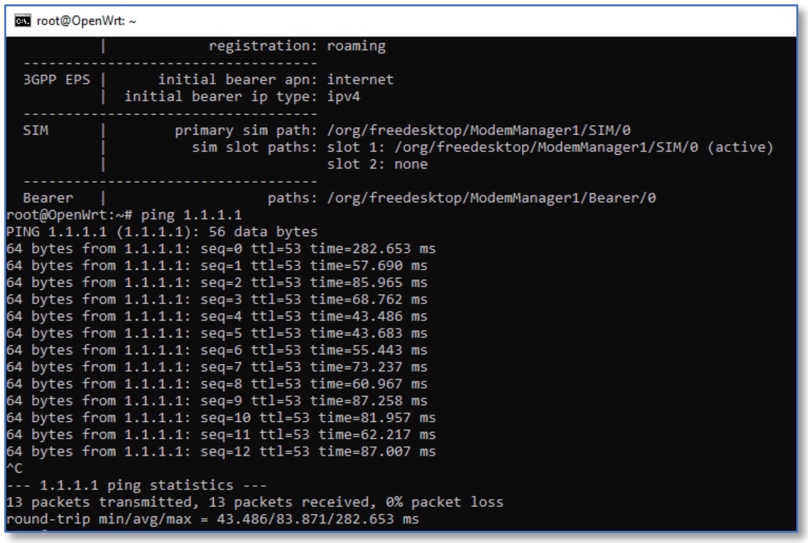
Figure 15: 5G modem developed by Fivecomm and validated in ABB lab, as part of UC4 final trials.

During the last stage of the project, up to three modems were used in different locations.

- ***Modem A***: available and validated in ABB premises for UC4 trials.
- ***Modem B***: available and validated in IDE lab first, shipped to ABB premises as a second modem for the UC4 trials.
- ***Modem C***: available and validated in Cumucore lab. Its software was modified to include all needed features for TSN support.

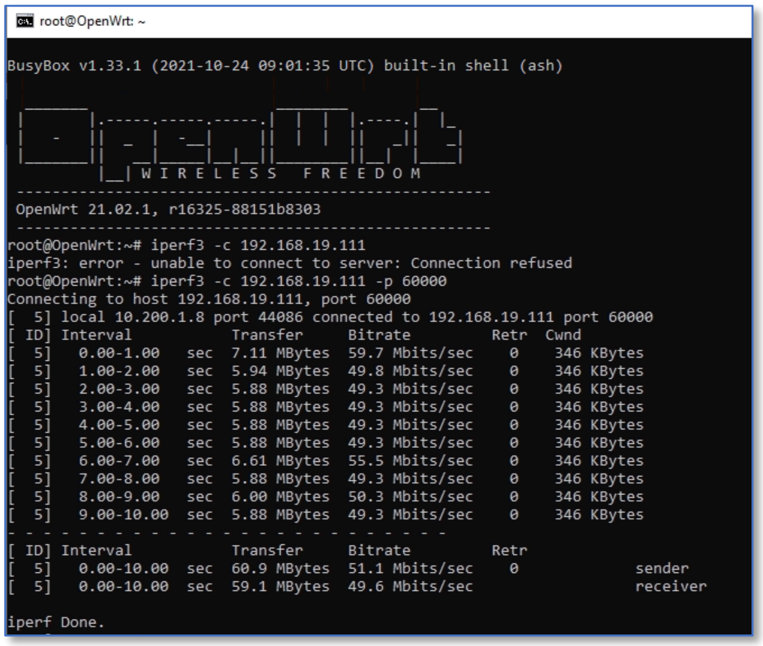
The Industry 4.0 validation has been performed in two different tracks: ABB (end-to-end trial of test cases with 5GLAN) and Cumucore (TSN setup). The modem in Cumucore was already validated during the first half of the project and reported in D4.1 [10].

Regarding the ABB setup, a first validation was made at Telenor premises by using the network architecture explained in D3.2 [3]. Once the integration was fully completed, the following results were obtained.



```
root@OpenWrt: ~
          |              registration: roaming
  ----------------------------------
  3GPP EPS |       initial bearer apn: internet
           |   initial bearer ip type: ipv4
  ----------------------------------
  SIM      |         primary sim path: /org/freedesktop/ModemManager1/SIM/0
           |           sim slot paths: slot 1: /org/freedesktop/ModemManager1/SIM/0 (active)
           |                           slot 2: none
  ----------------------------------
  Bearer   |                    paths: /org/freedesktop/ModemManager1/Bearer/0
root@OpenWrt:~# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1): 56 data bytes
64 bytes from 1.1.1.1: seq=0 ttl=53 time=282.653 ms
64 bytes from 1.1.1.1: seq=1 ttl=53 time=57.690 ms
64 bytes from 1.1.1.1: seq=2 ttl=53 time=85.965 ms
64 bytes from 1.1.1.1: seq=3 ttl=53 time=68.762 ms
64 bytes from 1.1.1.1: seq=4 ttl=53 time=43.486 ms
64 bytes from 1.1.1.1: seq=5 ttl=53 time=43.683 ms
64 bytes from 1.1.1.1: seq=6 ttl=53 time=55.443 ms
64 bytes from 1.1.1.1: seq=7 ttl=53 time=73.237 ms
64 bytes from 1.1.1.1: seq=8 ttl=53 time=60.967 ms
64 bytes from 1.1.1.1: seq=9 ttl=53 time=87.258 ms
64 bytes from 1.1.1.1: seq=10 ttl=53 time=81.957 ms
64 bytes from 1.1.1.1: seq=11 ttl=53 time=62.217 ms
64 bytes from 1.1.1.1: seq=12 ttl=53 time=87.007 ms
^C
--- 1.1.1.1 ping statistics ---
13 packets transmitted, 13 packets received, 0% packet loss
round-trip min/avg/max = 43.486/83.871/282.653 ms
```

Figure 16: 5G RTT latency results (ping) at Telenor premises before moving to ABB.



```
root@OpenWrt: ~

BusyBox v1.33.1 (2021-10-24 09:01:35 UTC) built-in shell (ash)

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 -----------------------------------------------------
 OpenWrt 21.02.1, r16325-88151b8303
 -----------------------------------------------------
root@OpenWrt:~# iperf3 -c 192.168.19.111
iperf3: error - unable to connect to server: Connection refused
root@OpenWrt:~# iperf3 -c 192.168.19.111 -p 60000
Connecting to host 192.168.19.111, port 60000
[  5] local 10.200.1.8 port 44086 connected to 192.168.19.111 port 60000
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  7.11 MBytes  59.7 Mbits/sec    0    346 KBytes
[  5]   1.00-2.00   sec  5.94 MBytes  49.8 Mbits/sec    0    346 KBytes
[  5]   2.00-3.00   sec  5.88 MBytes  49.3 Mbits/sec    0    346 KBytes
[  5]   3.00-4.00   sec  5.88 MBytes  49.3 Mbits/sec    0    346 KBytes
[  5]   4.00-5.00   sec  5.88 MBytes  49.3 Mbits/sec    0    346 KBytes
[  5]   5.00-6.00   sec  5.88 MBytes  49.3 Mbits/sec    0    346 KBytes
[  5]   6.00-7.00   sec  6.61 MBytes  55.5 Mbits/sec    0    346 KBytes
[  5]   7.00-8.00   sec  5.88 MBytes  49.3 Mbits/sec    0    346 KBytes
[  5]   8.00-9.00   sec  6.00 MBytes  50.3 Mbits/sec    0    346 KBytes
[  5]   9.00-10.00  sec  5.88 MBytes  49.3 Mbits/sec    0    346 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-10.00  sec  60.9 MBytes  51.1 Mbits/sec    0             sender
[  5]   0.00-10.00  sec  59.1 MBytes  49.6 Mbits/sec                  receiver

iperf Done.
```

Figure 17: 5G throughput (limited by the UL) results (iperf) at Telenor premises before moving to ABB.

Once the devices and the 5G network were validated, the next step was to move the equipment from Telenor to ABB premises and validate again by following the same procedure. The final network diagram is shown in Figure 17, for the particular measurement of 5G latency between a device and the 5GC. Note that a similar configuration was used for other KPIs, where two modems were used.

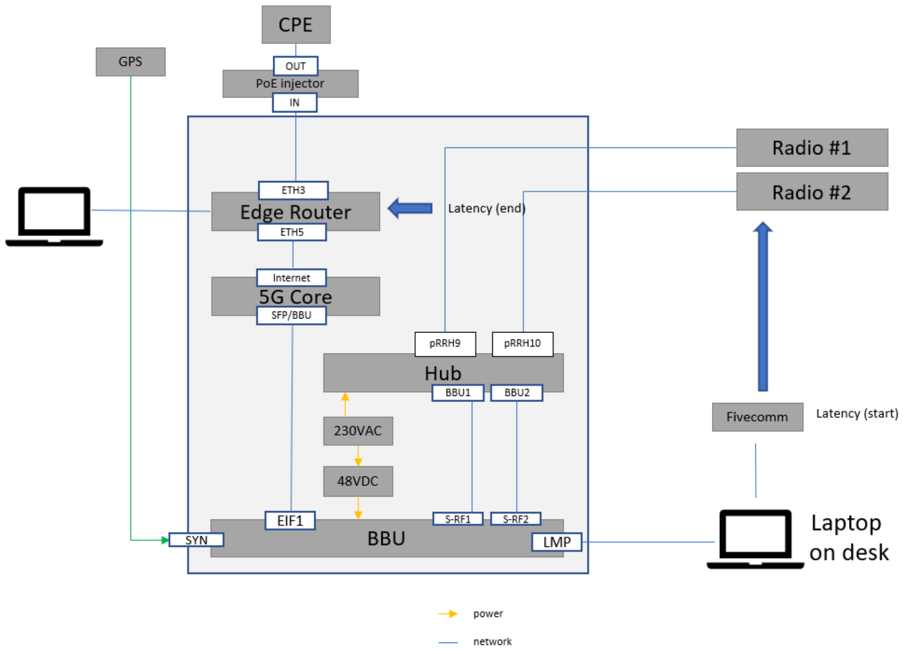
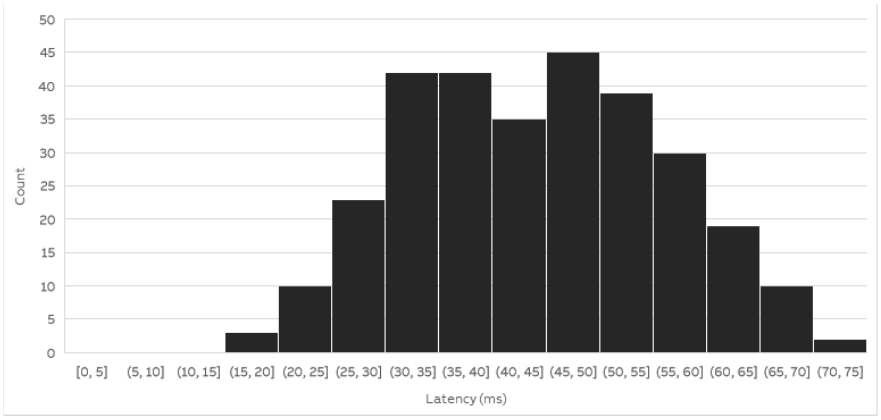


Figure 18: 5G network architecture. Case for 5G latency from the device to the 5GC.

In the following lines, we show the most relevant results obtained for validating the devices (and the 5G network) in ABB.

A. 5G RTT latency

Up to 4 tests were performed, whose results are the following.



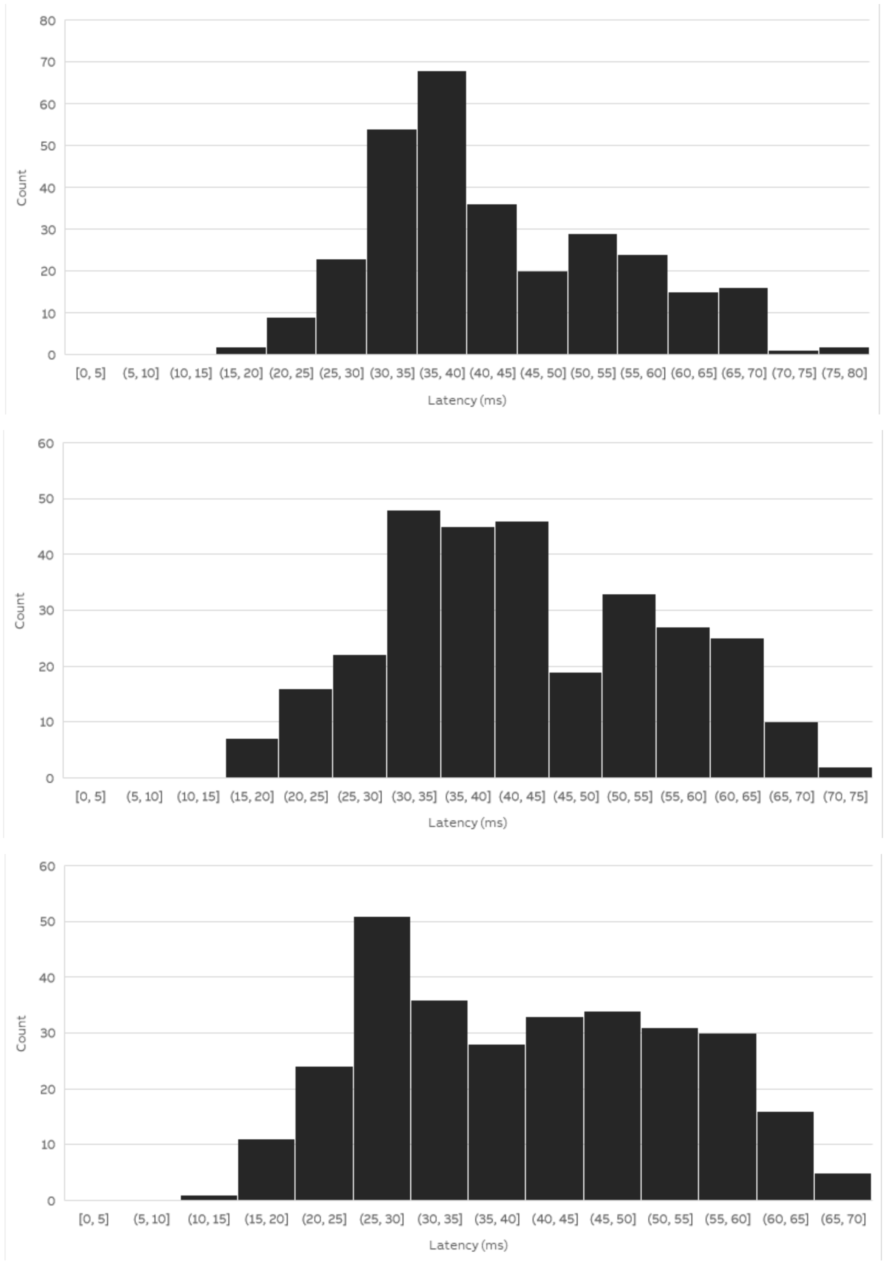


Figure 19. 5G RTT latency results (tests 1 to 4).

As it can be observed, similar results to those shown in Figure 15 (Telenor premises) were obtained. The values obtained, for instance, for the last test, were:

- Minimum: 12.305 ms
- Maximum: 68.56 ms
- Average: 40.30 ms

B. *One-way latency (device-5GC)*

Two different tests were performed. The first test provided the following latency values.

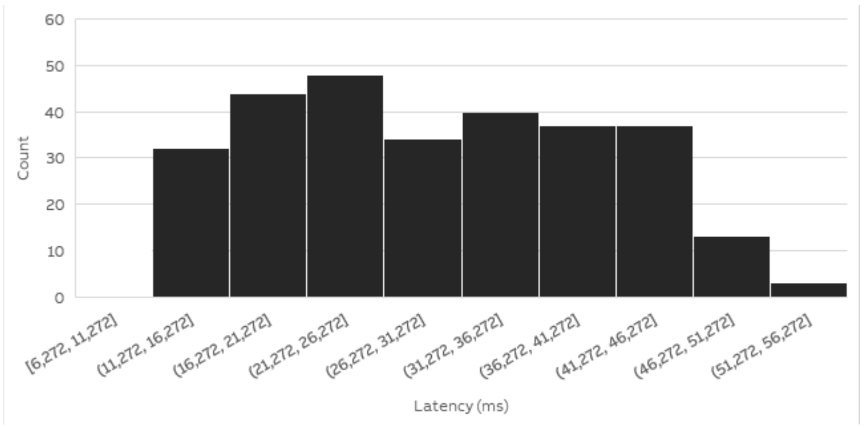


Figure 20. 5G one-way latency results (test 1).

The second test provided slightly better results, with the following values:

- Minimum: 9.347 ms
- Maximum: 54.128 ms
- Average: 17.78 ms

C. 5G throughput

Throughput was measured by using both UDP and TCP transmissions. The following results were obtained.

Table 8: 5G Throughput

Throughput			
TCP	Downlink	Sender	399Mbits/s
		Receiver	395Mbits/s
	Uplink	Sender	50.2Mbits/s
		Receive	42.9Mbits/s
UDP	Downlink	950Mbits/s	
	Uplink	43.8Mbits/s	

D. Radio signal quality

In this final test, RSRP, RSRQ and SINR results were obtained. Note that the 5G radio dot and the device were in a static position. The average values obtained were -72 dBm, -11 dB and 27 dB respectively.

5. Innovations

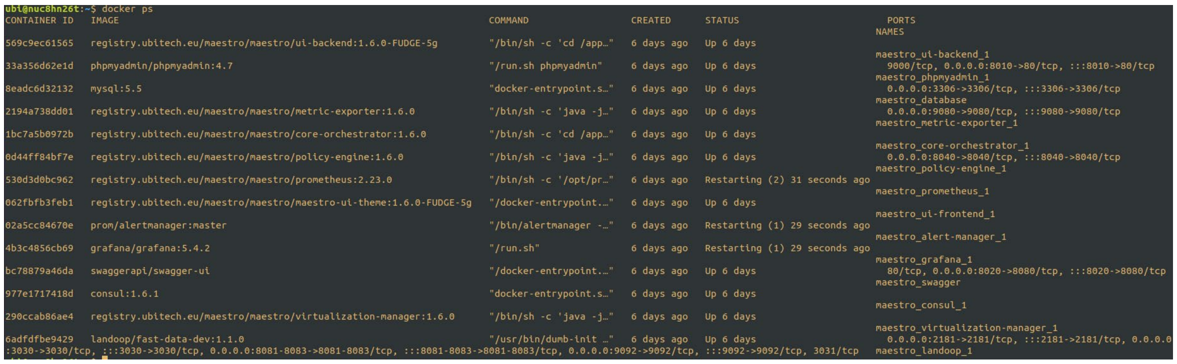
During the project, multiple technologies and innovations have been developed to complement and fulfil the objectives of the various use cases. This section details the validation efforts carried out for six of these innovations.

5.1. VAO

Vertical Application Orchestrator is a piece of the Fudge-5G Platform which its purpose is to provide an interfacing layer to the end user (i.e., the vertical service expert and the application developer) for managing the deployable applications and their features, without interfering with the network level functions and processes managed by the involved telecom operators and infrastructure owners. Thus, it decouples the application layer management procedures from the network layer management, providing application awareness to the network orchestration entity and specifically to the SFVO through(N6) as shown in Figure 2.

The VAO Administrative domain consists of all modules that are responsible for registering a vertical service and all its components, providing cloud-related and network-related metadata, authoring deployment and runtime management of the operational state of the vertical service per se.

Vertical Application Orchestrator has been successfully deployed and integrated over London's Testbed as already stated in Deliverable 3.2 [3]. The deployed version of the VAO is comprised by 14 micro services and is depicted in Figure 20.



```
ubi@nuc8hn26t:~$ docker ps
CONTAINER ID   IMAGE                                                          COMMAND                  CREATED      STATUS                            PORTS
                                                                                                                                                        NAMES
569c9ec61565   registry.ubitech.eu/maestro/maestro/ui-backend:1.6.0-FUDGE-5g  "/bin/sh -c 'cd /app…"   6 days ago   Up 6 days
                                                                                                                                                        maestro_ui-backend_1
33a356d62e1d   phpmyadmin/phpmyadmin:4.7                                      "/run.sh phpmyadmin"     6 days ago   Up 6 days                           9000/tcp, 0.0.0.0:8010->80/tcp, :::8010->80/tcp
                                                                                                                                                        maestro_phpmyadmin_1
8eadc6d32132   mysql:5.5                                                      "docker-entrypoint.s…"   6 days ago   Up 6 days                           0.0.0.0:3306->3306/tcp, :::3306->3306/tcp
                                                                                                                                                        maestro_database
2194a738dd01   registry.ubitech.eu/maestro/maestro/metric-exporter:1.6.0      "/bin/sh -c 'java -j…"   6 days ago   Up 6 days                           0.0.0.0:9080->9080/tcp, :::9080->9080/tcp
                                                                                                                                                        maestro_metric-exporter_1
1bc7a5b0972b   registry.ubitech.eu/maestro/maestro/core-orchestrator:1.6.0    "/bin/sh -c 'cd /app…"   6 days ago   Up 6 days
                                                                                                                                                        maestro_core-orchestrator_1
0d44ff84bf7e   registry.ubitech.eu/maestro/maestro/policy-engine:1.6.0        "/bin/sh -c 'java -j…"   6 days ago   Up 6 days                           0.0.0.0:8040->8040/tcp, :::8040->8040/tcp
                                                                                                                                                        maestro_policy-engine_1
530d3d0bc962   registry.ubitech.eu/maestro/maestro/prometheus:2.23.0          "/bin/sh -c '/opt/pr…"   6 days ago   Restarting (2) 31 seconds ago
                                                                                                                                                        maestro_prometheus_1
062fbfb3feb1   registry.ubitech.eu/maestro/maestro/maestro-ui-theme:1.6.0-FUDGE-5g   "/docker-entrypoint.…"   6 days ago   Up 6 days
                                                                                                                                                        maestro_ui-frontend_1
02a5cc84670e   prom/alertmanager:master                                       "/bin/alertmanager -…"   6 days ago   Restarting (1) 29 seconds ago
                                                                                                                                                        maestro_alert-manager_1
4b3c4856cb69   grafana/grafana:5.4.2                                          "/run.sh"                6 days ago   Restarting (1) 29 seconds ago
                                                                                                                                                        maestro_grafana_1
bc78879a46da   swaggerapi/swagger-ui                                          "/docker-entrypoint.…"   6 days ago   Up 6 days                           80/tcp, 0.0.0.0:8020->8080/tcp, :::8020->8080/tcp
                                                                                                                                                        maestro_swagger
977e1717418d   consul:1.6.1                                                   "docker-entrypoint.s…"   6 days ago   Up 6 days
                                                                                                                                                        maestro_consul_1
290ccab86ae4   registry.ubitech.eu/maestro/maestro/virtualization-manager:1.6.0   "/bin/sh -c 'java -j…"   6 days ago   Up 6 days
                                                                                                                                                        maestro_virtualization-manager_1
6adfdfbe9429   landoop/fast-data-dev:1.1.0                                    "/usr/bin/dumb-init …"   6 days ago   Up 6 days                           0.0.0.0:2181->2181/tcp, :::2181->2181/tcp, 0.0.0.0
:3030->3030/tcp, :::3030->3030/tcp, 0.0.0.0:8081-8083->8081-8083/tcp, :::8081-8083->8081-8083/tcp, 0.0.0.0:9092->9092/tcp, :::9092->9092/tcp, 3031/tcp   maestro_landoop_1
```

Figure 21: VAO Deployment over IDE's Testbed.

5.1.1. Technical Testing of the VAO

In this section, the technical evaluation results for the software mechanisms and the integration with the platform are provided.

5.1.1.1. Mechanism Testing - Unit testing

The main objective of this sub-section is to provide information about the unit tests applied during the deployment of the VAO. Unit tests are used for testing the functional mechanisms of a piece of software. In this particular case, unit tests are used to guarantee the quality of the mechanisms deployed as well as to validate the integration with the other parts of FUDGE-5G Platform.

Just to clarify that a unit test is applied to a piece of code without any dependencies on other code parts. Therefore, each microservice and each mechanism has been tested with unit test before integrating with the other pieces of the Platform.

A simplified list of unit tests can be seen in table below:

Table 9: Unit Tests Performed to validate the deployment.

Description	Result Status
Create a new Component Descriptor (serialized format)	Passed
Fetch Component Descriptor	Passed
Update Component Descriptor	Passed
Delete Component Descriptor	Passed
Create Application Graph	Passed
Fetch Application Graph	Passed
Update Application Graph	Passed
Delete Application Graph	Passed
Create Application Graph Instance	Passed
Fetch Application Graph Instance	Passed
Delete Application Graph Instance	Passed
Execute Deployment	Passed
Update Slice Intent	Passed
Un-deploy Application Graph Instance	Passed

Deploy Application Graph Instance	Passed

The unit testing is being conducted on the API exposed and has been thoroughly described in Deliverable 2.1 [7].

Two indicative unit tests can be found in the Annex B.

5.1.1.2. Integration Testing

This section presents the tests that are covering integration of multiple mechanisms of the VAO (internal integration test) and mechanisms of the VAO with other systems or mechanisms of the FUDGE-5G Platform. Integration testing identifies problems that occur when units are combined and is considered as an extension of unit testing, that tests interfaces and interactions between the components.

The main idea of integration testing is to start from two or more units that have already been tested and test the integration between them. In most cases methods from different mechanisms are combined in order to achieve the needed functionality, so testing requires combination of pieces of software that create a basic integrated functionality.

Next, we have defined and conducted the next integration actions:

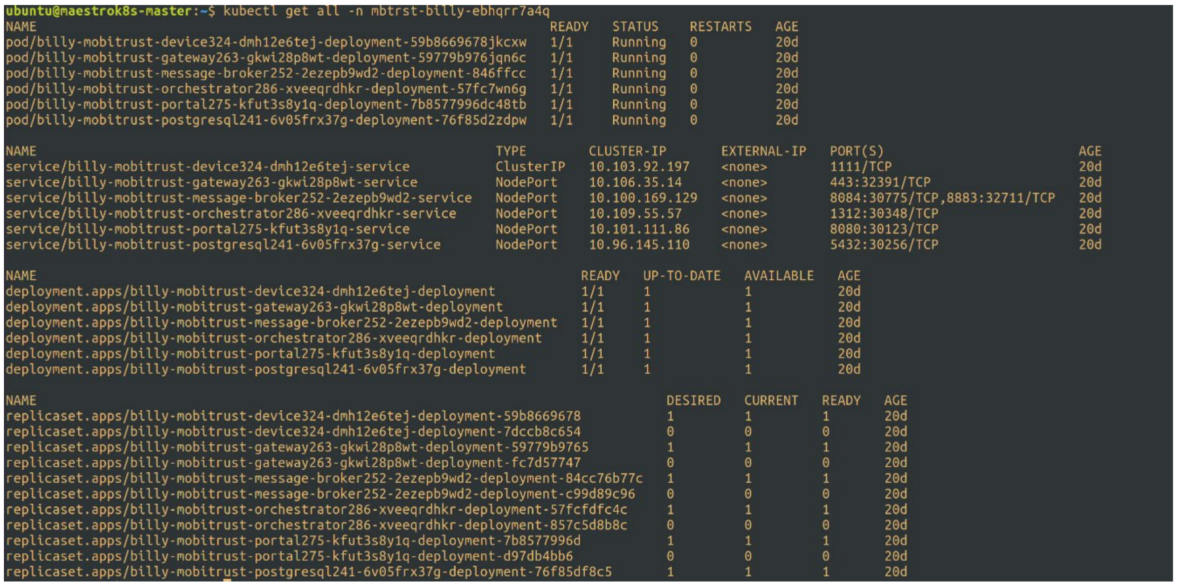
i. Onboard one vertical service
ii. Make a deployment of a vertical service through VAO
iii. Test the connectivity between the service graph components during runtime
iv. Test the connectivity with the 5G-Core

The results are summarised in the table below:

Table 10: Integration Tests performed.

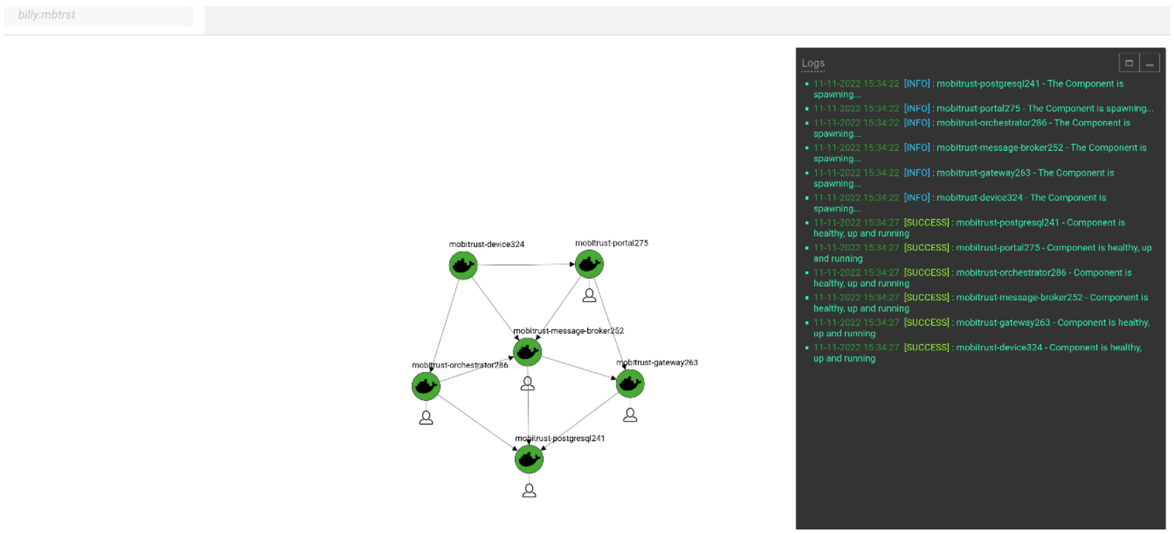
Description	Result Status	
Onboard one vertical service	Passed	Successful Onboarding of the OneSource's vertical service.
Make a deployment of a vertical service through VAO	Passed	Successful Deployment of OneSource's Vertical service on London Testbed (see Figure 21, Figure 22)
Test the connectivity between the service graph components during runtime	Passed	Successful Connection between two Vertical components (see Figure 23)

Test the connectivity with the 5G-Core	Passed	



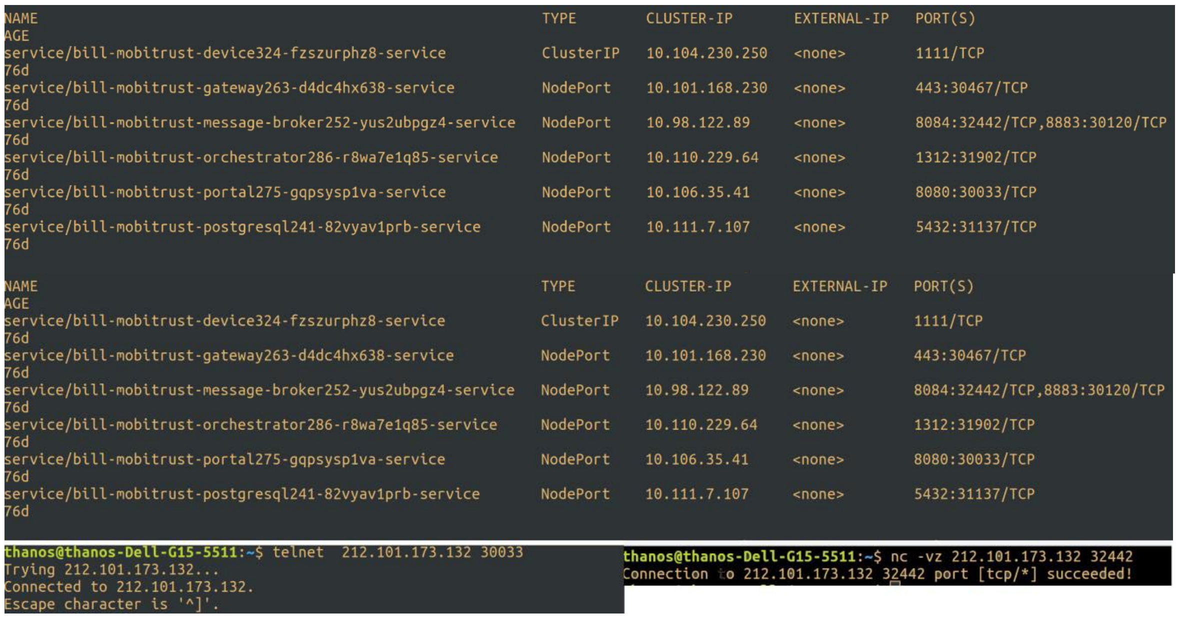
```
ubuntu@maestrok8s-master:~$ kubectl get all -n mbtrst-billy-ebhqrr7a4q
NAME                                                                READY   STATUS    RESTARTS   AGE
pod/billy-mobitrust-device324-dmh12e6tej-deployment-59b8669678jkcxw   1/1     Running   0          20d
pod/billy-mobitrust-gateway263-gkwi28p8wt-deployment-59779b976jqn6c   1/1     Running   0          20d
pod/billy-mobitrust-message-broker252-2ezepb9wd2-deployment-846ffcc   1/1     Running   0          20d
pod/billy-mobitrust-orchestrator286-xveeqrdhkr-deployment-57fc7wn6g   1/1     Running   0          20d
pod/billy-mobitrust-portal275-kfut3s8y1q-deployment-7b8577996dc48tb   1/1     Running   0          20d
pod/billy-mobitrust-postgresql241-6v05frx37g-deployment-76f85d2zdpw   1/1     Running   0          20d

NAME                                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                          AGE
service/billy-mobitrust-device324-dmh12e6tej-service          ClusterIP   10.103.92.197    <none>        1111/TCP                         20d
service/billy-mobitrust-gateway263-gkwi28p8wt-service         NodePort    10.106.35.14     <none>        443:32391/TCP                    20d
service/billy-mobitrust-message-broker252-2ezepb9wd2-service  NodePort    10.100.169.129   <none>        8084:30775/TCP,8883:32711/TCP    20d
service/billy-mobitrust-orchestrator286-xveeqrdhkr-service    NodePort    10.109.55.57     <none>        1312:30348/TCP                   20d
service/billy-mobitrust-portal275-kfut3s8y1q-service          NodePort    10.101.111.86    <none>        8080:30123/TCP                   20d
service/billy-mobitrust-postgresql241-6v05frx37g-service      NodePort    10.96.145.110    <none>        5432:30256/TCP                   20d

NAME                                                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/billy-mobitrust-device324-dmh12e6tej-deployment       1/1     1            1           20d
deployment.apps/billy-mobitrust-gateway263-gkwi28p8wt-deployment      1/1     1            1           20d
deployment.apps/billy-mobitrust-message-broker252-2ezepb9wd2-deployment 1/1   1            1           20d
deployment.apps/billy-mobitrust-orchestrator286-xveeqrdhkr-deployment 1/1     1            1           20d
deployment.apps/billy-mobitrust-portal275-kfut3s8y1q-deployment       1/1     1            1           20d
deployment.apps/billy-mobitrust-postgresql241-6v05frx37g-deployment   1/1     1            1           20d

NAME                                                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/billy-mobitrust-device324-dmh12e6tej-deployment-59b8669678          1         1         1       20d
replicaset.apps/billy-mobitrust-device324-dmh12e6tej-deployment-7dccb8c654          0         0         0       20d
replicaset.apps/billy-mobitrust-gateway263-gkwi28p8wt-deployment-59779b9765         1         1         1       20d
replicaset.apps/billy-mobitrust-gateway263-gkwi28p8wt-deployment-fc7d57747          0         0         0       20d
replicaset.apps/billy-mobitrust-message-broker252-2ezepb9wd2-deployment-84cc76b77c  1         1         1       20d
replicaset.apps/billy-mobitrust-message-broker252-2ezepb9wd2-deployment-c99d89c96   0         0         0       20d
replicaset.apps/billy-mobitrust-orchestrator286-xveeqrdhkr-deployment-57fcfdfc4c    1         1         1       20d
replicaset.apps/billy-mobitrust-orchestrator286-xveeqrdhkr-deployment-857c5d8b8c    0         0         0       20d
replicaset.apps/billy-mobitrust-portal275-kfut3s8y1q-deployment-7b8577996d          1         1         1       20d
replicaset.apps/billy-mobitrust-portal275-kfut3s8y1q-deployment-d97db4bb6           0         0         0       20d
replicaset.apps/billy-mobitrust-postgresql241-6v05frx37g-deployment-76f85df8c5      1         1         1       20d
```

Figure 22: Deployment of Onesources's Vertical service on London Testbed



```
▪ 11-11-2022 15:34:22 [INFO] : mobitrust-postgresql241 - The Component is spawning...
▪ 11-11-2022 15:34:22 [INFO] : mobitrust-portal275 - The Component is spawning...
▪ 11-11-2022 15:34:22 [INFO] : mobitrust-orchestrator286 - The Component is spawning...
▪ 11-11-2022 15:34:22 [INFO] : mobitrust-message-broker252 - The Component is spawning...
▪ 11-11-2022 15:34:22 [INFO] : mobitrust-gateway263 - The Component is spawning...
▪ 11-11-2022 15:34:22 [INFO] : mobitrust-device324 - The Component is spawning...
▪ 11-11-2022 15:34:27 [SUCCESS] : mobitrust-postgresql241 - Component is healthy, up and running
▪ 11-11-2022 15:34:27 [SUCCESS] : mobitrust-portal275 - Component is healthy, up and running
▪ 11-11-2022 15:34:27 [SUCCESS] : mobitrust-orchestrator286 - Component is healthy, up and running
▪ 11-11-2022 15:34:27 [SUCCESS] : mobitrust-message-broker252 - Component is healthy, up and running
▪ 11-11-2022 15:34:27 [SUCCESS] : mobitrust-gateway263 - Component is healthy, up and running
▪ 11-11-2022 15:34:27 [SUCCESS] : mobitrust-device324 - Component is healthy, up and running
```

Figure 23: Deployment of Onesources's Vertical service on London Testbed



```
NAME                                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/bill-mobitrust-device324-fzszurphz8-service           ClusterIP   10.104.230.250   <none>        1111/TCP
76d
service/bill-mobitrust-gateway263-d4dc4hx638-service          NodePort    10.101.168.230   <none>        443:30467/TCP
76d
service/bill-mobitrust-message-broker252-yus2ubpgz4-service   NodePort    10.98.122.89     <none>        8084:32442/TCP,8883:30120/TCP
76d
service/bill-mobitrust-orchestrator286-r8wa7e1q85-service     NodePort    10.110.229.64    <none>        1312:31902/TCP
76d
service/bill-mobitrust-portal275-gqpsysp1va-service           NodePort    10.106.35.41     <none>        8080:30033/TCP
76d
service/bill-mobitrust-postgresql241-82vyav1prb-service       NodePort    10.111.7.107     <none>        5432:31137/TCP
76d

NAME                                                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/bill-mobitrust-device324-fzszurphz8-service           ClusterIP   10.104.230.250   <none>        1111/TCP
76d
service/bill-mobitrust-gateway263-d4dc4hx638-service          NodePort    10.101.168.230   <none>        443:30467/TCP
76d
service/bill-mobitrust-message-broker252-yus2ubpgz4-service   NodePort    10.98.122.89     <none>        8084:32442/TCP,8883:30120/TCP
76d
service/bill-mobitrust-orchestrator286-r8wa7e1q85-service     NodePort    10.110.229.64    <none>        1312:31902/TCP
76d
service/bill-mobitrust-portal275-gqpsysp1va-service           NodePort    10.106.35.41     <none>        8080:30033/TCP
76d
service/bill-mobitrust-postgresql241-82vyav1prb-service       NodePort    10.111.7.107     <none>        5432:31137/TCP
76d
```

```
thanos@thanos-Dell-G15-5511:~$ telnet  212.101.173.132 30033
Trying 212.101.173.132...
Connected to 212.101.173.132.
Escape character is '^]'.
```

```
thanos@thanos-Dell-G15-5511:~$ nc -vz 212.101.173.132 32442
Connection to 212.101.173.132 32442 port [tcp/*] succeeded!
```

Figure 24:Successful Connection on two deployed Vertical components

5.2. Thales Slice Orchestrator

Within NEXIUM Defence Cloud suite, Thales's Orchestrator provides mission-oriented interfaces to easily deploy and manage end-to-end services in ICT architecture. Orchestration capabilities overcome infrastructure complexity by enabling automated processes.

In order to manage cloud and programmable networks, the Orchestrator gives ICT operators the capability to deploy end-to-end services.

Thales's Military Operations Orchestrator manages at the same time application deployment, IT resources and network while remaining compliant with safety and security rules.

Thales Nexium Orchestration Suite can achieve:

- Rapid deployment following the mission tempo
- Controlled deployment capability to ensure alignment with security policies as imposed by critical systems
- Reconfiguration even in hostile environments with low or no connectivity
- System management taking into account the rare access to available resources and the constrained networks
- Deployment of any kind of applications (legacy, cloud-native…) in order to ensure a smooth transition to new IT infrastructures.

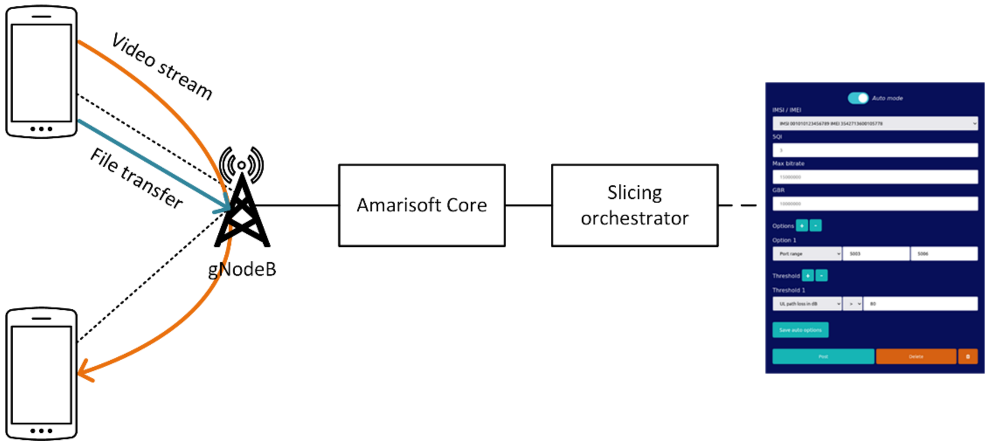
This solution is modular and can be used at the national infrastructure, HQ level or on the battlefield or dedicated to specific submodule of functionality.

For Fudge program and deployment constraint a submodule (Thales Slice Orchestrator) provide the capability to manage quality of service (QoS) rules based on slice management and Amarisoft network EPC constraint (EPC that has been agreed at beginning of the project), thus allowing an UE to be isolated from other UEs in its slice.

Set of capabilities has been delivered to the team allowing the management of the QoS rules. The QoS rules consist in a set of different values: the UE on which the rule is applied, the wanted 5G QoS Identifier (5QI), the allocated download and upload bitrates and the port range/type of service (ToS).

Possibility to provide manual or automatized mechanism to handle QoS Slice management orchestration.

The following figure provides the web interface of Slicing Orchestration tool.



IMSI / IMEI: IMSI 001010123456789 IMEI 3542713600105778

5QI: 3

Max bitrate: 15000000

GBR: 10000000

Option 1: Port range 5003 5006

Threshold 1: UL path loss in dB > 80

Figure 25: Slicing Orchestrator scenario

The refreshing time at which the data is monitored can be adjusted and is set every second by default. When an anomaly is detected, the rule is activated about one second after. The effect on the connection is visible in less than 5 seconds.

For the test scenario, shown in Figure 25, two UEs are connected to the core with a video stream between them. A large data stream, like a file transfer or an iperf test, is sent from one of the UEs to the base station. The video will be disrupted, and the script will detect the anomaly. A QoS is then applied, and the video will be fluid again.

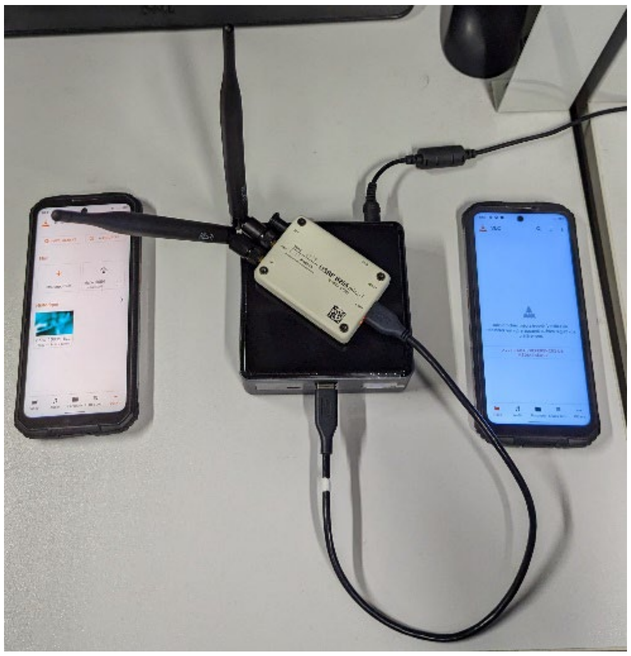


Figure 26: Scenario setup

5.3. 5GLAN

5GLAN feature was developed in Cumucore premises and tested together with 5Comm modem. After successful integration 5GLAN set up was sent to be used in Industry4.0 use case trial. 5GLAN was tested successfully using test set-up shown in diagram below.

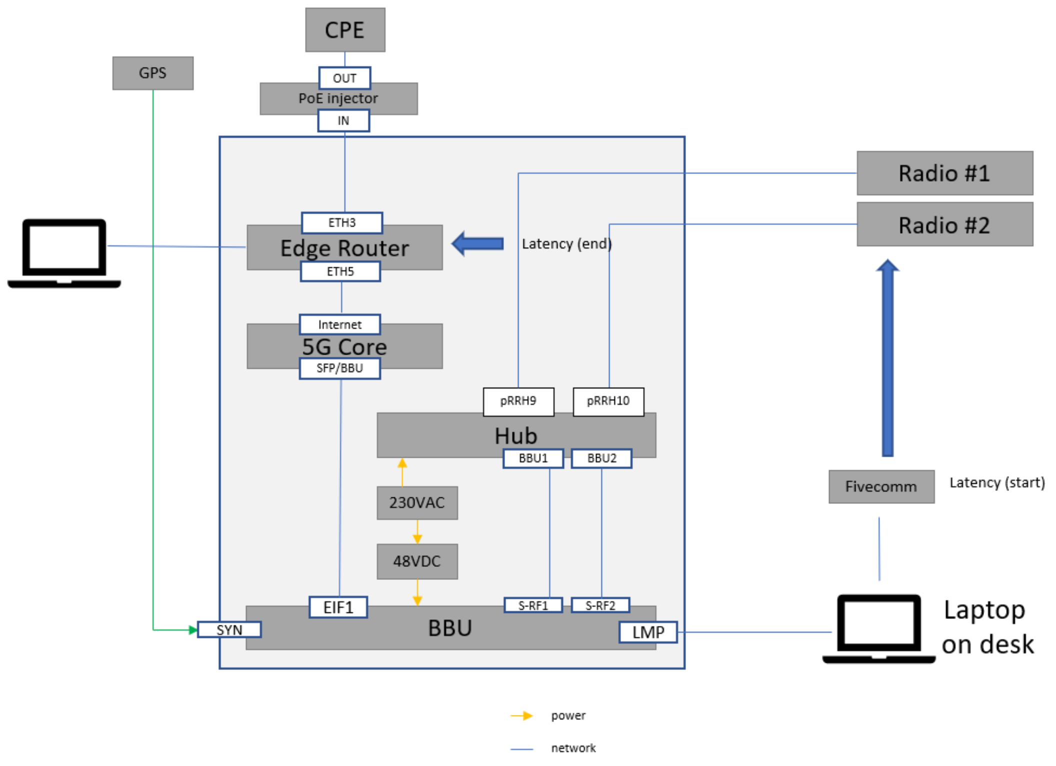


Figure 27 Industry 4.0 test set-up

In test we received following results:

Table 11: Test results

Metric	Result
Downlink speed	166Mbps
Uplink Speed	115Mbps
Ping from UE to laptop	1608/1593 0% loss ~322 seconds min/avg/max/mdev 5.0/10.7/81.7/3.7 ms

Test set-up was successfully using Profinet protocol to collect data from sensors and cameras to the automation hub and to connect motor controls using 5G air interface.

5.4. TSN

Cumucore (CMC) has been focusing on the design of 5G core specifically for Non Public Networks (NPN) primarily industrial networks that require deterministic communications following TSN features. Therefore, CMC 5G core includes the latest 3GPP standard specifications including all the necessary network functions (NF) for connecting UE devices to fixed LAN and become native TSN devices.

UE is defined to have attached functions of time stamping in data frames, using the device side of the TSN translator (DS-TT). On this Network Function, a step of time synchronisation is implemented using the TSi from the Suffix field of the gPTP messages (Sync or Follow Up messages), as it has been defined by 3GPP [11]. In order to achieve this function a 5G-Modem is integrated in a NPN following an inherent structure based on a 5G component, a TSN packet handler and wired network components. These last two components may be integrated in the same hardware such as a microprocessor but following the TSN standards defined on IEEE 802.1 Qbv [12], 802.1 CB [13], 802.1 As [14] and 802.1 Qbu [15].

In order to demonstrate the TSN functionality CMC has deployed the 5G core in ABB living labs following the topology below. In this setup we demonstrate successful time synchronization of TSN devices connected to UE device with fixed TSN devices connected to LAN.

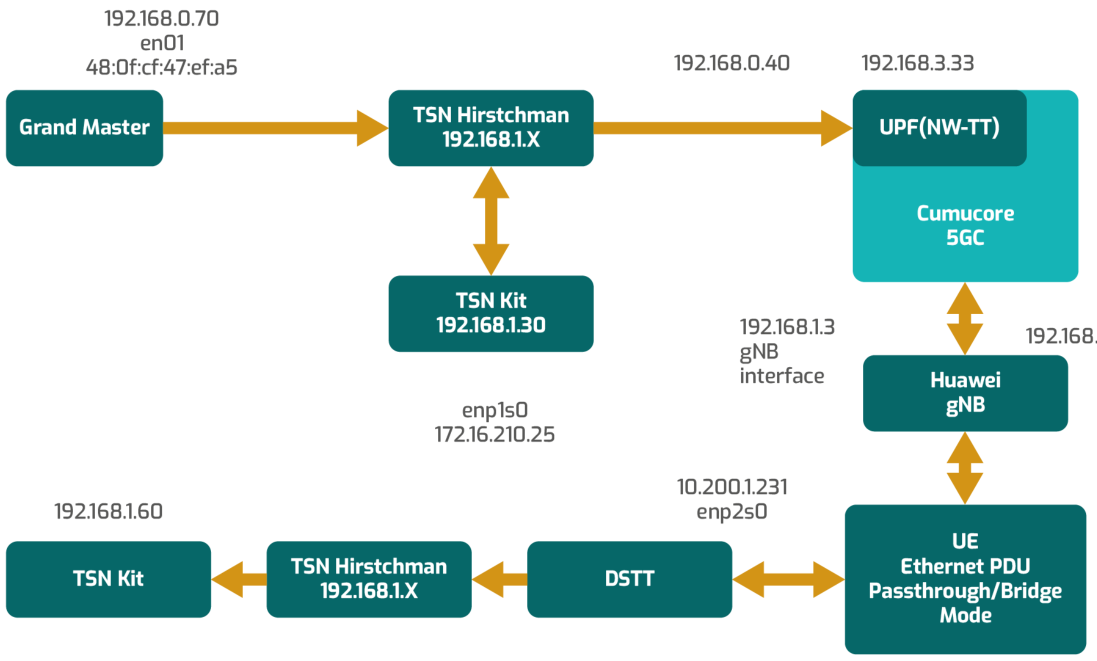


Figure 28: TSN network deployed at ABB living labs for time synchronization testing

After running the time synchronization process to get the UE in sync with the fixed devices following results were obtained.

Table 12: Time synchronization in TSN device connected UE

Location	Results
UE Local Time	1677594032.510012643

Grand Master Time	1677594032.495004032
Offset	-15008611

Table 13: Time synchronization in fixed TSN device

Location	Results
UE Local Time	1677594273.198258969
Grand Master Time	1677594273.198259137
Offset	168

The system was installed in ABB living labs as shown in the following figure for additional measurements.

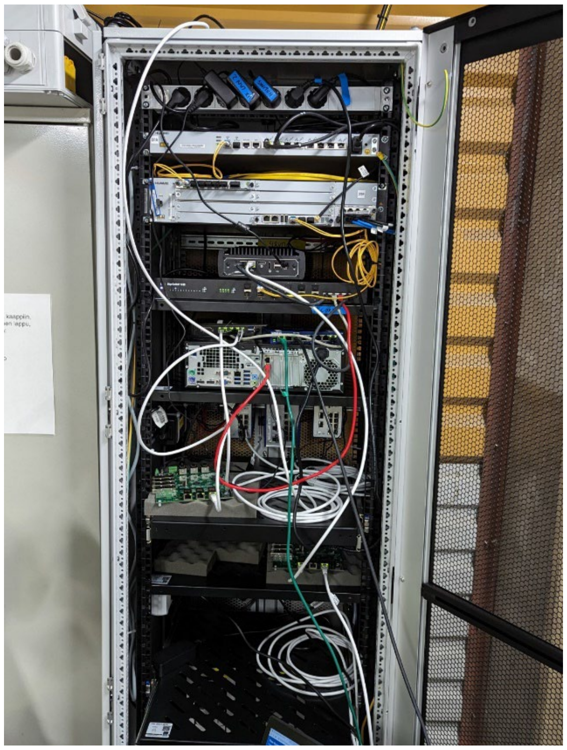


Figure 29: Equipment deployed at ABB living labs for time synchronization testing

The testing of TSN network was conducted both in CMC laboratory and in ABB living lab premises. The setup uses unique system that provides Ethernet PDU connection over 5G network. The commercial 5G networks only support IP PDU sessions for data exchange between mobile devices and data networks such as public Internet. Instead, Non-Public Networks (NPN) for industry requires Ethernet PDU together with 5GLAN functionality. This setup includes the support for Ethernet PDU which is uniquely available currently in CMC 5G Core. The Ethernet PDU allows to transfer gPTP messages from Grand Master (GM) located in fixed LAN to the 5G modem which will send the gPTP messages to DS-TT for synchronizing the moving devices.

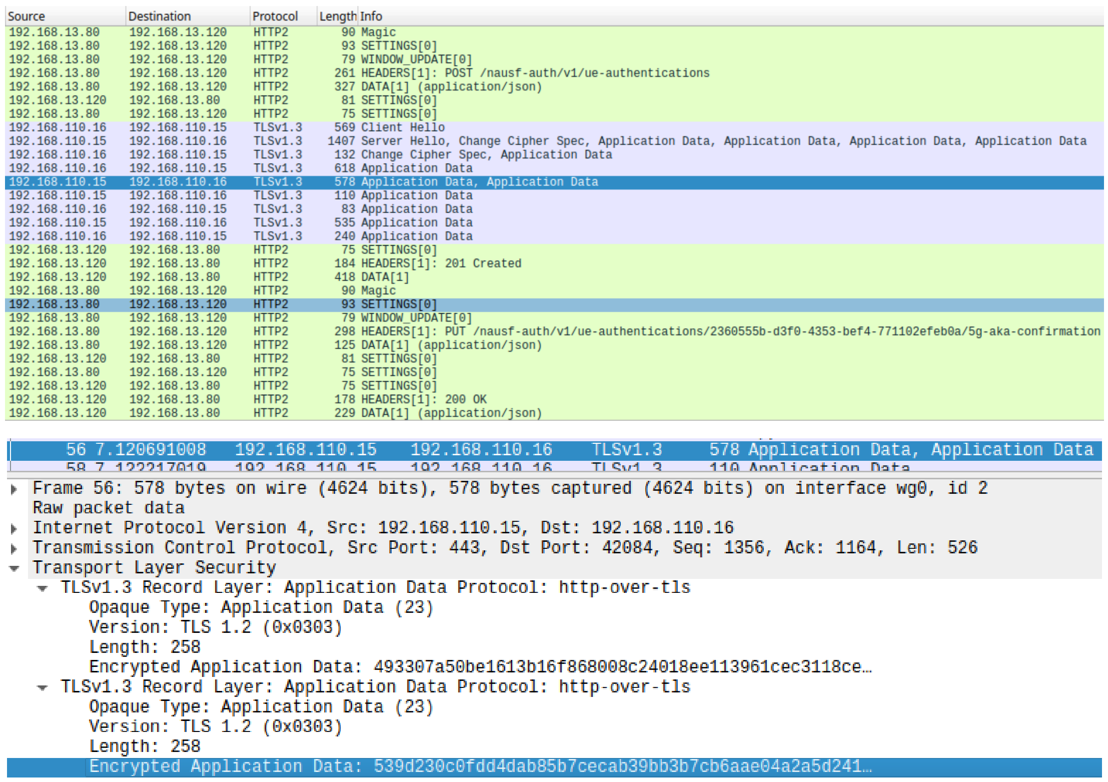
The preliminary results are shown in Table 12 and Table 13 where the mobile TSN device is synchronized but with different offset compared to the synchronization achieved by the fixed TSN device connected directly to the GM in the fixed LAN. The deployment in ABB labs had few limitations that were blocking the gPTP messages to reach the mobile TSN device. The end result is that gPTP was running over the 5G network using the Ethernet PDU, but the base station disconnected the mobile and synchronization was lost.

This behavior of the base station caused that gPTP messages did not reach the mobile TSN and went out of synch. Therefore, for providing reliable time synchronization the Ethernet PDU session should be supported, not only in the 5G Core, but also in the base station and 5G routers. Moreover, the jitter of the delay in the 5G radio link needs to be minimized to ensure the offset of the clocks in both mobile and fixed TSN are aligned.

Thus, initial results show the gPTP can be transferred over the 5G radio to certain degree of accuracy to synchronize mobile devices but still requires improvements to support natively Ethernet PDU and low delay jitter for reaching high levels of synchronization accuracy.

5.5. SBC

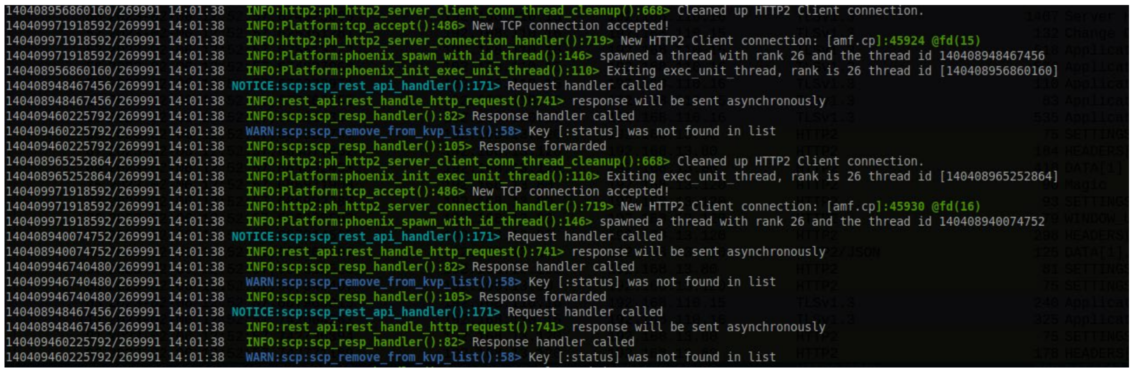
Session Border Controller (SBC) was developed in FUDGE, to exchange secured messages between interconnected small sized non-public networks. SBC has the functionalities of SCP for proxying and SEPP for encrypting messages. As part of the validation of the component registrations for roaming UEs were performed. The overhead introduced by SBC in the procedures was in average between 3-8 ms. Below in Figure 28, the traces for authentication request from FOKUS to UPV NPN is shown, transport layer security is applied on the communication between two domains resulting into encrypted message exchange. In Figure 29, the log shows SCP at FOKUS side received the authentication request from AMF, which it forwards to the UPV SCP that is shown by the log in Figure 30. Once the roaming UE is authenticated by the home network, it will get registered in the visited network. In Figure 31, the UE from UPV is registered at the network of FOKUS is shown using the command output in AMF.



```
Source          Destination     Protocol  Length Info
192.168.13.80   192.168.13.120  HTTP2        90 Magic
192.168.13.80   192.168.13.120  HTTP2        93 SETTINGS[0]
192.168.13.80   192.168.13.120  HTTP2        79 WINDOW_UPDATE[0]
192.168.13.80   192.168.13.120  HTTP2       261 HEADERS[1]: POST /nausf-auth/v1/ue-authentications
192.168.13.80   192.168.13.120  HTTP2       327 DATA[1] (application/json)
192.168.13.120  192.168.13.80   HTTP2        81 SETTINGS[0]
192.168.13.80   192.168.13.120  HTTP2        75 SETTINGS[0]
192.168.110.16  192.168.110.15  TLSv1.3     569 Client Hello
192.168.110.15  192.168.110.16  TLSv1.3    1407 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application Data
192.168.110.16  192.168.110.15  TLSv1.3     132 Change Cipher Spec, Application Data
192.168.110.16  192.168.110.15  TLSv1.3     618 Application Data
192.168.110.15  192.168.110.16  TLSv1.3     578 Application Data, Application Data
192.168.110.15  192.168.110.16  TLSv1.3     110 Application Data
192.168.110.16  192.168.110.15  TLSv1.3      83 Application Data
192.168.110.15  192.168.110.16  TLSv1.3     535 Application Data
192.168.110.16  192.168.110.15  TLSv1.3     240 Application Data
192.168.13.120  192.168.13.80   HTTP2        75 SETTINGS[0]
192.168.13.120  192.168.13.80   HTTP2       184 HEADERS[1]: 201 Created
192.168.13.120  192.168.13.80   HTTP2       418 DATA[1]
192.168.13.80   192.168.13.120  HTTP2        90 Magic
192.168.13.80   192.168.13.120  HTTP2        93 SETTINGS[0]
192.168.13.80   192.168.13.120  HTTP2        79 WINDOW_UPDATE[0]
192.168.13.80   192.168.13.120  HTTP2       298 HEADERS[1]: PUT /nausf-auth/v1/ue-authentications/2360555b-d3f0-4353-bef4-771102efeb0a/5g-aka-confirmation
192.168.13.80   192.168.13.120  HTTP2       125 DATA[1] (application/json)
192.168.13.120  192.168.13.80   HTTP2        81 SETTINGS[0]
192.168.13.80   192.168.13.120  HTTP2        75 SETTINGS[0]
192.168.13.120  192.168.13.80   HTTP2        75 SETTINGS[0]
192.168.13.120  192.168.13.80   HTTP2       178 HEADERS[1]: 200 OK
192.168.13.120  192.168.13.80   HTTP2       229 DATA[1] (application/json)
```

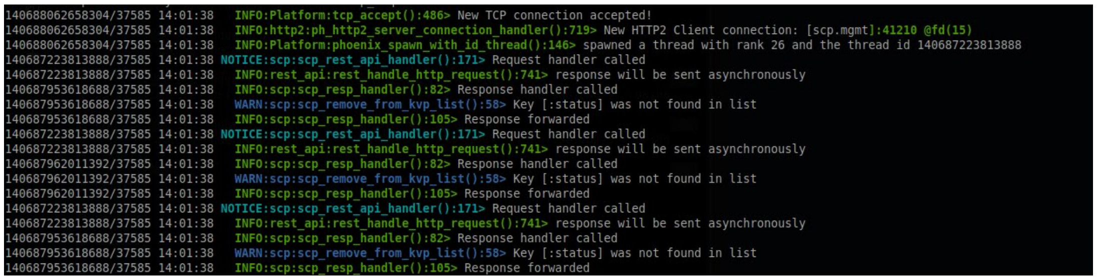
```
     56 7.120691008   192.168.110.15   192.168.110.16   TLSv1.3    578 Application Data, Application Data
     58 7.122217019   192.168.110.15   192.168.110.16   TLSv1.3    110 Application Data
▸ Frame 56: 578 bytes on wire (4624 bits), 578 bytes captured (4624 bits) on interface wg0, id 2
  Raw packet data
▸ Internet Protocol Version 4, Src: 192.168.110.15, Dst: 192.168.110.16
▸ Transmission Control Protocol, Src Port: 443, Dst Port: 42084, Seq: 1356, Ack: 1164, Len: 526
▾ Transport Layer Security
  ▾ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 258
      Encrypted Application Data: 493307a50be1613b16f868008c24018ee113961cec3118ce…
  ▾ TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 258
      Encrypted Application Data: 539d230c0fdd4dab85b7cecab39bb3b7cb6aae04a2a5d241…
```

Figure 30: Trace for encrypted message exchange from FOKUS NPN to UPV NPN through SBC



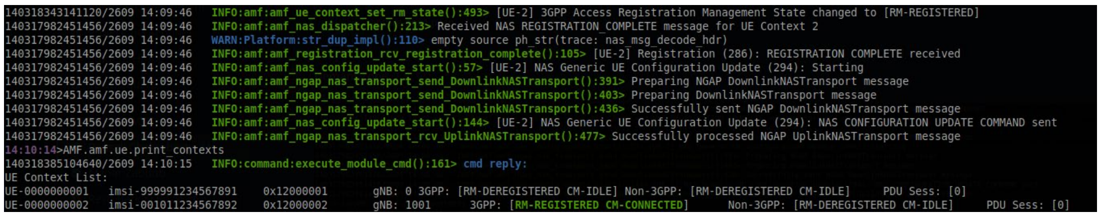
```
140408956860160/269991 14:01:38   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
140409971918592/269991 14:01:38   INFO:Platform:tcp_accept():486> New TCP connection accepted!
140409971918592/269991 14:01:38   INFO:http2:ph_http2_server_connection_handler():719> New HTTP2 Client connection: [amf.cp]:45924 @fd(15)
140409971918592/269991 14:01:38   INFO:Platform:phoenix_spawn_with_id_thread():146> spawned a thread with rank 26 and the thread id 140408948467456
140408956860160/269991 14:01:38   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [140408956860160]
140408948467456/269991 14:01:38 NOTICE:scp:scp_rest_api_handler():171> Request handler called
140408948467456/269991 14:01:38   INFO:rest_api:rest_handle_http_request():741> response will be sent asynchronously
140409460225792/269991 14:01:38   INFO:scp:scp_resp_handler():82> Response handler called
140409460225792/269991 14:01:38   WARN:scp:scp_remove_from_kvp_list():58> Key [:status] was not found in list
140409460225792/269991 14:01:38   INFO:scp:scp_resp_handler():105> Response forwarded
140408965252864/269991 14:01:38   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
140408965252864/269991 14:01:38   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [140408965252864]
140409971918592/269991 14:01:38   INFO:Platform:tcp_accept():486> New TCP connection accepted!
140409971918592/269991 14:01:38   INFO:http2:ph_http2_server_connection_handler():719> New HTTP2 Client connection: [amf.cp]:45930 @fd(16)
140409971918592/269991 14:01:38   INFO:Platform:phoenix_spawn_with_id_thread():146> spawned a thread with rank 26 and the thread id 140408940074752
140408940074752/269991 14:01:38 NOTICE:scp:scp_rest_api_handler():171> Request handler called
140408940074752/269991 14:01:38   INFO:rest_api:rest_handle_http_request():741> response will be sent asynchronously
140409946740480/269991 14:01:38   INFO:scp:scp_resp_handler():82> Response handler called
140409946740480/269991 14:01:38   WARN:scp:scp_remove_from_kvp_list():58> Key [:status] was not found in list
140409946740480/269991 14:01:38   INFO:scp:scp_resp_handler():105> Response forwarded
140408948467456/269991 14:01:38 NOTICE:scp:scp_rest_api_handler():171> Request handler called
140408948467456/269991 14:01:38   INFO:rest_api:rest_handle_http_request():741> response will be sent asynchronously
140409460225792/269991 14:01:38   INFO:scp:scp_resp_handler():82> Response handler called
140409460225792/269991 14:01:38   WARN:scp:scp_remove_from_kvp_list():58> Key [:status] was not found in list
```

Figure 31: FOKUS SCP received message from AMF for authentication



```
140688062658304/37585 14:01:38   INFO:Platform:tcp_accept():486> New TCP connection accepted!
140688062658304/37585 14:01:38   INFO:http2:ph_http2_server_connection_handler():719> New HTTP2 Client connection: [scp.mgmt]:41210 @fd(15)
140688062658304/37585 14:01:38   INFO:Platform:phoenix_spawn_with_id_thread():146> spawned a thread with rank 26 and the thread id 140687223813888
140687223813888/37585 14:01:38 NOTICE:scp:scp_rest_api_handler():171> Request handler called
140687223813888/37585 14:01:38   INFO:rest_api:rest_handle_http_request():741> response will be sent asynchronously
140687953618688/37585 14:01:38   INFO:scp:scp_resp_handler():82> Response handler called
140687953618688/37585 14:01:38   WARN:scp:scp_remove_from_kvp_list():58> Key [:status] was not found in list
140687953618688/37585 14:01:38   INFO:scp:scp_resp_handler():105> Response forwarded
140687223813888/37585 14:01:38 NOTICE:scp:scp_rest_api_handler():171> Request handler called
140687223813888/37585 14:01:38   INFO:rest_api:rest_handle_http_request():741> response will be sent asynchronously
140687962011392/37585 14:01:38   INFO:scp:scp_resp_handler():82> Response handler called
140687962011392/37585 14:01:38   WARN:scp:scp_remove_from_kvp_list():58> Key [:status] was not found in list
140687962011392/37585 14:01:38   INFO:scp:scp_resp_handler():105> Response forwarded
140687223813888/37585 14:01:38 NOTICE:scp:scp_rest_api_handler():171> Request handler called
140687223813888/37585 14:01:38   INFO:rest_api:rest_handle_http_request():741> response will be sent asynchronously
140687953618688/37585 14:01:38   INFO:scp:scp_resp_handler():82> Response handler called
140687953618688/37585 14:01:38   WARN:scp:scp_remove_from_kvp_list():58> Key [:status] was not found in list
140687953618688/37585 14:01:38   INFO:scp:scp_resp_handler():105> Response forwarded
```

Figure 32: UPV SCP received message from FOKUS SCP for authentication



```
140318343141120/2609 14:09:46   INFO:amf:amf_ue_context_set_rm_state():493> [UE-2] 3GPP Access Registration Management State changed to [RM-REGISTERED]
140317982451456/2609 14:09:46   INFO:amf:amf_nas_dispatcher():213> Received NAS REGISTRATION_COMPLETE message for UE Context 2
140317982451456/2609 14:09:46   WARN:Platform:str_dup_impl():110> empty source ph_str(trace: nas_msg_decode_hdr)
140317982451456/2609 14:09:46   INFO:amf:amf_registration_rcv_registration_complete():105> [UE-2] Registration (286): REGISTRATION COMPLETE received
140317982451456/2609 14:09:46   INFO:amf:amf_nas_config_update_start():57> [UE-2] NAS Generic UE Configuration Update (294): Starting
140317982451456/2609 14:09:46   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():391> Preparing NGAP DownlinkNASTransport message
140317982451456/2609 14:09:46   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():403> Preparing DownlinkNASTransport message
140317982451456/2609 14:09:46   INFO:amf:amf_ngap_nas_transport_send_DownlinkNASTransport():436> Successfully sent NGAP DownlinkNASTransport message
140317982451456/2609 14:09:46   INFO:amf:amf_nas_config_update_start():144> [UE-2] NAS Generic UE Configuration Update (294): NAS CONFIGURATION UPDATE COMMAND sent
140317982451456/2609 14:09:46   INFO:amf:amf_ngap_nas_transport_rcv_UplinkNASTransport():477> Successfully processed NGAP UplinkNASTransport message
14:10:14>AMF.amf.ue.print_contexts
140318385104640/2609 14:10:15   INFO:command:execute_module_cmd():161> cmd reply:
UE Context List:
UE-0000000001   imsi-999991234567891    0x12000001       gNB: 0 3GPP: [RM-DEREGISTERED CM-IDLE] Non-3GPP: [RM-DEREGISTERED CM-IDLE]     PDU Sess: [0]
UE-0000000002   imsi-001011234567892    0x12000002       gNB: 1001       3GPP: [RM-REGISTERED CM-CONNECTED]      Non-3GPP: [RM-DEREGISTERED CM-IDLE]     PDU Sess: [0]
```

Figure 33: UE belongs to UPV network registered and connected FOKUS network

5.6. Multicast

UPV has developed a prototype or minimum viable product to include the multicast/broadcast feature in the 5G Core, also known as 5G Multicast/Broadcast Services or 5MBS. This prototype is located in UPV premises and it is based on an extension of the FOKUS Open5GCore. Some initial tests using iperf have been already carried out and documented in D2.5 [16], which showed a small degradation of the multicast NFs when compared against their unicast counterparts. The overall prototype diagram can be seen in the figure below:

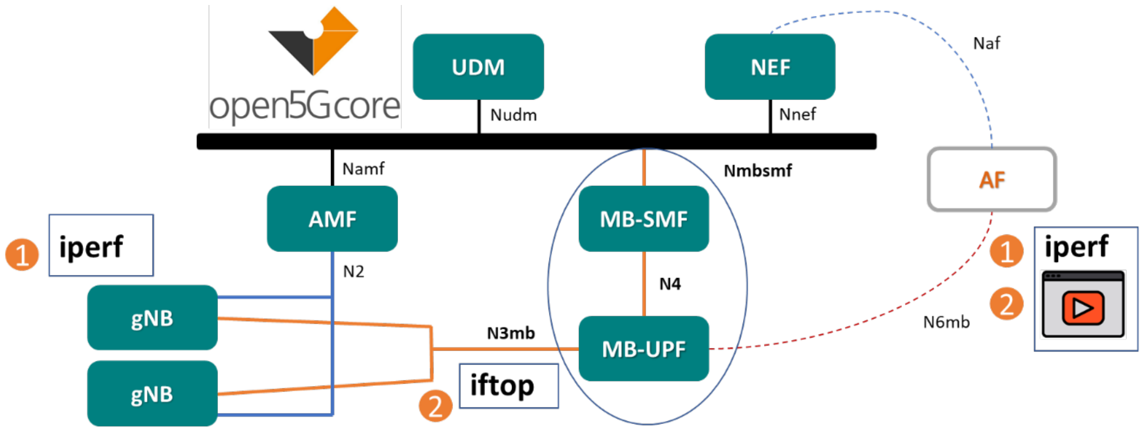


Figure 34: 5MBS prototype diagram, with the two tests carried out: one using iperf and documented in D2.5, and the other featuring video delivery and measuring the bandwidth over N3mb.

To validate the advantages of multicast i.e. that the number of users consuming multicast content has no effect on the total bandwidth coursed through the core; a video delivery experiment using ffmpeg has been devised, using a variable of virtualized users and gNBs in different topologies. The video delivery experiments are composed of two main tests, one having a 1 to 1 mapping between UE and gNB e.g. 1 UE, 1 gNB experiment; 2 UEs, 2 gNBs… and the other leaving one gNB fixed and varying the number of UEs. The traffic is measured over the N3mb interface, using the Linux console tool iftop, then the output is parsed and processed.

The first video is a 4K quality recorded video from a smartphone, with 2:47 duration and 6 Mb/s average bit rate. The video is sent twice, with a manual restart, to see if the tool is correctly capturing the traffic over N3mb.

The first video is a 4K quality recorded video from a smartphone, with 2:47 duration and 6 Mb/s average bit rate. The video is sent twice, with a manual restart, to see if the tool is correctly capturing the traffic over N3mb. The test has been carried out for 1, 2 and 3 UEs-gNBs pairs. Figure 33 contains the obtained graphs:

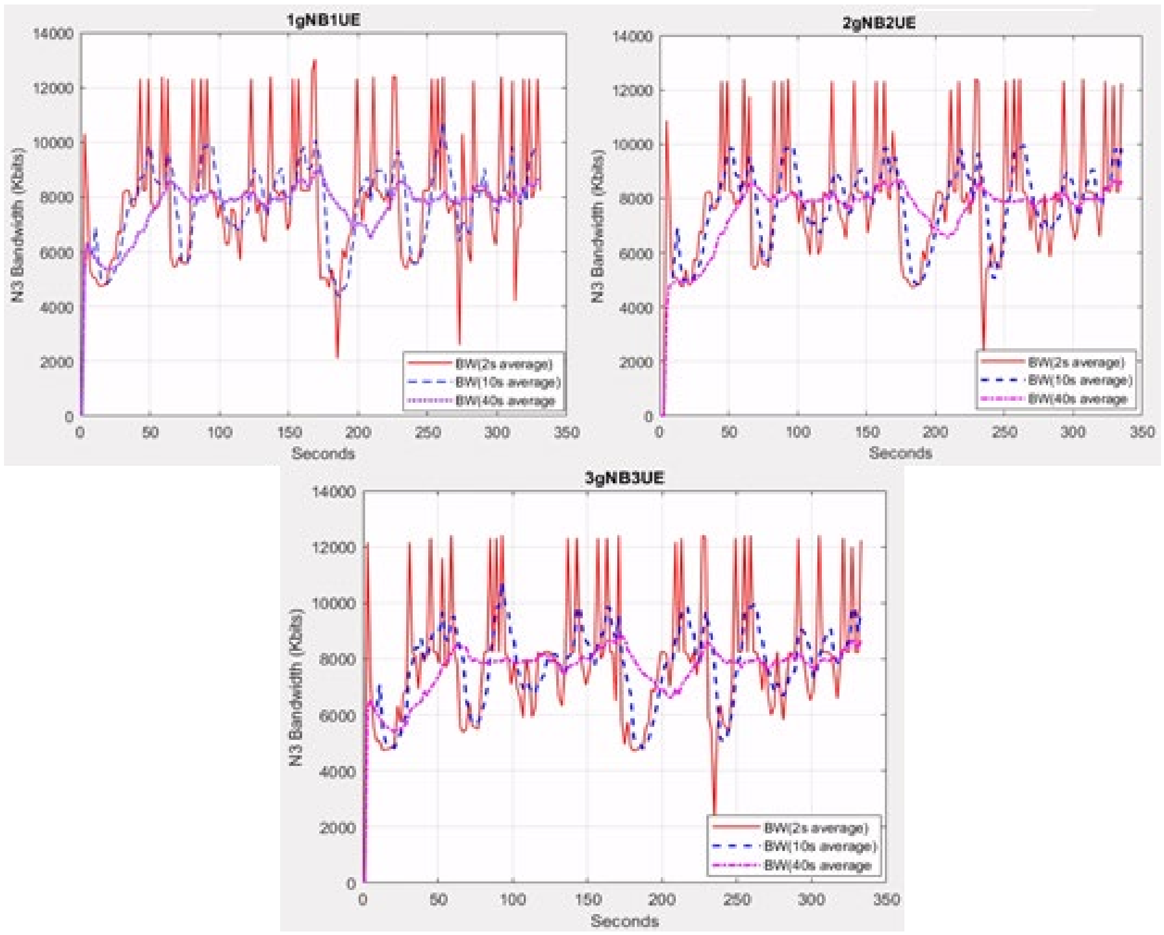


Figure 35: Bandwidth captured over the N3mb of the 5MBS prototype, with a variable number of UEs.

It can be concluded that the prototype is correctly sending the data in a point-to-multipoint mode over the 5G Core, as the graphs are very similar even if the number of UEs is variable. However, it can be seen that the bandwidth is saturating at 12 Mb/s, as all the peaks fall into the same range. In other words, the Multicast/Broadcast UPF is dropping video packets when this value is reached. The reasons for this incorrect performance have not been identified, as the prototype is implemented over several abstractions, including virtualized resources over a hypervisor.

The second video delivery test considers this newfound saturation threshold. The video sent is in lower quality, recorded with a smartphone camera, with an average bitrate of 1.6 Mb/s. In this test, 2 different flows are launched to 2 UEs over multicast, in two different experiments: one using 1 gNB and the other using 2 gNBs. The video is also launched twice for every service to increase the duration. The results are shown in the Figure below:

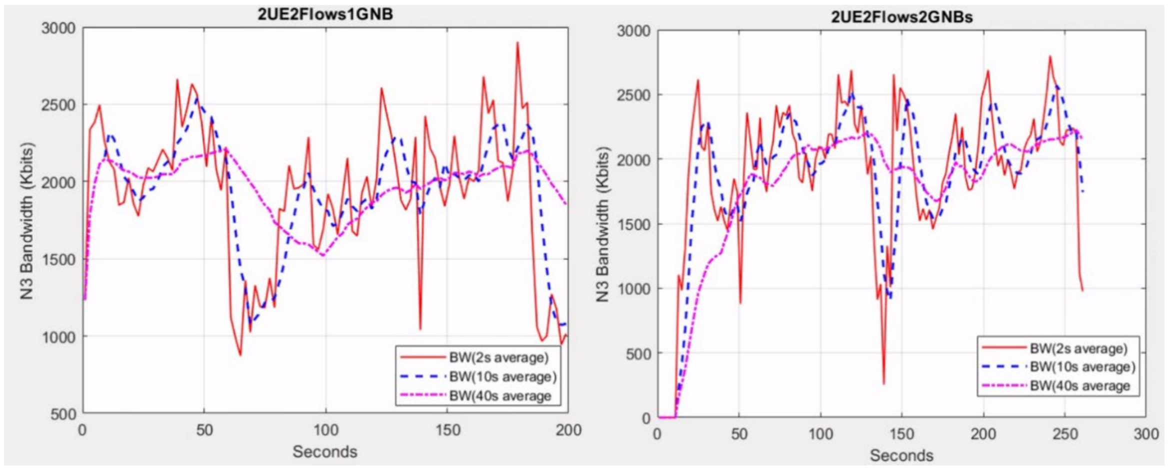


Figure 36: Video delivery experiment over 5MBS, using 2 Multicast flows to 2 UEs who are attached to 1 gNB (left) and 2 gNBs (right).

From the graphs, it can be derived that they are significantly different between them. This is introduced due to the human factor, where each service is launched and stopped manually for every setup, creating a new realization of the experiment and component under test. Nevertheless, the peaks still reach around 2.5 Mb/s for both cases and there is a significant bandwidth drop around 150 seconds which is the duration of the original video, before it gets relaunched again. It can be concluded that varying the number of gNBs has no effect on the multicast properties of the prototype.

This component is in an experimental phase has not been used in any use case of FUDGE-5G.

6. Conclusions

This deliverable provided a final report on the validation of FUDGE-5G components with vertical trials. Although some of the components do not have an extensive validation procedure, most of the validations were carried out in field trials. In trial validations, sometimes can be difficult to separate and obtain data for each component, as it is much more intuitive and meaningful to collect KPIs for the entire trial platform setup. Despite that barrier, throughout the previous sections, and for each component, the document highlighted how the validation work has been performed, what was collected, what were the outcomes and results and what they mean in terms of validation. Later, in D4.3, further results will be reported, giving more focus on the validation of the trials as a whole and not on single components.

7. References

[1] FUDGE-5G Consortium, “D1.3: FUDGE-5G Platform Architecture Final Release,” 2022.

[2] 3GPP, “TS 23.280: Common functional architecture to support mission critical services; Stage 2,” May 2022. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.280/23280-h90.zip.

[3] FUDGE-5G Consortium, “D3.2: FUDGE-5G On-boarding and Deploying of the Vertical Use Cases,” 2023.

[4] FUDGE-5G Consortium, “D4.3: Final Technical Validation of 5G Components with Vertical Trial,” 2023.

[5] 3GPP, “TS 23.502: Procedures for the 5G System (5GS),” December 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123502/17.07.00_60/ts_123502v170700p.pdf.

[6] FUDGE-5G Consortium, “D1.2: FUDGE-5G Platform Architecture: Components and Interfaces,” 2021.

[7] FUDGE-5G Consortium, “D2.1: FUDGE-5G Technology Components and Platform – Interim Release,” 2021.

[8] FUDGE-5G Consortium, “D2.4: Disintegrated Network Functions for Cloud-native Service Orchestration,” 2022.

[9] 3GPP, “TS 23.501: System architecture for the 5G System (5GS),” March 2022. [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/23_series/23.501/23501-f30.zip.

[10] FUDGE-5G Consortium, “D4.1: Interim Technical Validation of 5G Components with Vertical Trials,” 2022.

[11] 3GPP, “TS 24.535: 5G System (5GS); Device-Side Time Sensitive Networking (TSN) Translator (DS-TT) to Network-Side TSN Translator (NW-TT) protocol aspects; Stage 3,” November 2020. [Online]. Available: https://www.3gpp.org/ftp//Specs/archive/24_series/24.535/24535-g20.zip.

[12] *IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic, 802.1Qbv-2015,* 2015.

[13] *IEEE Standard for Local and metropolitan area networks--Frame Replication and Elimination for Reliability, 802.1CB-2017,* 2017.

[14] *IEEE Standard for Local and Metropolitan Area Networks--Timing and Synchronization for Time-Sensitive Applications, 802.1AS-2020,* 2020.

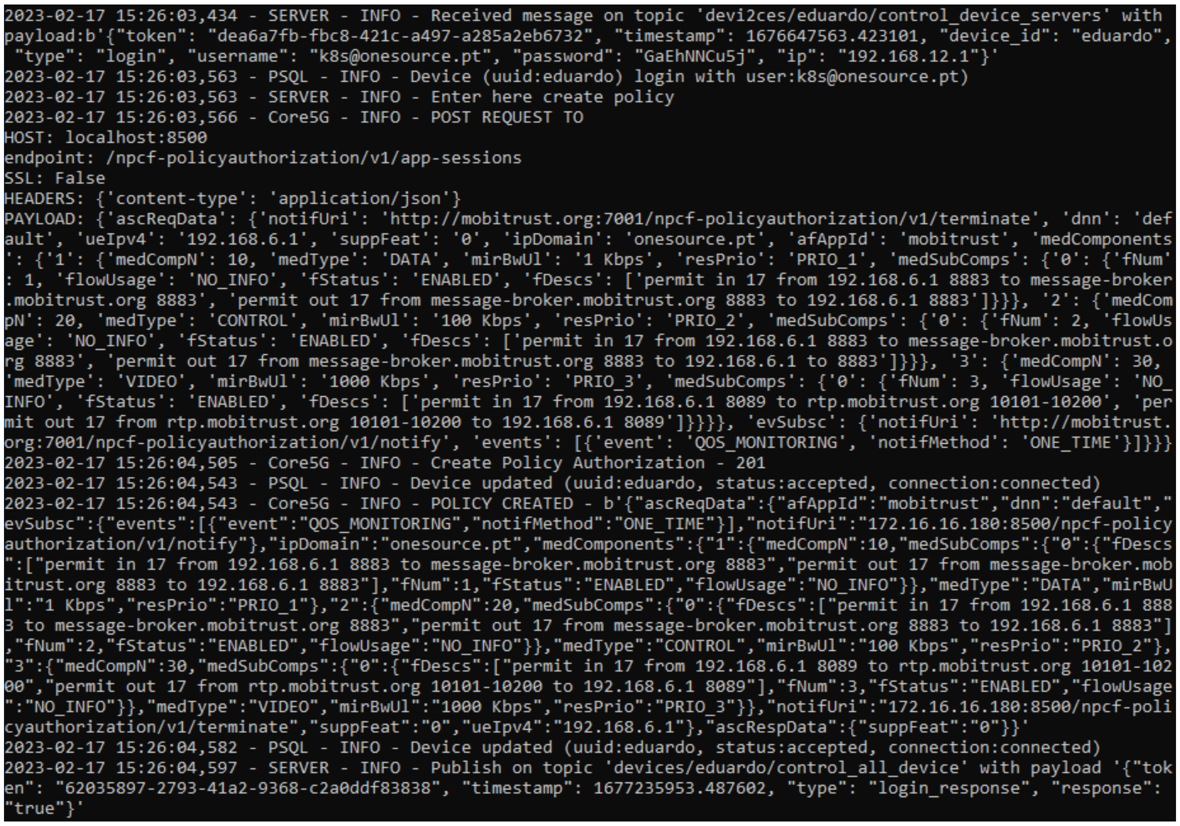
[15] *IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks -- Amendment 26: Frame Preemption, 802.1Qbu-2016,* 2016.

[16] FUDGE-5G Consortium, "D2.5: FUDGE-5G Technology Components and Platform Final Release," 2022.

8. Annex A

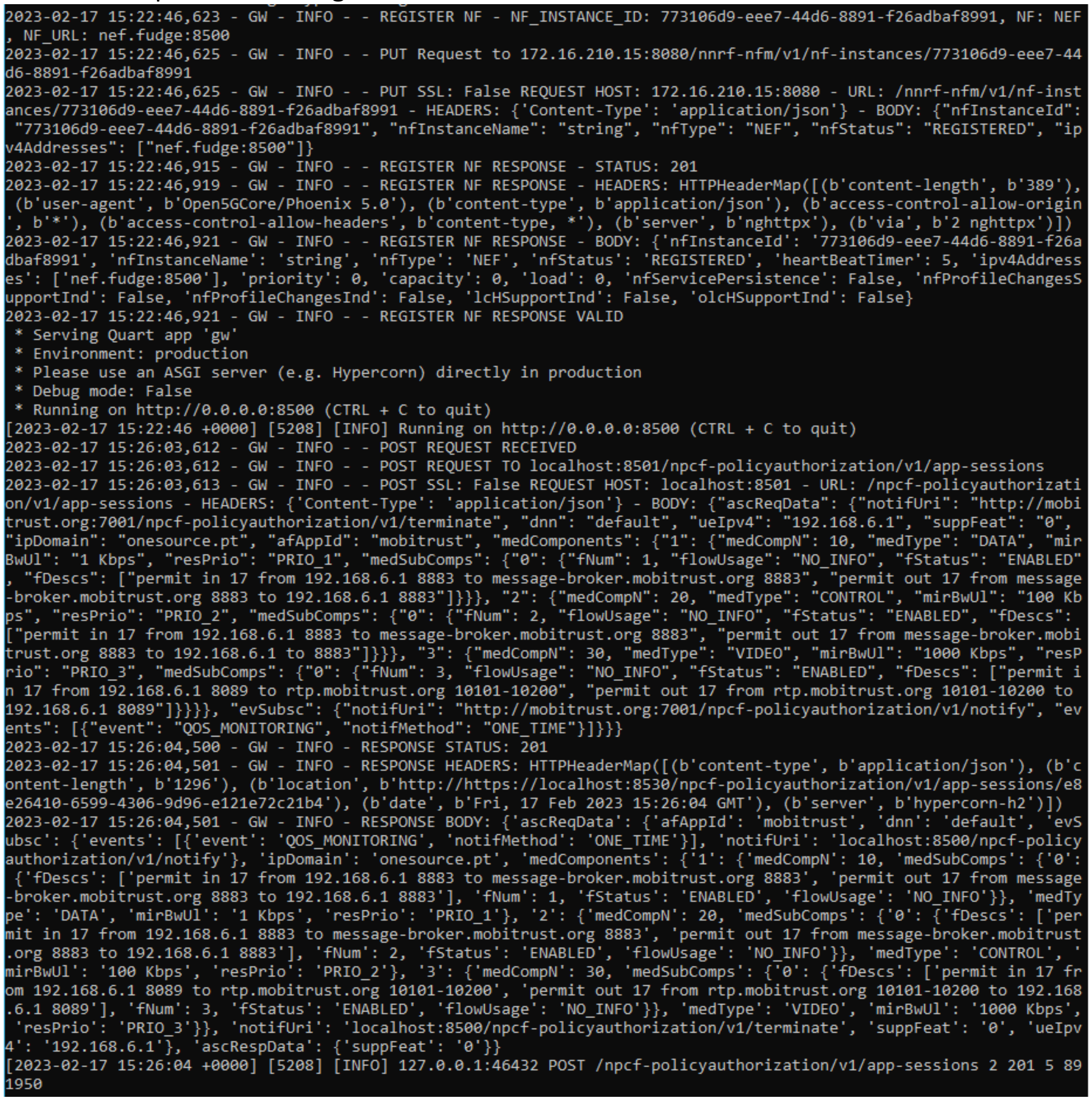
The following screenshots showcase the overall output by several NFs obtained when validating the Onesource NEF.

- Screen cap of the AF logs



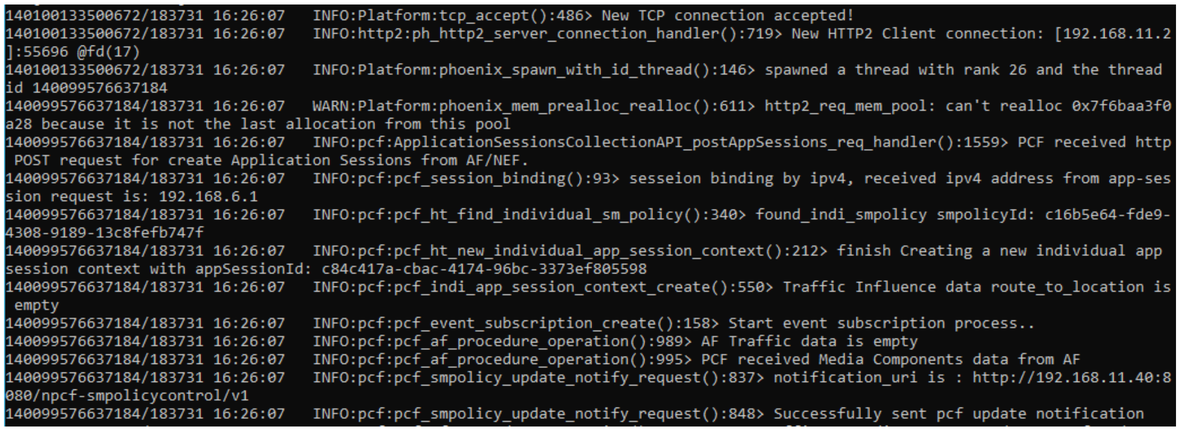
```
2023-02-17 15:26:03,434 - SERVER - INFO - Received message on topic 'devi2ces/eduardo/control_device_servers' with
payload:b'{"token": "dea6a7fb-fbc8-421c-a497-a285a2eb6732", "timestamp": 1676647563.423101, "device_id": "eduardo",
 "type": "login", "username": "k8s@onesource.pt", "password": "GaEhNNCu5j", "ip": "192.168.12.1"}'
2023-02-17 15:26:03,563 - PSQL - INFO - Device (uuid:eduardo) login with user:k8s@onesource.pt)
2023-02-17 15:26:03,563 - SERVER - INFO - Enter here create policy
2023-02-17 15:26:03,566 - Core5G - INFO - POST REQUEST TO
HOST: localhost:8500
endpoint: /npcf-policyauthorization/v1/app-sessions
SSL: False
HEADERS: {'content-type': 'application/json'}
PAYLOAD: {'ascReqData': {'notifUri': 'http://mobitrust.org:7001/npcf-policyauthorization/v1/terminate', 'dnn': 'def
ault', 'ueIpv4': '192.168.6.1', 'suppFeat': '0', 'ipDomain': 'onesource.pt', 'afAppId': 'mobitrust', 'medComponents
': {'1': {'medCompN': 10, 'medType': 'DATA', 'mirBwUl': '1 Kbps', 'resPrio': 'PRIO_1', 'medSubComps': {'0': {'fNum'
: 1, 'flowUsage': 'NO_INFO', 'fStatus': 'ENABLED', 'fDescs': ['permit in 17 from 192.168.6.1 8883 to message-broker
.mobitrust.org 8883', 'permit out 17 from message-broker.mobitrust.org 8883 to 192.168.6.1 8883']}}}, '2': {'medCom
pN': 20, 'medType': 'CONTROL', 'mirBwUl': '100 Kbps', 'resPrio': 'PRIO_2', 'medSubComps': {'0': {'fNum': 2, 'flowUs
age': 'NO_INFO', 'fStatus': 'ENABLED', 'fDescs': ['permit in 17 from 192.168.6.1 8883 to message-broker.mobitrust.o
rg 8883', 'permit out 17 from message-broker.mobitrust.org 8883 to 192.168.6.1 to 8883']}}}, '3': {'medCompN': 30,
'medType': 'VIDEO', 'mirBwUl': '1000 Kbps', 'resPrio': 'PRIO_3', 'medSubComps': {'0': {'fNum': 3, 'flowUsage': 'NO_
INFO', 'fStatus': 'ENABLED', 'fDescs': ['permit in 17 from 192.168.6.1 8089 to rtp.mobitrust.org 10101-10200', 'per
mit out 17 from rtp.mobitrust.org 10101-10200 to 192.168.6.1 8089']}}}}, 'evSubsc': {'notifUri': 'http://mobitrust.
org:7001/npcf-policyauthorization/v1/notify', 'events': [{'event': 'QOS_MONITORING', 'notifMethod': 'ONE_TIME'}]}}}
2023-02-17 15:26:04,505 - Core5G - INFO - Create Policy Authorization - 201
2023-02-17 15:26:04,543 - PSQL - INFO - Device updated (uuid:eduardo, status:accepted, connection:connected)
2023-02-17 15:26:04,543 - Core5G - INFO - POLICY CREATED - b'{"ascReqData":{"afAppId":"mobitrust","dnn":"default","
evSubsc":{"events":[{"event":"QOS_MONITORING","notifMethod":"ONE_TIME"}],"notifUri":"172.16.16.180:8500/npcf-policy
authorization/v1/notify"},"ipDomain":"onesource.pt","medComponents":{"1":{"medCompN":10,"medSubComps":{"0":{"fDescs
":["permit in 17 from 192.168.6.1 8883 to message-broker.mobitrust.org 8883","permit out 17 from message-broker.mob
itrust.org 8883 to 192.168.6.1 8883"],"fNum":1,"fStatus":"ENABLED","flowUsage":"NO_INFO"}},"medType":"DATA","mirBwU
l":"1 Kbps","resPrio":"PRIO_1"},"2":{"medCompN":20,"medSubComps":{"0":{"fDescs":["permit in 17 from 192.168.6.1 888
3 to message-broker.mobitrust.org 8883","permit out 17 from message-broker.mobitrust.org 8883 to 192.168.6.1 8883"]
,"fNum":2,"fStatus":"ENABLED","flowUsage":"NO_INFO"}},"medType":"CONTROL","mirBwUl":"100 Kbps","resPrio":"PRIO_2"},
"3":{"medCompN":30,"medSubComps":{"0":{"fDescs":["permit in 17 from 192.168.6.1 8089 to rtp.mobitrust.org 10101-102
00","permit out 17 from rtp.mobitrust.org 10101-10200 to 192.168.6.1 8089"],"fNum":3,"fStatus":"ENABLED","flowUsage
":"NO_INFO"}},"medType":"VIDEO","mirBwUl":"1000 Kbps","resPrio":"PRIO_3"}},"notifUri":"172.16.16.180:8500/npcf-poli
cyauthorization/v1/terminate","suppFeat":"0","ueIpv4":"192.168.6.1"},"ascRespData":{"suppFeat":"0"}}'
2023-02-17 15:26:04,582 - PSQL - INFO - Device updated (uuid:eduardo, status:accepted, connection:connected)
2023-02-17 15:26:04,597 - SERVER - INFO - Publish on topic 'devices/eduardo/control_all_device' with payload '{"tok
en": "62035897-2793-41a2-9368-c2a0ddf83838", "timestamp": 1677235953.487602, "type": "login_response", "response":
"true"}'
```

- Screencap of the NEF logs



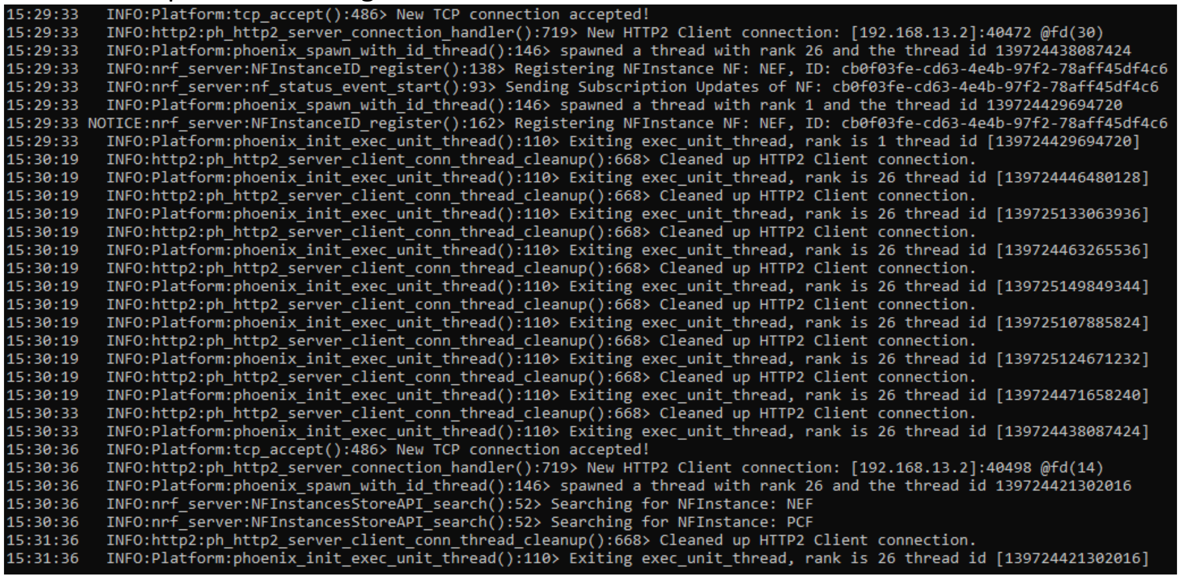
```
2023-02-17 15:22:46,623 - GW - INFO - - REGISTER NF - NF_INSTANCE_ID: 773106d9-eee7-44d6-8891-f26adbaf8991, NF: NEF
, NF_URL: nef.fudge:8500
2023-02-17 15:22:46,625 - GW - INFO - - PUT Request to 172.16.210.15:8080/nnrf-nfm/v1/nf-instances/773106d9-eee7-44
d6-8891-f26adbaf8991
2023-02-17 15:22:46,625 - GW - INFO - - PUT SSL: False REQUEST HOST: 172.16.210.15:8080 - URL: /nnrf-nfm/v1/nf-inst
ances/773106d9-eee7-44d6-8891-f26adbaf8991 - HEADERS: {'Content-Type': 'application/json'} - BODY: {"nfInstanceId":
 "773106d9-eee7-44d6-8891-f26adbaf8991", "nfInstanceName": "string", "nfType": "NEF", "nfStatus": "REGISTERED", "ip
v4Addresses": ["nef.fudge:8500"]}
2023-02-17 15:22:46,915 - GW - INFO - - REGISTER NF RESPONSE - STATUS: 201
2023-02-17 15:22:46,919 - GW - INFO - - REGISTER NF RESPONSE - HEADERS: HTTPHeaderMap([(b'content-length', b'389'),
 (b'user-agent', b'Open5GCore/Phoenix 5.0'), (b'content-type', b'application/json'), (b'access-control-allow-origin
', b'*'), (b'access-control-allow-headers', b'content-type, *'), (b'server', b'nghttpx'), (b'via', b'2 nghttpx')])
2023-02-17 15:22:46,921 - GW - INFO - - REGISTER NF RESPONSE - BODY: {'nfInstanceId': '773106d9-eee7-44d6-8891-f26a
dbaf8991', 'nfInstanceName': 'string', 'nfType': 'NEF', 'nfStatus': 'REGISTERED', 'heartBeatTimer': 5, 'ipv4Address
es': ['nef.fudge:8500'], 'priority': 0, 'capacity': 0, 'load': 0, 'nfServicePersistence': False, 'nfProfileChangesS
upportInd': False, 'nfProfileChangesInd': False, 'lcHSupportInd': False, 'olcHSupportInd': False}
2023-02-17 15:22:46,921 - GW - INFO - - REGISTER NF RESPONSE VALID
 * Serving Quart app 'gw'
 * Environment: production
 * Please use an ASGI server (e.g. Hypercorn) directly in production
 * Debug mode: False
 * Running on http://0.0.0.0:8500 (CTRL + C to quit)
[2023-02-17 15:22:46 +0000] [5208] [INFO] Running on http://0.0.0.0:8500 (CTRL + C to quit)
2023-02-17 15:26:03,612 - GW - INFO - - POST REQUEST RECEIVED
2023-02-17 15:26:03,612 - GW - INFO - - POST REQUEST TO localhost:8501/npcf-policyauthorization/v1/app-sessions
2023-02-17 15:26:03,613 - GW - INFO - - POST SSL: False REQUEST HOST: localhost:8501 - URL: /npcf-policyauthorizati
on/v1/app-sessions - HEADERS: {'Content-Type': 'application/json'} - BODY: {"ascReqData": {"notifUri": "http://mobi
trust.org:7001/npcf-policyauthorization/v1/terminate", "dnn": "default", "ueIpv4": "192.168.6.1", "suppFeat": "0",
"ipDomain": "onesource.pt", "afAppId": "mobitrust", "medComponents": {"1": {"medCompN": 10, "medType": "DATA", "mir
BwUl": "1 Kbps", "resPrio": "PRIO_1", "medSubComps": {"0": {"fNum": 1, "flowUsage": "NO_INFO", "fStatus": "ENABLED"
, "fDescs": ["permit in 17 from 192.168.6.1 8883 to message-broker.mobitrust.org 8883", "permit out 17 from message
-broker.mobitrust.org 8883 to 192.168.6.1 8883"]}}}, "2": {"medCompN": 20, "medType": "CONTROL", "mirBwUl": "100 Kb
ps", "resPrio": "PRIO_2", "medSubComps": {"0": {"fNum": 2, "flowUsage": "NO_INFO", "fStatus": "ENABLED", "fDescs":
["permit in 17 from 192.168.6.1 8883 to message-broker.mobitrust.org 8883", "permit out 17 from message-broker.mobi
trust.org 8883 to 192.168.6.1 to 8883"]}}}, "3": {"medCompN": 30, "medType": "VIDEO", "mirBwUl": "1000 Kbps", "resP
rio": "PRIO_3", "medSubComps": {"0": {"fNum": 3, "flowUsage": "NO_INFO", "fStatus": "ENABLED", "fDescs": ["permit i
n 17 from 192.168.6.1 8089 to rtp.mobitrust.org 10101-10200", "permit out 17 from rtp.mobitrust.org 10101-10200 to
192.168.6.1 8089"]}}}}, "evSubsc": {"notifUri": "http://mobitrust.org:7001/npcf-policyauthorization/v1/notify", "ev
ents": [{"event": "QOS_MONITORING", "notifMethod": "ONE_TIME"}]}}}
2023-02-17 15:26:04,500 - GW - INFO - RESPONSE STATUS: 201
2023-02-17 15:26:04,501 - GW - INFO - RESPONSE HEADERS: HTTPHeaderMap([(b'content-type', b'application/json'), (b'c
ontent-length', b'1296'), (b'location', b'http://https://localhost:8530/npcf-policyauthorization/v1/app-sessions/e8
e26410-6599-4306-9d96-e121e72c21b4'), (b'date', b'Fri, 17 Feb 2023 15:26:04 GMT'), (b'server', b'hypercorn-h2')])
2023-02-17 15:26:04,501 - GW - INFO - RESPONSE BODY: {'ascReqData': {'afAppId': 'mobitrust', 'dnn': 'default', 'evS
ubsc': {'events': [{'event': 'QOS_MONITORING', 'notifMethod': 'ONE_TIME'}], 'notifUri': 'localhost:8500/npcf-policy
authorization/v1/notify'}, 'ipDomain': 'onesource.pt', 'medComponents': {'1': {'medCompN': 10, 'medSubComps': {'0':
 {'fDescs': ['permit in 17 from 192.168.6.1 8883 to message-broker.mobitrust.org 8883', 'permit out 17 from message
-broker.mobitrust.org 8883 to 192.168.6.1 8883'], 'fNum': 1, 'fStatus': 'ENABLED', 'flowUsage': 'NO_INFO'}}, 'medTy
pe': 'DATA', 'mirBwUl': '1 Kbps', 'resPrio': 'PRIO_1'}, '2': {'medCompN': 20, 'medSubComps': {'0': {'fDescs': ['per
mit in 17 from 192.168.6.1 8883 to message-broker.mobitrust.org 8883', 'permit out 17 from message-broker.mobitrust
.org 8883 to 192.168.6.1 8883'], 'fNum': 2, 'fStatus': 'ENABLED', 'flowUsage': 'NO_INFO'}}, 'medType': 'CONTROL', '
mirBwUl': '100 Kbps', 'resPrio': 'PRIO_2'}, '3': {'medCompN': 30, 'medSubComps': {'0': {'fDescs': ['permit in 17 fr
om 192.168.6.1 8089 to rtp.mobitrust.org 10101-10200', 'permit out 17 from rtp.mobitrust.org 10101-10200 to 192.168
.6.1 8089'], 'fNum': 3, 'fStatus': 'ENABLED', 'flowUsage': 'NO_INFO'}}, 'medType': 'VIDEO', 'mirBwUl': '1000 Kbps',
 'resPrio': 'PRIO_3'}}, 'notifUri': 'localhost:8500/npcf-policyauthorization/v1/terminate', 'suppFeat': '0', 'ueIpv
4': '192.168.6.1'}, 'ascRespData': {'suppFeat': '0'}}
[2023-02-17 15:26:04 +0000] [5208] [INFO] 127.0.0.1:46432 POST /npcf-policyauthorization/v1/app-sessions 2 201 5 89
1950
```

- Screencap of the PCF logs



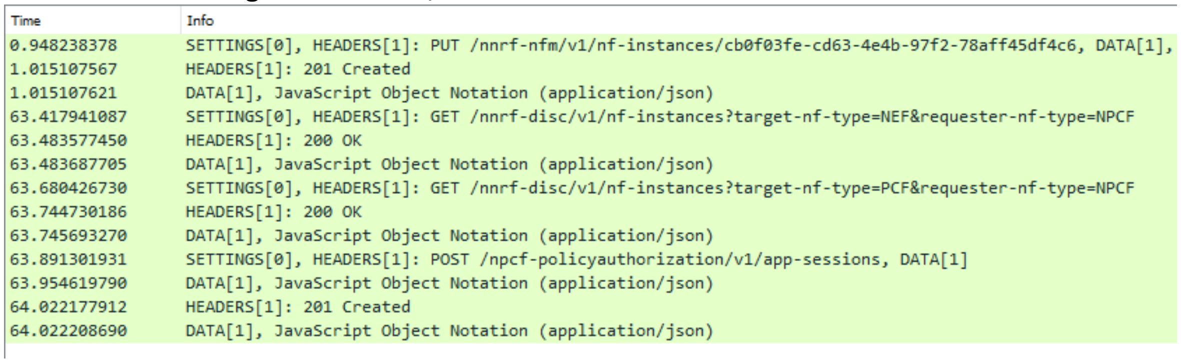
```
140100133500672/183731 16:26:07   INFO:Platform:tcp_accept():486> New TCP connection accepted!
140100133500672/183731 16:26:07   INFO:http2:ph_http2_server_connection_handler():719> New HTTP2 Client connection: [192.168.11.2
]:55696 @fd(17)
140100133500672/183731 16:26:07   INFO:Platform:phoenix_spawn_with_id_thread():146> spawned a thread with rank 26 and the thread
id 140099576637184
140099576637184/183731 16:26:07   WARN:Platform:phoenix_mem_prealloc_realloc():611> http2_req_mem_pool: can't realloc 0x7f6baa3f0
a28 because it is not the last allocation from this pool
140099576637184/183731 16:26:07   INFO:pcf:ApplicationSessionsCollectionAPI_postAppSessions_req_handler():1559> PCF received http
 POST request for create Application Sessions from AF/NEF.
140099576637184/183731 16:26:07   INFO:pcf:pcf_session_binding():93> sesseion binding by ipv4, received ipv4 address from app-ses
sion request is: 192.168.6.1
140099576637184/183731 16:26:07   INFO:pcf:pcf_ht_find_individual_sm_policy():340> found_indi_smpolicy smpolicyId: c16b5e64-fde9-
4308-9189-13c8fefb747f
140099576637184/183731 16:26:07   INFO:pcf:pcf_ht_new_individual_app_session_context():212> finish Creating a new individual app
session context with appSessionId: c84c417a-cbac-4174-96bc-3373ef805598
140099576637184/183731 16:26:07   INFO:pcf:pcf_indi_app_session_context_create():550> Traffic Influence data route_to_location is
 empty
140099576637184/183731 16:26:07   INFO:pcf:pcf_event_subscription_create():158> Start event subscription process..
140099576637184/183731 16:26:07   INFO:pcf:pcf_af_procedure_operation():989> AF Traffic data is empty
140099576637184/183731 16:26:07   INFO:pcf:pcf_af_procedure_operation():995> PCF received Media Components data from AF
140099576637184/183731 16:26:07   INFO:pcf:pcf_smpolicy_update_notify_request():837> notification_uri is : http://192.168.11.40:8
080/npcf-smpolicycontrol/v1
140099576637184/183731 16:26:07   INFO:pcf:pcf_smpolicy_update_notify_request():848> Successfully sent pcf update notification
```

- Screencap of the NRF logs



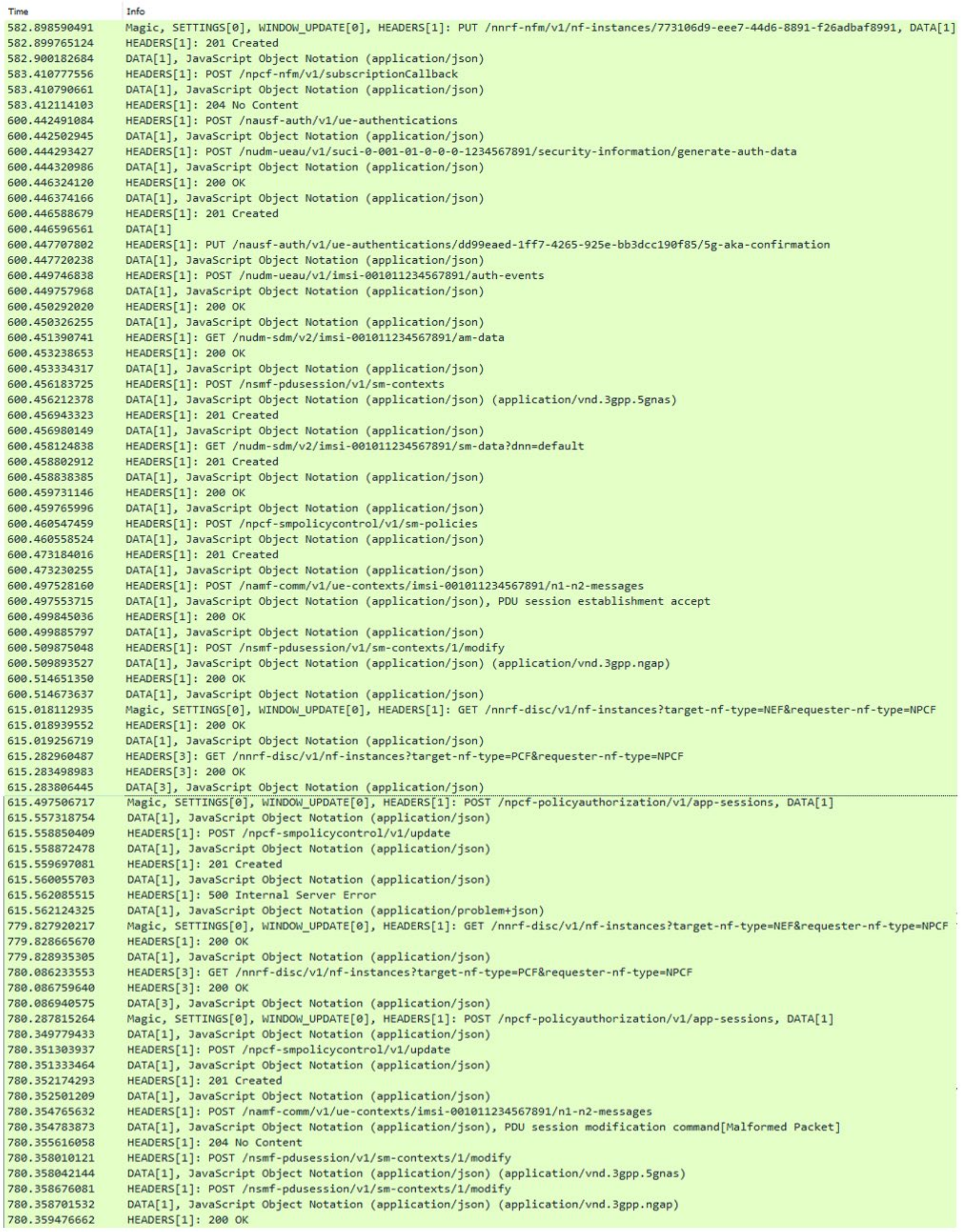
```
15:29:33   INFO:Platform:tcp_accept():486> New TCP connection accepted!
15:29:33   INFO:http2:ph_http2_server_connection_handler():719> New HTTP2 Client connection: [192.168.13.2]:40472 @fd(30)
15:29:33   INFO:Platform:phoenix_spawn_with_id_thread():146> spawned a thread with rank 26 and the thread id 139724438087424
15:29:33   INFO:nrf_server:NFInstanceID_register():138> Registering NFInstance NF: NEF, ID: cb0f03fe-cd63-4e4b-97f2-78aff45df4c6
15:29:33   INFO:nrf_server:nf_status_event_start():93> Sending Subscription Updates of NF: cb0f03fe-cd63-4e4b-97f2-78aff45df4c6
15:29:33   INFO:Platform:phoenix_spawn_with_id_thread():146> spawned a thread with rank 1 and the thread id 139724429694720
15:29:33 NOTICE:nrf_server:NFInstanceID_register():162> Registering NFInstance NF: NEF, ID: cb0f03fe-cd63-4e4b-97f2-78aff45df4c6
15:29:33   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 1 thread id [139724429694720]
15:30:19   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:30:19   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139724446480128]
15:30:19   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:30:19   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139725133063936]
15:30:19   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:30:19   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139724463265536]
15:30:19   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:30:19   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139725149849344]
15:30:19   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:30:19   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139725107885824]
15:30:19   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:30:19   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139725124671232]
15:30:19   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:30:19   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139724471658240]
15:30:33   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:30:33   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139724438087424]
15:30:36   INFO:Platform:tcp_accept():486> New TCP connection accepted!
15:30:36   INFO:http2:ph_http2_server_connection_handler():719> New HTTP2 Client connection: [192.168.13.2]:40498 @fd(14)
15:30:36   INFO:Platform:phoenix_spawn_with_id_thread():146> spawned a thread with rank 26 and the thread id 139724421302016
15:30:36   INFO:nrf_server:NFInstancesStoreAPI_search():52> Searching for NFInstance: NEF
15:30:36   INFO:nrf_server:NFInstancesStoreAPI_search():52> Searching for NFInstance: PCF
15:31:36   INFO:http2:ph_http2_server_client_conn_thread_cleanup():668> Cleaned up HTTP2 Client connection.
15:31:36   INFO:Platform:phoenix_init_exec_unit_thread():110> Exiting exec_unit_thread, rank is 26 thread id [139724421302016]
```

- Trace of messages in the NEF/AF



Time	Info
0.948238378	SETTINGS[0], HEADERS[1]: PUT /nnrf-nfm/v1/nf-instances/cb0f03fe-cd63-4e4b-97f2-78aff45df4c6, DATA[1],
1.015107567	HEADERS[1]: 201 Created
1.015107621	DATA[1], JavaScript Object Notation (application/json)
63.417941087	SETTINGS[0], HEADERS[1]: GET /nnrf-disc/v1/nf-instances?target-nf-type=NEF&requester-nf-type=NPCF
63.483577450	HEADERS[1]: 200 OK
63.483687705	DATA[1], JavaScript Object Notation (application/json)
63.680426730	SETTINGS[0], HEADERS[1]: GET /nnrf-disc/v1/nf-instances?target-nf-type=PCF&requester-nf-type=NPCF
63.744730186	HEADERS[1]: 200 OK
63.745693270	DATA[1], JavaScript Object Notation (application/json)
63.891301931	SETTINGS[0], HEADERS[1]: POST /npcf-policyauthorization/v1/app-sessions, DATA[1]
63.954619790	DATA[1], JavaScript Object Notation (application/json)
64.022177912	HEADERS[1]: 201 Created
64.022208690	DATA[1], JavaScript Object Notation (application/json)

- Trace of messages in the 5G Core



```
Time                Info
582.898590491       Magic, SETTINGS[0], WINDOW_UPDATE[0], HEADERS[1]: PUT /nnrf-nfm/v1/nf-instances/773106d9-eee7-44d6-8891-f26adbaf8991, DATA[1]
582.899765124       HEADERS[1]: 201 Created
582.900182684       DATA[1], JavaScript Object Notation (application/json)
583.410777556       HEADERS[1]: POST /npcf-nfm/v1/subscriptionCallback
583.410790661       DATA[1], JavaScript Object Notation (application/json)
583.412114103       HEADERS[1]: 204 No Content
600.442491084       HEADERS[1]: POST /nausf-auth/v1/ue-authentications
600.442502945       DATA[1], JavaScript Object Notation (application/json)
600.444293427       HEADERS[1]: POST /nudm-ueau/v1/suci-0-001-01-0-0-0-1234567891/security-information/generate-auth-data
600.444320986       DATA[1], JavaScript Object Notation (application/json)
600.446324120       HEADERS[1]: 200 OK
600.446374166       DATA[1], JavaScript Object Notation (application/json)
600.446588679       HEADERS[1]: 201 Created
600.446596561       DATA[1]
600.447707802       HEADERS[1]: PUT /nausf-auth/v1/ue-authentications/dd99eaed-1ff7-4265-925e-bb3dcc190f85/5g-aka-confirmation
600.447720238       DATA[1], JavaScript Object Notation (application/json)
600.449746838       HEADERS[1]: POST /nudm-ueau/v1/imsi-001011234567891/auth-events
600.449757968       DATA[1], JavaScript Object Notation (application/json)
600.450292020       HEADERS[1]: 200 OK
600.450326255       DATA[1], JavaScript Object Notation (application/json)
600.451390741       HEADERS[1]: GET /nudm-sdm/v2/imsi-001011234567891/am-data
600.453238653       HEADERS[1]: 200 OK
600.453334317       DATA[1], JavaScript Object Notation (application/json)
600.456183725       HEADERS[1]: POST /nsmf-pdusession/v1/sm-contexts
600.456212378       DATA[1], JavaScript Object Notation (application/json) (application/vnd.3gpp.5gnas)
600.456943323       HEADERS[1]: 201 Created
600.456980149       DATA[1], JavaScript Object Notation (application/json)
600.458124838       HEADERS[1]: GET /nudm-sdm/v2/imsi-001011234567891/sm-data?dnn=default
600.458802912       HEADERS[1]: 201 Created
600.458838385       DATA[1], JavaScript Object Notation (application/json)
600.459731146       HEADERS[1]: 200 OK
600.459765996       DATA[1], JavaScript Object Notation (application/json)
600.460547459       HEADERS[1]: POST /npcf-smpolicycontrol/v1/sm-policies
600.460558524       DATA[1], JavaScript Object Notation (application/json)
600.473184016       HEADERS[1]: 201 Created
600.473230255       DATA[1], JavaScript Object Notation (application/json)
600.497528160       HEADERS[1]: POST /namf-comm/v1/ue-contexts/imsi-001011234567891/n1-n2-messages
600.497553715       DATA[1], JavaScript Object Notation (application/json), PDU session establishment accept
600.499845036       HEADERS[1]: 200 OK
600.499885797       DATA[1], JavaScript Object Notation (application/json)
600.509875048       HEADERS[1]: POST /nsmf-pdusession/v1/sm-contexts/1/modify
600.509893527       DATA[1], JavaScript Object Notation (application/json) (application/vnd.3gpp.ngap)
600.514651350       HEADERS[1]: 200 OK
600.514673637       DATA[1], JavaScript Object Notation (application/json)
615.018112935       Magic, SETTINGS[0], WINDOW_UPDATE[0], HEADERS[1]: GET /nnrf-disc/v1/nf-instances?target-nf-type=NEF&requester-nf-type=NPCF
615.018939552       HEADERS[1]: 200 OK
615.019256719       DATA[1], JavaScript Object Notation (application/json)
615.282960487       HEADERS[3]: GET /nnrf-disc/v1/nf-instances?target-nf-type=PCF&requester-nf-type=NPCF
615.283498983       HEADERS[3]: 200 OK
615.283806445       DATA[3], JavaScript Object Notation (application/json)
615.497506717       Magic, SETTINGS[0], WINDOW_UPDATE[0], HEADERS[1]: POST /npcf-policyauthorization/v1/app-sessions, DATA[1]
615.557318754       DATA[1], JavaScript Object Notation (application/json)
615.558850409       HEADERS[1]: POST /npcf-smpolicycontrol/v1/update
615.558872478       DATA[1], JavaScript Object Notation (application/json)
615.559697081       HEADERS[1]: 201 Created
615.560055703       DATA[1], JavaScript Object Notation (application/json)
615.562085515       HEADERS[1]: 500 Internal Server Error
615.562124325       DATA[1], JavaScript Object Notation (application/problem+json)
779.827920217       Magic, SETTINGS[0], WINDOW_UPDATE[0], HEADERS[1]: GET /nnrf-disc/v1/nf-instances?target-nf-type=NEF&requester-nf-type=NPCF
779.828665670       HEADERS[1]: 200 OK
779.828935305       DATA[1], JavaScript Object Notation (application/json)
780.086233553       HEADERS[3]: GET /nnrf-disc/v1/nf-instances?target-nf-type=PCF&requester-nf-type=NPCF
780.086759640       HEADERS[3]: 200 OK
780.086940575       DATA[3], JavaScript Object Notation (application/json)
780.287815264       Magic, SETTINGS[0], WINDOW_UPDATE[0], HEADERS[1]: POST /npcf-policyauthorization/v1/app-sessions, DATA[1]
780.349779433       DATA[1], JavaScript Object Notation (application/json)
780.351303937       HEADERS[1]: POST /npcf-smpolicycontrol/v1/update
780.351333464       DATA[1], JavaScript Object Notation (application/json)
780.352174293       HEADERS[1]: 201 Created
780.352501209       DATA[1], JavaScript Object Notation (application/json)
780.354765632       HEADERS[1]: POST /namf-comm/v1/ue-contexts/imsi-001011234567891/n1-n2-messages
780.354783873       DATA[1], JavaScript Object Notation (application/json), PDU session modification command[Malformed Packet]
780.355616058       HEADERS[1]: 204 No Content
780.358010121       HEADERS[1]: POST /nsmf-pdusession/v1/sm-contexts/1/modify
780.358042144       DATA[1], JavaScript Object Notation (application/json) (application/vnd.3gpp.5gnas)
780.358676081       HEADERS[1]: POST /nsmf-pdusession/v1/sm-contexts/1/modify
780.358701532       DATA[1], JavaScript Object Notation (application/json) (application/vnd.3gpp.ngap)
780.359476662       HEADERS[1]: 200 OK
```

9. Annex B

The following tables describe the HTTP messaging exchange produced by Ubitech VAO in several test cases.

Test 1: Create Component Descriptor

Endpoint

http:/10.6.226.239:8080/api/v1/component (POST)

Request Body:

{"name":"MyComponent","scaling":"NONE","architecture":"X86","lfInterfaceType":"","publicComponent":false,"dockerRegistry":"ubitech-public-repository.ubitech.eu","dockerImage":"mysql:5.5","dockerCredentialsUsing":true,"dockerCustomRegistry":true,"dockerUsername":"maestro","dockerPassword":"!maestro!$","requirement":{"vCPUs":2,"ram":2048,"storage":20,"hypervisorType":"KVM","gpuRequired":false},"environmentalVariables":[{"key":"MYSQL_DATABASE","value":"example"},{"key":"MYSQL_ROOT_PASSWORD","value":"fudge"}],"exposedInterfaces":[{"name":"mysqlpublicII","port":"33096","interfaceType":"CORE","transmissionProtocol":"BOTH"}],"requiredInterfaces":[],"healthCheck":{"name":"","httpURL":"","args":"mysqladmin -uroot --pfudge status","interval":10},"plugins":[{"pluginID":26}],"volumes":[],"devices":[],"labels":[{"labelID":2,"name":"SQL Database"},{"labelID":8,"name":"Custom Component"}],"hostname":"","networkModeHost":false,"privilege":false,"user":null,"oganization":null}

Response

201 Created

Response Body

Test 2: Fetch Service Graph

Endpoint

http://10.6.226.239:8080/api/v1/application/267 (GET id==267)

Request Body:

Response

200 OK

Response Body

{"code":"23","message":"Application has been fetched successfully","returnobject":{"id":267,"hexID":"AJB1JayvQm","name":"ABC","publicApplication":false,"componentNodes":[{"componentNodeID":915,"hexID":"W1cunD9PJF","name":"MyComponent1301","component":{"id":130,"name":"MyComponent","hexID":"CdBKGtg08u","publicComponent":false,"dateCreated":null,"lastModified":null,"scaling":null,"exposedInterfaces":[{"interfaceID":109,"name":"mysqlpublicII","port":"33096","interfaceType":"CORE","transmissionProtocol":"BOTH","componentID":null,"dateCreated":null,"lastModified":null}],"allowEdit":null,"allowDelete":null,"organization":"Admin_Organization"}}],"graphLinkNodes":null,"organization":"Admin_Organization"}}