AGH University of Science and Technology

Faculty of Computer Science, Electronics and Telecommunications

Engineering Thesis

Extendable deployment of 5G core network in cloud environment

Author: *Mateusz Zięba*
Degree programme: *Information And Communication Technologies*
Supervisor: *Piotr Boryło, PhD. Eng.*

Cracow, 2023

For Wiktoria, who kept me sane.

Contents

1. Introduction

In recent years, the deployment of 5th generation (5G) mobile networks has become increasingly important as demand for high-speed and low-latency connectivity continues to grow. One promising approach to improving the scalability and flexibility of 5G networks is the use of cloud-based infrastructure. This allows for the deployment of 5G core networks in a virtualized, highly-scalable environment. Thus, operators are able to meet the demands of their users easier and in more cost-effective manner.

As the 5th generation mobile networks deployments are on the increase, gradually covering further square kilometers of urbanized areas each year, mobile equipment vendors flood the market with 5G-enabled devices. The collaboration between network service providers and device manufacturers has led to the creation of new carrier services and an increase in the number of 5G subscribers. On the other hand, cloud services have never been more popular. The both small companies and global enterprises adopt cloud solutions, which are tailored to customer demands. Global trends are driving the development of cloud-based applications, hosting data, and even deploying complete networks as the new standard. Mobile network vendors and operators could not ignore all the benefits stemming from employing cloud solutions in mobile telecommunications. As a result, 5th generation mobile networks and the cloud have become inherently tied concepts.

1.1. Motivation and Scope

The incentive behind the thesis was exploring the relation between 5G mobile networks and the cloud. Furthermore, a delve into the topic of 5G core networks was to gain an insight into the emerging mobile networks. Thus, the work focuses on two aspects related to 5G networks. The first is the deployment of a 5G core network in an infrastructure that can be easily transferred to cloud environment. The second aspect concerns the current state of open-source software implementations of 5G core networks.

1.2. Structure of the thesis

The thesis is structured as follows. Chapter 2 provides an overview of 5th generation networks, virtualization and the architecture of 5G core network. Chapter 3 presents selected projects that implement 5G core networks. Chapter 4 presents selected procedures for 5G systems that will be tested on a deployed 5G network. Chapter 5 describes infrastructure used for the deployment of a 5G core network. Chapter 6 verifies the deployed 5G core network. Conclusively, Chapter 7 concerns extending a selected 5G core network implementation with an additional module. Chapter 8 concludes the thesis.

2. 5th generation mobile networks overview

This section briefly presents various technical aspects mentioned in the thesis. Section 2.1 introduces 5G mobile networks, while Section 2.2 describes virtualization in the context of mobile network. Finally, Section 2.3 presents the characteristics of the core part of 5G.

2.1. 5G cellular services

The evolution of mobile networks (MN) has been mainly driven by growing demand for new services and improvement of the present ones. 5th generation networks should meet requirements for three primary use-case scenarios: Enhanced Mobile Broadband (eMBB), Massive Machine Type Communication (mMTC), and Ultra-reliable and Low Latency Communications (URLLC) [16, 3]. Each scenario, presented in Figure 1, puts a different priority on network performance indicators. For instance, URLLC emphasizes the latency and mobility capabilities of 5G networks, whereas mMTC focuses on the battery life of user equipment and network capacity in terms of device density.

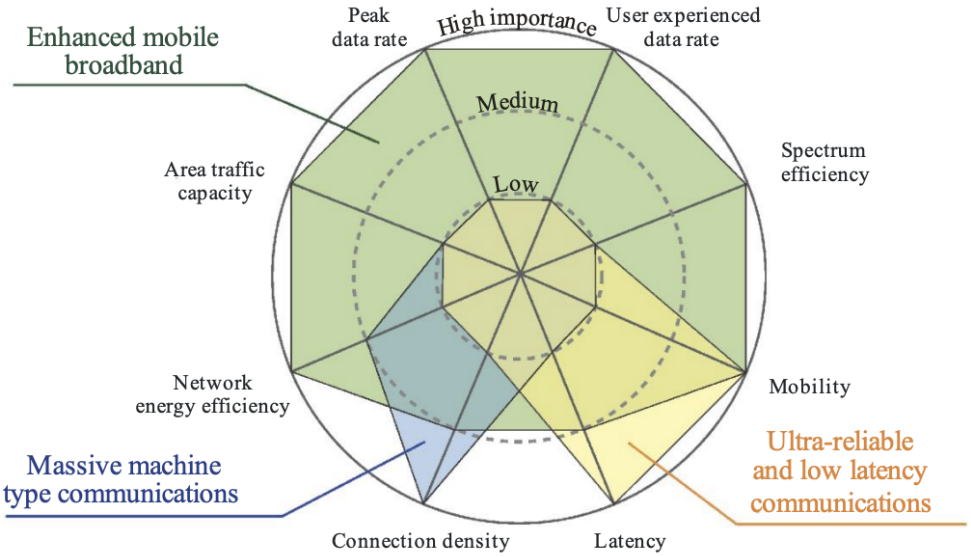

Figure 1. Service group requirements to be met by 5th generation networks[1].

The eMMB concerns the user-centric services. Mobile broadband aspect of mobile network caters for easier and non-interrupted access to multimedia content. The 5G networks should handle high data-rate services, whose requirements for bandwidth exceed the capacity of existing mobile networks. The multimedia applications such as Ultra High Definition, (UHD)/3D video streaming, Augmented Reality are to benefit from transition to 5G networks as they seek not only improved downlink (DL) throughput, but also reduced latency. With improved user-experience in mind, eMBB should facilitate greater area coverage [3, 16]. The eMBB also entails increased spectral efficiency to facilitate the requirements for greater bandwidth. Furthermore, it attempts to constrain energy consumption of access network (AN) [16].

As IoT solutions gain momentum and yet novel use-cases are invented, thus, mobile networks face a challenge to match the market expectations. Millions of appliances functioning without a direct human intervention require stable communication measure. The 5G networks are designed to expose services to extraordinarily high number of devices via a group of services under the name mMTC [16].

These services indicate that access to the network is possible in an area densely populated with IoT devices (usually sensors) [16, 1]. The mMTC services are presumed not only to facilitate access, but also to allow for bidirectional communication, as sensors are occasionally expected to send uplink data to a managing application or cloud. To reduce the service unavailability, mMTC facilitates the 5G access network to reach devices located in highly inaccessible places [1], e.g. located in underground parking lots.

The mMTC services also emphasize the energy management, which is crucial in numerous IoT scenarios, e.g. pressure sensors embedded in roads. Replacement of such is impossible without physically extracting the sensor and therefore interfering with the surrounding environment.

The final group of services introduced in 5G systems concerns ultra-high availability, low user-plane latency and mobility. The URLLC group names all applications critical to public safety that simultaneously require significant reduction of delay and packet loss imposed by the network. Exemplary use-cases applicable for URRLC services include remote medical surgery, transportation safety or control of production processes.

In order to provision novel services, 5G networks underwent a complete redesign process, including both the access network (5G-AN) and Core Network (5GCN). To simplify the transition towards 5G networks for the mobile carriers, 3GPP defined two ways of 5G network deployments. Since many mobile network operators incurred significant costs in replacing their 3G networks with LTE technology, the expense of deploying a 5G core network with a dedicated access network may not be well received. Therefore, 5G networks can be realized on top of existing LTE infrastructure. Non-Standalone (NSA) architecture allows mobile network operators to use the 5G access network (5G-AN) with an existing LTE core network (Enhanced Packet Core or EPC). Standalone (SA) architecture fully uses 5G technology, including both the 5G access network and the 5G core network.

5G networks, as per 3GPP Release 15 [28], embrace abstraction of network functions (NFs) from the hardware via virtualization.

2.2. Virtualization in 5G core networks

As the evolution in mobile networks progressed, one aspect remained intact: the reliance on 'black-box' solutions in terms of mobile network components. The vendors provided integrated hardware with dedicated software that would properly interact only with other proprietary equipment. This requirement has forced mobile network operators to use network solutions from a single producer, even if other options might be more suitable for their needs.

To overcome the limitations of proprietary equipment, 5G networks utilize virtualization of their core components and use commercial off-the-shelf (COTS) hardware. Virtualization allows for additional flexibility, as NFs can be dynamically scaled in or out according to the network state. Additionally, it

reduces time-to-market (TTM) [24] of new services, since the adaptation time of network architecture is reduced. In principle, virtualization significantly impacts both operational (OPEX) and capital (CAPEX) expenses. CAPEX of deploying 5G core networks is greatly reduced as NFs no longer require to run on dedicated, costly hardware. OPEX of maintaining such aggregation of virtualized resources and adjusting them to provide novel services is also reduced.

5G networks have been designed to support deployment that uses the concept of Network Function Virtualization (NFV) [30]. This virtualization framework describes the network node functions as a software-entities that can run independently of the hardware. There are three main domains of NFV framework:

- Virtual Network Function (*VNF*), which is a software implementation of a network component, capable of running on various platforms.
- Network Function Virtualization Infrastructure (*NFVI*) denotes the COTS hardware and overall physical architecture that VNFs are deployed on.
- NFV management and orchestration (*NFV-MANO*) architectural framework, which caters for administrating the physical and virtual infrastructures [15, 31].

There are two ways of deploying NFs. The first one involves containerization techniques, whereas the second one employs virtual machines.

2.2.1. Containerized Network Function (CNF)

Mobile core networks have transitioned from monolithic architecture into a mesh of limited-functionality interconnected components. These components (NFs) might be deployed using containerized network services [15]. A container is a lightweight, standalone unit of software, encapsulated with the runtime environment and all required dependencies [38, 32]. Containerized NF applications may run as isolated processes in user host operating system (OS).

Additionally, a containerized NF is infrastructure-agnostic [38], since it is deployed on top of a host OS. This intermediary layer allows the NF to ignore the physical aspects of a host platform, as it interacts only with the OS. Furthermore, a containerized network function requires less disk space of commodity equipment, as container does not necessarily include an OS image of its own. Instead, a containerized NF shares the host kernel with other NFs [8, 38].

By containerizing NFs, mobile operators can capitalize on their physical network resources more effectively. This is because a server with limited disk space can host more NFs of this deployment type [25, 15, 4].

Implementing the core network components as containerized functions in clustered worker nodes enables a straightforward vertical scaling, increased reliability and swift fault recovery [4]. The paper [21] highlights a cloud-native deployment of 5GCN as a Kubernetes-based cluster. The network functions are deployed in worker nodes that are based on dedicated hardware platforms. Authors of [26] implement a public-cloud based 5GCN using containerized images of NFs, managed by Kubernetes.

2.2.2. Virtual machine based VNF (VM-VNF)

Another way to virtualize network functions is to use virtual machines. In this deployment type, a NF runs as a process on the guest operating system inside a VM. A VM-based NF requires a commodity hardware server and an underlying operating system with various dependencies. Like other NFs in a 5GCN, it may have redundant OS features that are not needed for its operation [38].

To ensure proper isolation of network functions hosted in virtual machines, the VNF VMs are not allowed to access the resources of the deployment platform. Instead, a hypervisor [15, 13], also known as a virtual machine monitor (VMM), is responsible for scheduling CPU, memory access, and networking for the guest VMs [8]. The hypervisor dynamically assigns resources to the monitored VMs as needed. The VMs are unaware of the resource management, as it is designed to mimic actual hardware operations. There are two types of hypervisors.

Type-1, also known as a *bare-metal* hypervisor, is installed directly on the hardware without an operating system in between (Figure 2). This type of hypervisor has direct access to the hardware resources and can run multiple VMs on top of it.

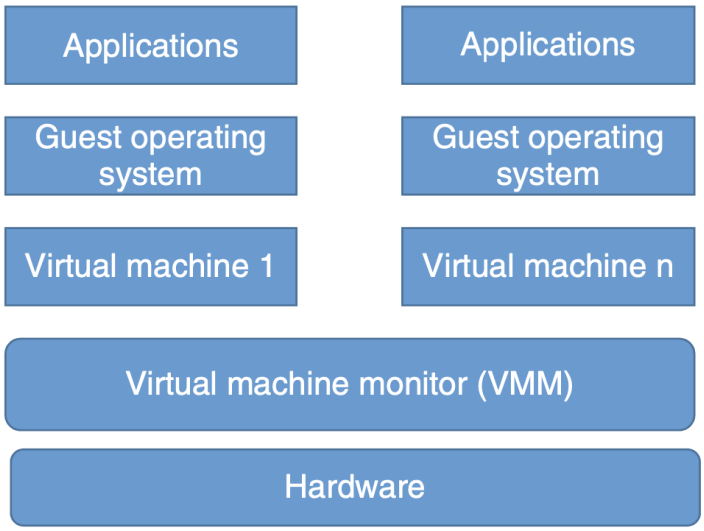

Figure 2. Bare-metal hosted VMs.

Source: Network Function Virtualization Concepts and Applicability in 5G Networks, Wiley [8]

Type-2 hypervisors, known as *hosted* hypervisors, are installed on top of an operating system and run within it (Figure 3). They do not have direct access to the hardware resources and rely on the host OS to provide it for the VMs running on top of them.

Omitting the operating system as an intermediate layer between the hypervisor and VMs mitigates the security risk originating from it [39]. Type-1 hypervisors allow the running software to become independent of the hardware, drivers or devices, as bare-metal hypervisors handle the translation of requests between physical and virtual layer [37].

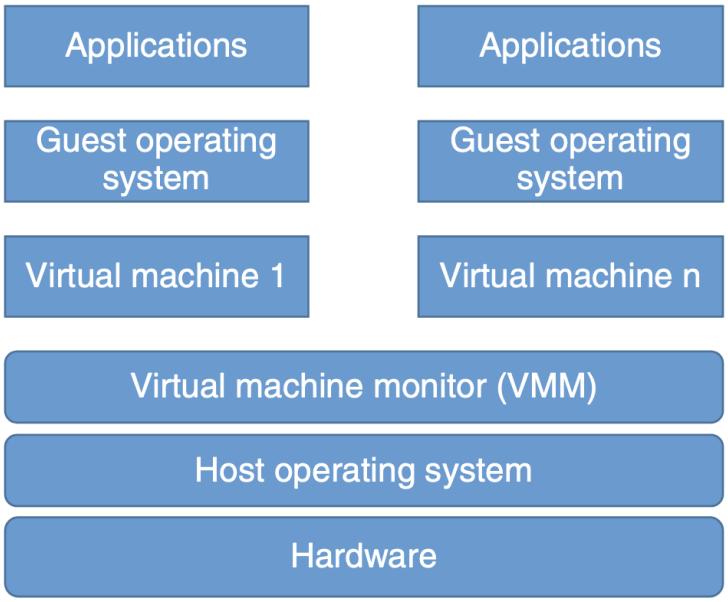

Figure 3. An overview of type-2 hosted VMs.

Source: Network Function Virtualization Concepts and Applicability in 5G Networks, Wiley [8]

Type-2 (hosted) hypervisors, have an additional layer created by the host operating system, which can limit their performance when translating requests. They also have greater latency when allocating and scheduling resources among VMs compared to type-1 (bare-metal) hypervisors [39]. Hosted hypervisors are commonly used for software testing as they are less expensive than bare-metal hypervisors. As a result, they are popular among end users who can host them on top of the OS of a commodity device.

Unfortunately, conventional VNF implementations induce substantial overhead in comparison with the lightweight containerization [26]. In general, VM-VNF requires more disk space as it includes an operating system necessary to host the NF process. The size of the NF implemented as a VM-VNF affects the time required for deployment and migration. Additionally, hardware virtualization consumes a significant portion of the physical resources, such as the CPU.

Despite the disadvantages of the virtual machine approach, VNFs have been successfully deployed on top of VMs. As the adoption of network function virtualization concepts in mobile networks has increased, containerized network functions have been replacing VM-based VNFs.

2.3. 5G core architecture

In order to fulfill the promise of IMT-2020 [1], the core part of 5G (as presented in Release 15 [6]) had to be redefined. The 5G cellular network was divided into two functional parts: the control plane (CP) and the user plane (UP). The CP is responsible for signalling traffic, while the UP is responsible for user data.

2.3.1. Service-based architecture

To achieve granularity and scalability, 5GCN networks use a new approach for interactions between core components. Service-based architecture (SBA) defines a model where a set of network functions provide services to other eligible NFs. This can be represented as a reference point view (Figure 4). A reference point represents point-to-point links between NFs that exchange services [7].

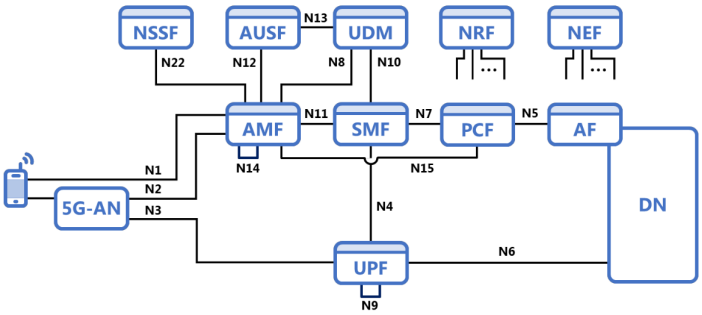

Figure 4. Reference point representation of interfaces between NFs.

Source: Sploty 6.0 5G Systems Overview [24]

The second way to represent SBA is through the service-based representation, where the network functions are interconnected by the services they provide (Figure 5). This representation is only applicable to the control plane. The interfaces between the CP and the user plane are implemented through reference points (e.g. the interface between the user plane function (UPF) and the session management function (SMF) [24]).

In practice, the SBA is mainly implemented through the service-based representation, as reference point interfaces are replaced with corresponding service-based interfaces (SBIs) wherever possible.

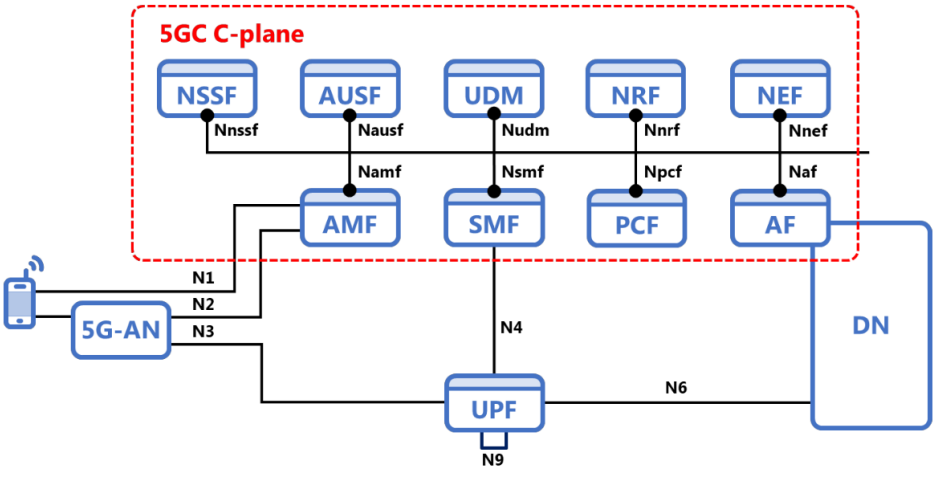

Figure 5. Service based representation of interfaces between NFs.

Source: Sploty 6.0 5G Systems Overview [24]

In SBA, 5GC NFs communicate via a common Service Framework, which defines two interaction models.

Depending on situation, a network function might request access to a resource that is provided by another function via *Request-Response* mechanism. The mechanism is applied when resource of a NF should be deleted, fetched or altered. Figure 6 depicts the communication flow of NF service consumption, where an eligible NF service Consumer issues request to resources of NF Service Producer.

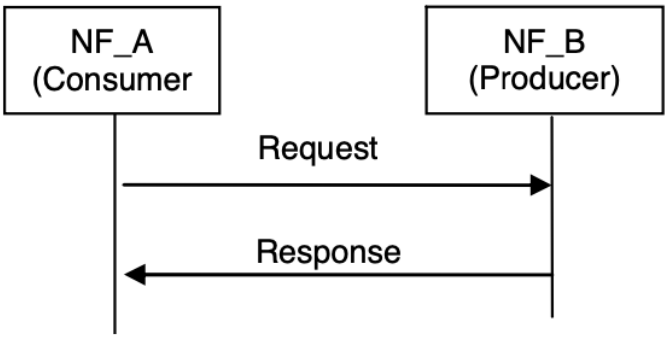

Figure 6. *Request-Response NF Service* example.

Source: 3GPP TS 23.501 version 16.6.0 Release 16 [30]

Alternatively, NF might request to be notified of an event that has occurred in the network, as depicted on Figure 7. Such interaction is defined by *Subscribe-Notify* model, where a NF Service Consumer subscribes to services offered by a NF Service Producer [7]. If a given event occurs in the network, the Service Produces is obliged to send a notification to all subscribed NFs.

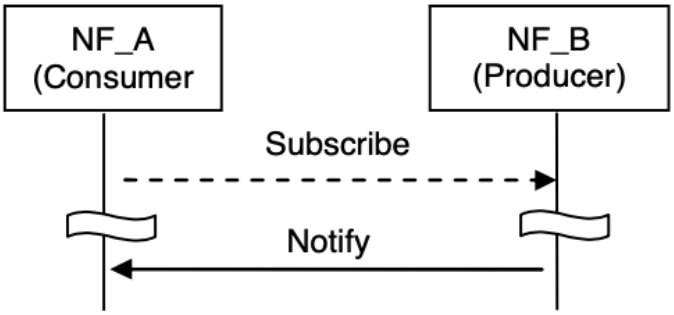

Figure 7. *Subscribe-Notify NF Service* example.

Source: 3GPP TS 23.501 version 16.6.0 Release 16 [30]

The collection of a Network Function's Services becomes accessible upon the NF registration to the network via Network Repository Function (NRF). Prior to executing a procedure that requires an external Service, the attempting NF first queries the NRF to discover NFs, which provide a given Service [7]. Service unavailability due to NF error is transparent to consumer NFs, and can only be discovered by a direct query to NRF [7]. If a NF does not refresh its registration in NRF, it is

considered broken and the corresponding event is propagated to all subscribed NFs. Such facilitates straightforward 5GC migration to cloud, as NFs can be deployed and discarded at will with no direct impact to remaining NFs [7, 28].

In practice, NFs Services are exposed via ReEpresentational State Transfer (REST) application programming interfaces (APIs). The architecture employs Hyper Text Transfer Protocol version 2 (HTTP/2) to facilitate communication between NFs. As per [28], such interface is denoted a Service Based Interface.

2.3.2. NFs overview

The functionality of 5GC is distributed over the NFs that build it. Each NF is assigned a specific service to cover. Upon a NF failure, only a subset of Services becomes unavailable until a NF is recovered. The complexity of 5G mobile networks had to be distributed over a number of NFs in such way that a NF serves a concrete purpose. For instance, a NF may only provide authorization of access to the network, but the access itself would be facilitated by a different NF. As a result, the 5GCN consists of a total of 26 NFs [30], each responsible for a subset of functionalities.

Below, selected NFs are described:

- **Access and Mobility Function** (*AMF*) is in charge of UE admission control, access authorization and authentication. Additionally, it provides support for network slicing via Network Slice-Specific Authentication and Authorization [24, 30].
- **Session Management Function** (*SMF*) is responsible for establishing and managing PDU sessions. It also manages the tunnel maintenance between UPF and the access network node. Additionally, the SMF handles UE IP allocation.
- **Unified Data Management** (*UDM*) function is a central repository for subscriber-related information. UDM performs access authorization based on subscription data. Additionally, it acts as an intermediary in subscription management; eligible NFs interface UDR to extract subscriber data [24].
- **Policy Control Function** (*PCF*) is a unified policy framework to govern network behavior; this NF provides policy rules that are based on UDR structured data resources to the remaining Control Plane NFs to expedite policy enforcement [24, 30].
- **Unified Data Repository** (*UDR*) is responsible for managing subscriber data. Multiple entities of UDR can be deployed in the network [24].
- **Authentication Server Function** (*AUSF*) handles authentication requests for 3GPP/non-3GPP access [24, 36].
- **Network Slice Selection Function** (*NSSF*) manages UE allowed/configured NSSAIs. It is responsible for anchoring UE in a proper AMF instance based on undertaken slice selection measures [30].

- **Network Repository Function** (*NRF*) is in charge of providing NF discovery to requesting functions. It holds information about functions registered in the network and services they provide.
- **Binding Support Function** (*BSF*) correlates the data sessions and policy-delivering PCF entities [29].
- **Network Data Analytic Function** (*NWDAF*) provides data analysis related to the 5G networks for service optimization.

2.3.3. Slicing

With the introduction of support for diversified services, 5G networks must handle various, and often impossible to be achieved simultaneously, requirements for capacity, isolation or resilience. Services such as mMTC and URLLC, as described in Section 2.2, cannot impact one another and should be handled transparently. 5G networks ensure that, for instance, a mesh of sensors does not interfere with a service belonging to a critical time and latency category, such as telesurgery [24]. The mechanism responsible for network isolation is called slicing.

A slice is a logical network with components customized to meet the needs of a specific service (Figure 8) [16]. It uses a dedicated core network architecture and provides access to services located in external data networks (DNs). A DN can provide operator services, internet access, or third-party services [16].

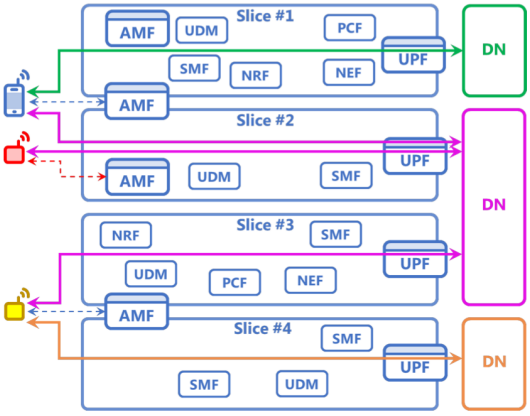

Figure 8. Sliced 5GCN with various components in each slice

.

Source: Sploty 6.0 5G Systems Overview [24]

Therefore, mobile service operators focus on deploying only the necessary functions, resources, or storage in a given slice. A slice can also include the 5G access network resources, such as radio resources or gNB processing units. This type of logical network is called an end-to-end (E2E) slice [16], as shown in Figure 9.

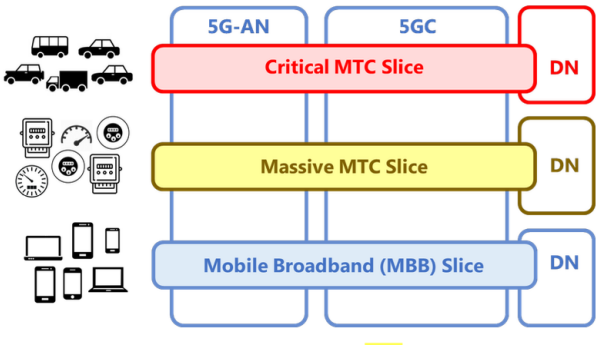

Figure 9. *E2E Slices for diverse service groups.*

Source: Sploty 6.0 5G Systems Overview [24]

Common slices in 5G networks are related to the basic service groups: eMMB, mMTC, URLLC (as mentioned in Section 2.1).

A network slice is uniquely identified by a Single Network Slice Selection Assistance Information (S-NSSAI). S-NSSAI consists of two values:

- Slice/Service Type (SST), which describes the expected behavior of a slice in terms of services.
- The slice differentiator (SD), which is used to distinguish slices belonging to the SST.

The SD, which is optional, is represented by a combination of 24 bits, while the mandatory SST is described using 8 bits. The differentiator allows Public Line Mobile Network (PLMN) operators to provide the same features, but to a different group of customers (i.e. groups of User Equipment (UE).

Certain values for Service Types must be consistent within all deployed 5G core networks. These values have been standardized by 3GPP and are reserved for basic service types in 5G networks (Table 1).

Table 1. Standardized Slices defined in 3GPP TS 23.501 version 16.6.0 Release 16 [30].

Slice/Service type	SST value
eMBB	1
URLLC	2
mMTC	3

S-NSSAIs are not required to have a dedicated subset of resources, as they might be served by the same Network Slice instance (NSI). Similarly, there might be numerous NSIs associated with a S-NSSAI. It is useful when a subset of resources in a 5G core network becomes inaccessible, i.e. a platform hosting NFs of a NSI fails. The other NSI serves as a fallback instance to provision the service [24, 30].

UE might be configured with multiple S-NSSAI. Such collection of S-NSSAIs is defined as Network Slice Selection Assistance Information (NSSAI). When UE requests access to the 5GN, it signals the NSSAI to the 5G base station in order to determine a correct AMF, Network Slice or NSI for a given service.

3. 5GCN implementations

This section concludes the research into the open source 5G core network implementations. Section 3.1 describes the aspects of a framework and scenarios in which they need to be considered. Further, Sections 3.2 and 3.3 highlight the characteristics of Open5GS and free5GC implementations, respectively.

3.1. Rationale determining 5GCN selection

A number of 5GCN implementations are accessible as open license programs [19]. However, a choice of the framework has to match the adopted goal. This might involve evaluating the number of Network Functions supported.

Procedures for 5G Systems, as defined in [28], involve various NFs. In order to conduct research of event exposure mechanisms in 5G networks, a Network Exposure Function (NEF) is necessary [24].

Although the NF by itself may seem sufficient, the list of its implemented services determines the feasibility of certain research. For example, if the study involves forwarding event notifications to an external subscribing NF, the NEF must be able to subscribe to a given service of a core network NF after receiving a request from an eligible party (such as an AF or third-party application). Additionally, the NF consuming the event exposure service of the NEF (such as an AF) must be able to send a query to trigger it. This consideration leads to the second factor in selecting a 5GCN: *mutual service support*.

Additionally, one might want to take into account the following aspects:

- Performance of a given implementation.
- The community around the project, which might indicate its likelihood to be abandoned.
- The number of open issues and bugs regarding code implementation.
- The quality of documentation.

Two 5G core network implementations are briefly described in the following subsections: Open5GS and Free5GC.

3.2. Open5GS

Open5GS is a framework developed in C language, initially covering LTE-A/LTE EPC network functionalities [23, 15]. It has evolved into a fully-fledged 5G NSA/SA service. Its core components can be used to deliver either EPC based 5G NSA core or 5G SA core [18]. The repository covers 3GPP Release 16 5G System Architecture requirements [30, 23].

Open5GS follows NFV principle, as the services deployed in a network might be disassociated from the hardware. A dual approach is possible:

- The NFs can be deployed as separate processes on VMs.
- The 5G core NFs can be deployed as containers, which is a common practice for implementing 5G core networks in public cloud environments[14, 17].

5GCN functions, by default, are meant to be run as programs in the Linux environment. As of now, the Open5GS has been tested to function properly on the following operating systems: Debian, Ubuntu, CentOS, openSUSE, Fedora, Mac OSX [15]. If the VNF approach is desired, the framework can be implemented on virtual machines hosted by a hypervisor of choice. Depending on the test scenario, the NFs may be spread across multiple VMs. If this is the case, then the routing between them must be ensured (i.e. the IP addresses of NFs SBI interfaces should be accessible).

Each NF shares and consumes services utilizing a dedicated SBI interface [30], which Open5GS by default exposes via loopback interface of the commodity server it is run on. If virtualizing the NFs involves distributing them over multiple VMs, the SBIs IP addresses should be accessible either through the hypervisor or an external router.

The NF of high significance in terms of service exposure via SBI is Unified Data Repository. The UDR implemented by Open5GS does not store Subscriber Data. Instead, it interfaces with MongoDB, a NoSQL document-oriented database that stores data in a Java Script Object Notation (JSON) structure [2]. The external database is accessible via IP address that is bound to a loopback interface. Therefore, if NF scattering is considered, it should be changed to other address in the MongoDB configuration file. Such change must be reflected in the UDR YAML file as well, since the database connection is executed when the NF process is called. The above is possible as the framework maintains a set of docker images of Network Functions.

The Open5GS implementation of 5GC requires an ordered execution of NFs. Prior to deployment of the remaining Network Functions, a NRF has to be up and running to enable NF and Service discovery.

Open5GS offers a thorough and well-documented understanding of the core network configuration process. The framework's NFs have already been used in a variety of research scenarios.

3.3. Free5GC

Free5GC is an open-source Golang implementation of 5th generation mobile core network that reflects Release 15 3GPP. It is maintained by National Chiao Tung University [22].

Unlike Open5GS, Free5GC does not provide 5G NSA functionality because it lacks the LTE network functions [3]. However, it includes an implementation of the NEF necessary to delegate event notification and facilitate events subscription to the remaining eligible functions [30, 28].

There is a difference between the number of Network Functions provided by Free5GC and Open5GS. Open5GS includes additional NFs (BSF and SCP) that are not present in Free5GC. This difference is partly due to the coverage of different 3GPP releases. SCP is defined in Release 16, so it is not implemented in Free5GC. BSF is defined in Release 15, which Free5GC should comply with, but it is not implemented in Free5GC.

4. Fundamental 5G procedures

This Section presents selected 3GPP procedures defined for 5G mobile networks. Subsection 4.1 presents the general admission process of the UE to 5G network. Subsection 4.2 shows the procedure of establishing a PDU session, which has been requested by UE. Finally, Subsection 4.3 gathers the procedures and presents the verification scenario of a deployed 5G core network.

4.1. General UE registration

In order to get access to the network services, UE must register with the 5G network. The procedure is performed upon an initial UE access to the 5G. Alternatively, it might be performed if UE was inactive for a certain period. Figure 10 shows the steps that are executed in the 5G network in order to facilitate UE access.

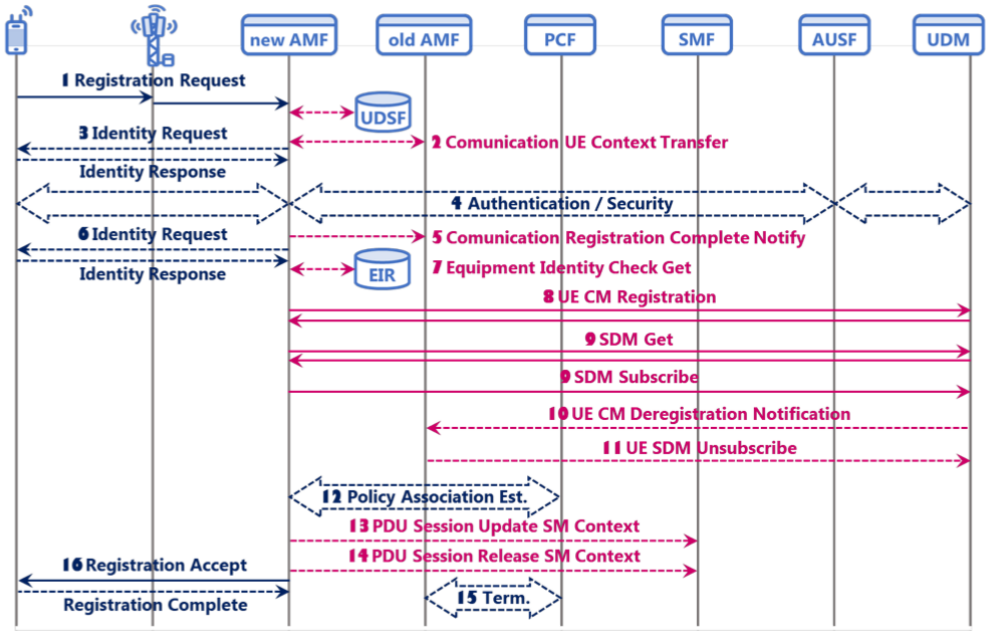

Figure 10. The general registration procedure for 5G Systems.

Source: Sploty 6.0 5G Systems Overview [24]

As UE requests access to a 5G network, the 5G access network performs a selection of an appropriate AMF. The selection is based on NSSAI and the PLMN signaled by the UE in non-access stratum (NAS) registration request. Upon a successful choice, the gNB forwards the UE access request to the selected AMF in *Initial UE Message*. The communication is performed over N2 interface. The AMF then initiates the authentication procedure of the UE. It selects an appropriate AUSF based on Subscription Concealed Identifier (SUCI), which might be signaled by the UE in NAS registration request.

The AUSF extracts the authentication data from UDM and performs the UE authenticate through NAS security messages. The authoriztion result is forwarded to the requesting AMF. The AMF then

selects an UDM and signals that it serves a given UE in the 5G network. Additionally, it subscribes to UDM notification of any deregistration procedures that may involve the UE. Next, AMF attempts to establish UE policy regarding the access network. In order to establish Access and Mobility (AM) Policy Association, AMF first queries the NRF to discover and select a PCF. The PCF creates a requested policy and notifies the AMF of any changes.

Finally, AMF checks for any PDU sessions on the NAS registration request. Based on the PDU session status, AMF decides to either reestablish or release a PDU session. It is achieved by querying a SMF associated with a PDU session. AMF might release a session by invoking *Nsmf PDU session release SM context* service of an associated SMF. Alternatively, AMF might signal a User Plane connection reactivation by invoking *Nsmf PDU session update SM context request*. AMF informs UE of the completion of a registration process through NAS registration accept message, to which UE is expected to respond to with NAS registration complete.

4.2. UE-triggered service request

Upon accessing the network, UE may also request access to the 5G services. As shown on Figure 11, UE signals the list of PDU Sessions in a NAS service request message. The request is forwarded by the 5G-AN to the AMF inside the Next Generation Application Protocol (NGAP) Initial UE Message. The NGAP facilitates the control plane signaling between 5G-AN node and the AMF.

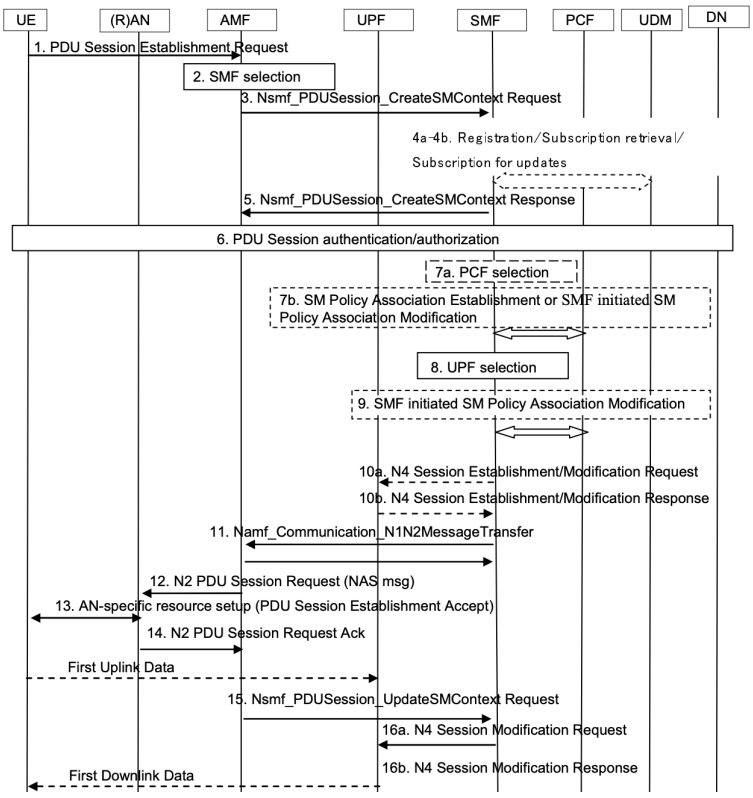

Figure 11. The UE Triggered Service Request procedure for 5G network.

Source: 3GPP TS 23.502 version 15.2.0 Release 15 [28]

The AMF extracts the S-NSSAI signaled by the UE in the service request and performs SMF selection. If it does not have any information regarding SMF, AMF performs NF discovery using NRF. The NRF choice is based on the S-NSSAI. If the AMF cannot find a NRF for a given slice, it queries the NSSF for such association. Upon achieving SMF information from the NRF, AMF requests PDU session establishment from an appropriate SMF by invoking *Nsmf_PDUSession_CreateSMContext* request. The SMF selection is based on S-NSSAI and DN requested by the UE.

If the called SMF is capable of establishing the requested PDU session, it informs the AMF by invoking *Nsmf_PDUSession_CreateSMContext* response and proceeds further.

The selected SMF performs PCF selection and instructs a PCF to create a policy for a given PDU session. The SMF then performs UPF selection based on the requested DN. As a result, an association over N4 link is established with a chosen UPF.

If both policy and N4 link establishments are successful, the SMF requests the AMF to inform the 5G-AN about the created PDU session parameters. This includes the User Plane tunnel for UE data (i.e. N3 tunnel between gNB and UPF selected for the session), QoS profiles and PDU session ID. The communication over the N3 tunnel is handled with GPRS Tunneling Protocol for user data (GTP-U). It is a standard protocol used for carrying user data between 5G-AN and the core network.

The serving gNB requests reconfiguration from the User Equipment, signaling that the PDU session has been successfully established.

Next, the 5G-AN informs the serving AMF of the access network tunnel created for the established PDU session (that is, the address of the N3 interface of the gNB for the data session) in N2 message. The AMF is then excepted to forward the message to the SMF, which requests reconfiguration of the UPF for the data session over N4 link. SMF initiates N4 session modification procedure, informing the UPF of gNB tunnel address for the established PDU session for the UE.

After UPF modification, both uplink and downlink data transfer is possible and the UE can use the PDU tunnel to access the requested service.

4.3. Slice-aware service establishment scenario

The procedures mentioned in Sections 4.1 and 4.3 can be combined, as the UE might request access to a data service during initial registration to the 5G network.

As shown in Section 2.3.3, UE might request access to a specific logical network, based on the S-NSSAI. Upon registration, UEs that are configured with different S-NSSAIs should access the corresponding network slice. Additionally, the requested PDU session should be established in the slice indicated by the UE. Therefore, the joint procedures of initial registration and service request could be supplemented with slicing.

Initial registration with PDU session establishment are key operations to be supported by 5G network. The joint procedures can be used to determine that a deployed 5G network operates properly. Therefore, the following verification scenario has been proposed. It is based on one of the examples of Open5GS deployments [35].

A deployed 5G Core Network is logically divided into two slices, each dedicated to provisioning a data service. The S-NSSAI describing the slices are presented in Table 4.3.

Table 2. Network slices in the implemented 5GCN.

SST value	SD value	DN name	UE identity
1	0x000001	internet	UE1
2	0x000002	ims	UE2

As presented in Figure 12, the slices are assigned with a dedicated SMF in the CP, and a UPF catering for the data traversing. The slices can be distinguished by referring to the respective colour (*pink* or *green*) of the NF blocks and the S-NSSAI mentioned within the NFs. The remaining NFs of the 5G core control plane are shared among the two slices, these are: NRF, UDR, UDM, BSF, PCF, AUSF, NSSF and AMF.

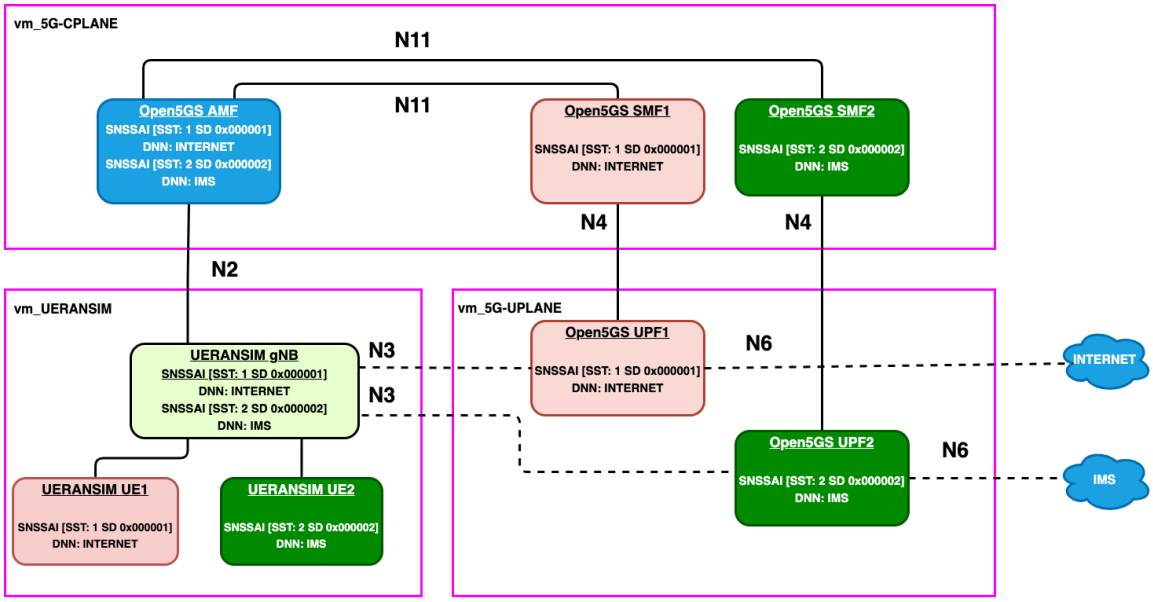

Figure 12. Virtualized 5G core network with slice-dedicated NFs.

The scenario to be realized using the mentioned 5G network architecture (Figure 12) is as follows. Two UEs (*UE1, UE2*) send a registration request via Non Access Stratum (NAS) message to a shared AMF. As the UEs are authorized, the separation related to slicing happens. The AMF performs SMF selection procedure and finds a SMF suitable for the particular S-NSSAI and data network. If the selection is positive, AMF requests PDU session creation from the SMF.

The selected SMF is supposed to associate the session with an appropriate UPF, based on the requested DN. As a result, the PDU session for a given UE should be created in a respective logical network, as shown on Figure 12. That is, UE1 PDU session request to *internet* DN should be established and managed by SMF1, and the User Plane tunnel for its data should be anchored in UPF1. The exact procedure should be repeated for the UE2 PDU session establishment request.

The 5G core network should allow for:

- Requesting a PDU Session establishment request issued by the common AMF to an adequate SMF instance, based on configured S-NSSAI.
- Creating a policy for the PDU session by the PCF.
- Establishing a N4 link between UPF and SMF for a given PDU session.
- Issuing NGAP (i.e. the message over the N2 link between AMF and gNB) message to request establishing User Plane for the PDU session between 5G-AN (gNB) and UPF.
- Verifying that the User Plane creation has been successful, i.e. whether N3 link (Figure 12) has been established.
- Tunneling UE-generated traffic to the outbound DN (i.e. public internet) through UPF.

5. Proposed environment

This section presents the practical deployment of a 5G mobile network core in an environment similar to the cloud. Section 5.1 describes the architecture of a virtualized 5GCN, while Section 5.2 presents the parameters of the VMs used to host it. Section 5.3 describes the configuration of the 5GCN NFs required for scenario realization mentioned in Section 4.3. Finally, Section 5.4 describes the 5G-AN configuration.

5.1. Architecture overview

In this thesis, the 5GCN network functions are deployed as separate processes on virtual machines (Section 2.2.2), as the optimization of resource allocation and efficient scaling is not a main issue.

NFs of a SA 5th generation mobile core network are hosted on virtualized nodes with the use of type-2 hypervisor (Section 2) [13]. From numerous type-2 hypervisors available, VirtualBox (VBox) is chosen for orchestrating the guest VMs that are designated as the network testbed. VBox is an open source virtual machine manager maintained by Oracle.

Essentially, the platform for hosting 5G core and access networks consists of three VMs. Each virtual machine is equipped with a northbound interface used for remote access by the host. The actual communication related to 5GCN is handled via data interfaces and the traffic is routed via an internal VirtualBox router. The separation is purely illustrative, as both types of interfaces employ the same networking type: host-only network [27]. It emulates a physical Ethernet switch, allowing for bidirectional communication between VMs, but dropping all traffic routed to the internet [27].

Open5GS is chosen as the 5th generation mobile core network framework suitable for the proposed architecture. The selection has been motivated by the broad and precise documentation, the number of services provided by NFs and the vast community around the project.

Since an emulated access network, together with UE, are required for a proper verification of 5GCN, an additional virtualized node is deployed for 5G access network. UERANSIM is an open

source framework delivering 5G-AN functionalities [33]. It provides a software implementation of gNodeB (gNB), which is a base station for 5th generation AN, and a 5G-enabled UE.

In the deployed 5G network, the Service Based Architecture (SBA) communication via SBI interfaces and N2/N1/N4 link message exchange is routed over private address space. This approach requires little effort to be set up with public cloud network service, e.g., Azure virtual network [20] or Amazon VPC [17].

The architecture has been carefully designed to ensure repeatability and facilitate cloud migration. Thus, the 5GCN and access network can be fully deployed in a public cloud.

5.2. Setup parameters

The testbed for 5G network (5GCN + 5G-AN) deployment via VirtualBox is based on a personal laptop. Its physical parameters are as listed in Table 3).

Table 3. VBox host platform: MacBook Pro 13-inch (2019) specification.

CPU	1,4 GHz Quad-Core Intel Core i5 Turbo Boost up to 3.9GHz
GPU	Intel Iris Plus Graphics 645 1536 MB
Disk Size	128GB SSD
RAM	8 GB 2133 MHz LPDDR3
Operating System	macOS Monterey

The VirtualBox virtual machines are assigned with the following roles:

- **vm_5GC-CP** hosts the control plane NFs.
- **vm_5GC-UPF** hosts 5GC User Plane Functions.
- **vm_5G-UERANSIM** hosts the 5G access network together with UEs.

To accommodate more NFs deployed on **vm_5GC-CP** compared to the other VMs, its physical resources (i.e., RAM) have been doubled. Each VM has been equipped with the 64-bit version of *Debian 11 Bullseye* OS. The configuration of the physical parameters of the deployed VMs is presented in the Table 4.

Table 4. VBox guest platforms parameters.

VM name	RAM	disk space	OS
vm_5GC-CP	2048 MB	6 GB	Debian 11 64 bit
vm_5GC-UPF	1024 MB	6 GB	Debian 11 64 bit
vm_5G-UERANSIM	1024 MB	6 GB	Debian 11 64 bit

When a VM generates any outbound traffic, the NAT adapter of Oracle VM VirtualBox's NAT engine extracts the TCP/IP data and forwards it through the host operating system to a remote network (e.g., the internet). The data is therefore interpreted as if it originated from the VirtualBox application rather than a VM.

As stated in Section 5.1, the networking between VM-VNFs is delegated to VirtualBox internal switch (*Oracle VM VirtualBox networking engine* [27]), which also ensures connectivity between VMs and the host platform. Each VM is equipped with three network adapters.

The first one, i.e. *enp0s3* for Debian distributions, is configured as NAT adapter to enable access to the internet from within the VM. The NAT interface is used as the default gateway device. According to the VBox manual [27], the IP address used for the interface is locally significant (cannot be reachable from the outside VBox Networking engine). All outbound traffic that cannot be routed using the inbuilt OS IP table is directed to the default gateway address, which equals to 10.0.2.2 (for the *enp0s3* interface on VM). This address is bounded to the loopback interface of the host OS. The IP address of the NAT interface of the guest VM is acquired from VBox DHCP engine and resides within a NAT address pool (10.0.2.0/24 [27]).

The remaining two interfaces of each VM are configured as *host-only networking* type. The network space created within VBox is logically split to reflect the responsibility of each part and ensure data separation. Upon VM deployment, VBox created two virtual interfaces on the host platform, each for the logical network space:

- *data networking space* address pool: 192.168.56.0/24
  - The virtual interface *vboxnet0* on the host OS is assigned an IP address: 192.168.56.101.
  - The networking space is exclusive used for host-to-VM access via SSH.
- *northbound networking space* address pool: 192.168.58.0/24.
  - virtual interface *vboxnet1* on the host OS is assigned an IP address 192.168.58.101.
  - This interface is dedicated to all the traffic related to SBA and reference-point communication of the deployed 5G network.

Each of the VMs binds *enp0s8* and *enp0s9* to one networking space. Interface *enp0s8* is attached to a *northbound networking space* and is used by the host platform to remotely configure the NFs on the VM. *Enp0s9* binds to the *data networking space*, in which all the 5GC and 5G-AN traffic is handled.

The OS of *vm_5GCN-UPF*, which hosts the 5G control plane functions, must support Network Address Port Translation (NAPT) to enable data tunneling to an outbound network. Furthermore, it must support IPv4 forwarding and be equipped with TUN interfaces for each of the PDU Sessions [23].

The presented network configuration is designed to verify deployed 5GCN. Additionally, it restricts access to the virtualized 5GN from the external networks (e.g. internet).

5.3. NFs configuration

In order to realize the scenario mentioned in Section 4.3 the configuration files of the 5GCN and 5G-AN should to be tailored. This involved changing the YAML files of the 5GCN Network Functions. YAML is a human-friendly markup language used for creating configuration files.

To integrate the UERANSIM and Open5GS frameworks, it is necessary to ensure consistent Public Land Mobile Network identity between the 5G-AN and 5GCN. A PLMN embraces two values:

- Mobile Country Code (*MCC*), which names the country of the deployed network [24, 16].
- Mobile Network Code (*MNC*), which denotes the code of the mobile carrier [24, 16].

The PLMN value is constantly referred in configuration files of UEs (PLMN components build up a part of IMSI - Figure 13), the gNB and AMF.

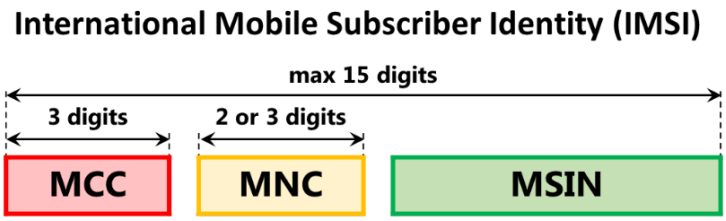

Figure 13. The IMSI components.

Source: Sploty 6.0 5G Systems Overview [24]

Additionally, both emulated UEs should be aware of the assigned S-NSSAI. The Data Network Name (DNN) for a S-NSSAI should be consistent within the deployed 5GCN. Furthermore, International Mobile Subscription Identity of the UEs should be unique (the MSIN part as per Figure 13).

UEs IMSI, together with a full Subscriber Profile, should already exist in the UDR in order to facilitate a positive authentication via AUSF. In practice, there should be an adequate entry in the MongoDB database for Open5GS.

5.3.1. AMF

In order to enable a shared AMF access to the 5GN, the common NF has been configured to support both slices in the verification scenario mentioned in the Table 4.3. The Globally Unique AMF ID (GUAMI) reflects the common PLMN and Tracking Area Identity (TAI) component - Tracking

Area Code (TAC) is shared by AMF and gNB. Listing 1 highlights these obligatory modifications of the default configuration.

The *ngap* interface of AMF, responsible for handling NAS signaling, is bound to the physical interface *enp0s9* of the *vm_5GC-UP*.

Listing 1. Listing presenting the AMF configuration to support multi-slice network topology.

```
amf:
    sbi:
      - addr: 127.0.0.5
        port: 7777
    ngap:
      - addr: 192.168.58.110
    guami:
      - plmn_id:
          mcc: 666
          mnc: 01
        amf_id:
          region: 2
          set: 1
    tai:
      - plmn_id:
          mcc: 666
          mnc: 01
        tac: 1
    plmn_support:
      - plmn_id:
          mcc: 666
          mnc: 01
        s_nssai:
          - sst: 1
            sd: 000001
          - sst: 2
            sd: 000002
```

The SBI (*sbi* in Listing 1) interface of the AMF is used for service exposure and access. A default Open5GS configuration is used since the 5GCN CP is deployed on a singe VM. Therefore, usage of a loopback interface with unique port numbering simplifies the scenario and reduces risk of failure due to a routing error.

5.3.2. SMF1 and UPF1

Each of the slices (Table 4.3) is assigned with a dedicated pair of SMF/UPF, reflecting the provisioned service. SMF1/UPF1 pair supports slice [SST: 1, SD: 0x000001], as indicated in the YAML configuration (Listing 2 and 3).

SMF1, deployed on *vm_5GC-CP*, binds its PFCP interface (the N4 interface in reference-point architecture) to *enp0s9* physical interface, with an IP address of 192.168.58.111. The same line is present in UPF1 configuration (Listing 3) to ensure a N4 link is established and PFCP peering has been achieved [28, 30].

Listing 2. Listing showing the SMF1 configuration to support INTERNET slice provisioning.

```
smf:
    sbi:
      - addr: 127.0.0.4
        port: 7777
    pfcp:
      - addr: 192.168.58.111
    gtpu:
      - addr: 192.168.58.111
    subnet:
      - addr: 10.45.0.1/16
        dnn:  internet
    dns:
      - 8.8.8.8
      - 8.8.4.4
    freeDiameter: /root/open5gs/install/etc/freeDiameter/smf.conf
    info:
      - s_nssai:
         - sst: 1
           sd: 000001
           dnn:
             - internet
        tai:
         - plmn_id:
             mcc: 666
             mnc: 01
           tac: 1
```

The *info* key in Listing 2 outlines the network slice characteristics and is a base for AMF for SMF selection procedure [28]. Therefore, it is crucial that the value assigned to the *info* key properly

reflects the desired network behavior. The *tai* key must reflect the respective key in AMF and gNB configuration (Listing 1 and Listing 8).

Additionally, the *subnet* key in Listing 2 must include the desired DNN that UE1 requests upon *UE triggered service request* procedure: *internet*. The IP address pool in the subkey *addr* refers to a range of IP addresses from which a UE IP address will be selected for a particular PDU Session. Such configuration is also reflected in UFP1 YAML file (Listing 3 - key *subnet.addr*).

Listing 3. Listing presenting the UPF1 configuration to support INTERNET slice provisioning.

```
upf:
    pfcp:
      - addr: 192.168.58.121
    gtpu:
      - addr: 192.168.58.121
    subnet:
      - addr: 10.45.0.1/16
        dnn: internet
        dev: ogstun

smf:
    pfcp:
      - addr: 192.168.58.111
```

The UPF1 configuration file has been edited in order to alter the default PFCP address, which now points to the same network as PFCP address of SMF1. The PFCP address is bound to the *enp0s9* physical interface on *vm_5GC-UPF* VM, and equals 192.168.58.121. The GTP-U interface, i.e. the N3 reference point for data transfer, also points to 192.168.58.121 address. The *subnet* values reflect the SMF1 configured IP address pool and DNN. The *dev* value denotes the tunnel interface name, which is used for data tunneling via N6 interface [24] to DN, i.e. public internet.

5.3.3. SMF2 and UPF2

The configuration of SMF2 and UPF2 for slice [SST:2 SD:0x000002] support is similar to the SMF1/UPF1. The changes however reflect the fact that both SMF1/SMF2 and UPF1/UPF2 are based on the same VM: *vm_5GC-CP* and *vm_5GC-UPF* respectively. Therefore, the IP addresses of PFCP and GTP-U interfaces, although bound to the same interfaces as addresses SMF1/UPF1, are changed:

- SMF2:
  - PFCP → 192.168.58.112
  - GTP-U → 192.168.58.112

- UPF2:
  - PFCP → 192.168.58.122
  - GTP-U → 192.168.58.122

The Listing 4 highlights the necessary changes to the SMF2 configuration. The subnet for a UE2 IP address derivation is 10.46.0.1/16. The *info* values reflect the desired slice configuration for IMS DN.

Listing 4. Listing showing the SMF2 configuration to support IMS slice provisioning.

```
smf:
    sbi:
      - addr: 127.0.0.24
        port: 7777
    pfcp:
      - addr: 192.168.58.112
    gtpu:
      - addr: 192.168.58.112
    subnet:
      - addr: 10.46.0.1/16
        dnn:  ims
    dns:
      - 8.8.8.8
      - 8.8.4.4

    freeDiameter: /root/VM5GC/2slice_topo/smf2.conf
    info:
      - s_nssai:
         - sst: 2
           sd: 000002
           dnn:
             - ims
        tai:
         - plmn_id:
             mcc: 666
             mnc: 01
           tac: 1
```

Due to the fact that NFs in the 5GCN CP are deployed on one virtual machine, the SMF2 SBI interface cannot point to the same IP address as the loopback interface of the *vm_5GC-CP*. The

SMF2 SBI address has been changed to 127.0.0.24 (port remains intact). This change is later reflected upon NF registration in the NRF [28], thus the SMF2 services are accessible via the altered SBI.

Furthermore, the *freeDiameter* configuration file has been swapped to indicate the distinction between SMF1 and SMF2 in terms of SBI addresses - Listing 5. This file is used to configure the freeDiameter server. FreeDiameter implements the Diameter protocol, which is used for carrying the Authentication, Authorization and Accounting (AAA) payload. Key *ListenOn* names the port, on which the NF will listen for authorization requests.

Listing 5. Listing showing swapping the SBI address for SMF2 in freeDiameter config file.

```
#ListenOn = "127.0.0.4";
ListenOn = "127.0.0.24";
```

According to the configuration of the UPF2 (Listing 6), the key *pfcp*, responsible for establishing N4 link, has been set to the IP address of the SMF2 PFCP interface (Listing 4). The *subnet* values for this interface differ from the UPF1 settings. To route the data traffic of UEs belonging to different PDU Sessions to the appropriate destination, the device (*dev*) responsible for forwarding UE2 data traffic points to the *ogstun2* interface on the *vm_5GC-UPF* virtual machine.

Listing 6. Listing depicting the UPF2 configuration to support IMS slice provisioning.

```
upf:
    pfcp:
      - addr: 192.168.58.122
    gtpu:
      - addr: 192.168.58.122
    subnet:
      - addr: 10.46.0.1/16
        dnn: ims
        dev: ogstun2
smf:
    pfcp:
      - addr: 192.168.58.112
```

5.3.4. NSSF

According to 3GPP Release 15 [28], the AMF obtains information about the slice configuration from the NSSF. Therefore, the NSSF YAML file must be customized for the specific test scenario to include the necessary slice information (Table 4.3).

Listing 7. Listing presenting the NSSF configuration to support slice info provisioning.

```
nssf:
    sbi:
      - addr: 127.0.0.14
        port: 7777
    nsi:
      - addr: 127.0.0.10
        port: 7777
        s_nssai:
          sst: 1
          sd: 000001
      - addr: 127.0.0.10
        port: 7777
        s_nssai:
          sst: 2
          sd: 000002
```

As presented in the Listing 7, both slices are associated with one NRF. This approach does not impose any further consequences, since both pairs of SMF/UPF are registered with the same NRF. Therefore, when the AMF requests service from any of the SMFs, it has complete knowledge that there is no other SMF outside the queried NRF within the slice, as described in the SMF selection process [28].

5.4. UERANSIM gNB and UEs

The syntax of UERANSIM configuration files differ from those used by Open5GS framework. However, the key parameters are still recognizable. In the verification scenario, a single common gNB is used to support a multi-slice infrastructure and to relay PDU session establishment requests to a shared AMF (as described in Section 4.2).

5.4.1. gNB

The UERANSIM software implementation of gNB shares the combination of MCC and MNC values with the AMF to indicate that emulated 5G-AN belongs to the same PLMN as the 5GCN (Listing 8). The configuration file for the gNB also specifies the *tac* value (tac: 1), which is present in AMF (Section 5.3.1) and SMF1/SMF2 YAML files (Section 4.4.3 and 4.4.2) for each of the supported slices. As per network configuration, the Radio link, N2 and N3 interfaces are bound to the *enp0s9* with an IP address of 192.168.58.130.

Listing 8. Listing presenting the gNB configuration to support slice provisioning.

```
mcc: '666'          # Mobile Country Code value
mnc: '01'           # Mobile Network Code value (2 or 3 digits)
tac: 1              # Tracking Area Code

linkIp: 192.168.58.130   # gNB's local IP address for Radio Link
ngapIp: 192.168.58.130   # gNB's local IP address for N2 Interface
gtpIp: 192.168.58.130    # gNB's local IP address for N3 Interface

amfConfigs:
  - address: 192.168.58.110
    port: 38412

slices:
  - sst: 1
    sd: 0x000001
  - sst: 2
    sd: 0x000002
```

In the verification setup, the gNB is associated with a single AMF, which is identified by the IP address 192.168.58.110. The gNB also supports both slices, as indicated in Table 4.3.

5.4.2. UE1 and UE2

The UEs are configured with the initial slice to the topology shown in Figure 12.

The configuration files for UE1 and UE2 (Listings 9 and 10) specify the PLMN compliance, identify the gNB that provides access to the AN, and describe the default/supported NSSAIs. UE1 and UE2 have been assigned unique IMSIs, which have been registered in the MongoDB subscriber database.

Listing 9. Listing presenting the UE1 configuration and the default slice info.

```
supi: 'imsi-666010000000001'
mcc: '666'
mnc: '01'

# List of gNB IP addresses for Radio Link Simulation
gnbSearchList:
  - 192.168.58.130

# Initial PDU sessions to be established
```

```
sessions:
  - type: 'IPv4'
    apn: 'internet'
    slice:
      sst: 1
      sd: 0x000001

configured-nssai:
  - sst: 1
    sd: 0x000001
default-nssai:
  - sst: 1
    sd: 0x000001
```

Listing 10. Listing presenting the UE2 configuration and the default slice info.

```
supi: 'imsi-666010000000002'
mcc: '666'
mnc: '01'

# List of gNB IP addresses for Radio Link Simulation
gnbSearchList:
  - 192.168.58.130

# Initial PDU sessions to be established
sessions:
  - type: 'IPv4'
    apn: 'ims'
    slice:
      sst: 2
      sd: 0x000002

configured-nssai:
  - sst: 2
    sd: 0x000002

default-nssai:
  - sst: 2
    sd: 0x000002
```

6. Core network topology verification

As mentioned in Section 4.3, the verification of the virtualized 5GCN consists of two subparts: Initial Registration and UE triggered service request. However, since the UEs are configured to initiate the PDU session request immediately after a 5GN access, both procedures are combined.

Section 6.1 discusses the process of NFs registering to NRF. Section 6.2 focuses on accessing the 5G PDU service for the specified DN. Section 6.3 presents all issues encountered during the proposed verification scenario. Finally, Section 6.4 shows a successful establishment of the PDU session.

6.1. 5G NFs registering

In order to reduce risk of 5GN initialization error, the following deployment schedule has been adopted:

1. 5GCN CP: starting all Open5GS Control Plane NFs.
2. 5GCN UP:
   (a) Enabling NAT and tunneling.
   (b) Starting Open5GS Control Plane NFs: UPF1 and UPF2.
3. 5G-AN: Starting UERANSIM gNB instance.

The execution order of 5GCN CP is as follows:

NRF → AMF → UDR → UDM → NSSF → AUSF → PCF → BSF → SMF1 → SMF2

Each of the NFs perform registration procedure to the NRF through its SBI dedicated to NF Management (Figure 14).

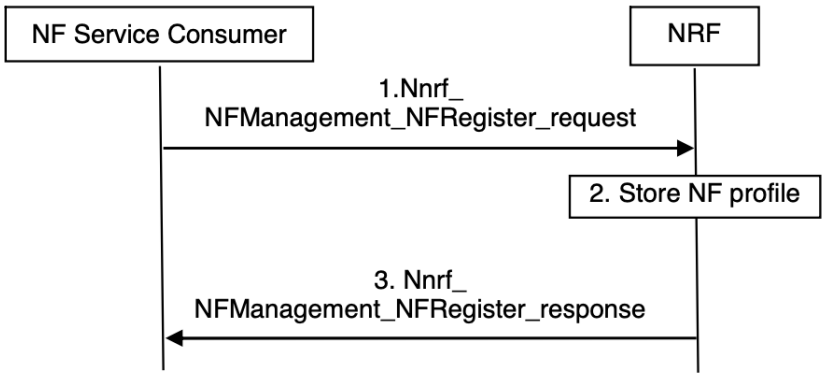

Figure 14. Nnrf_NF Registration procedure of a NF to NRF.

Source: 3GPP Release 15 5G Systems Procedures [28]

In practice, the NFs post HTTP/2 requests to the proper REST API [34] managing a given service. In case of a NF registration, the HTTP/2 PUT-type request is issued to the */nnrf-nfm/v1/nf-instances/* endpoint of the NRF.

Below, the results of the verification scenario (Section 4.3) of the deployed 5G network (Section 5) are presented.

Listing 11. AMF registration to the NRF.

```
 11/06 15:17:04.802: [sbi] DEBUG: [PUT]
/nnrf-nfm/v1/nf-instances/1236f57a-5de6-41ed-9e10-15ca63749b13
 11/06 15:17:04.802: [sbi] DEBUG: {
 "nfInstanceId": "AMF",
 "nfType": "1236f57a-5de6-41ed-9e10-15ca63749b13",
 "nfStatus": "REGISTERED",
 "ipv4Addresses": ["127.0.0.5"],
 "allowedNfTypes": ["SMF " "SCP"],
 "priority": 0,
 "capacity": 100,
 "load": 0,
 "amfInfo": {
    "amfSetId": "001",
    "amfRegionId": "02",
    "guamiList": [{
        "plmnId": {
            "mcc": "666",
            "mnc": "01"
            },
        "amfId": "020040"
        }],
    "taiList": [{
        "plmnId": {
            "mcc": "666",
            "mnc": "01",
            },
        "tac": "000001"
        }],
    },
    "nfServiceList":{
        12384e0c-5de6-41ed-9e10-15ca63749b13: {
            "serviceInstanceId": "12384e0c-5de6-41ed-9e10-15ca63749b13",
            "serviceName": "namf-comm",
            "versions": [{
                "apiVersionInUri": "V1",
                "apiFullVersion": "1.0.0"
                }],
```

```
            "scheme": "http",
            "nfServiceStatus": "REGISTERED",
            "ipEndPoints": [{
                "ipv4Address": "127.0.0.5",
                "port": 7777
                }],
            "allowedNfTypes": ["SMF"],
            "priority": 0,
            "capacity": 100,
            "load": 0
            }
        },
    "nfProfileChanges Support Ind": true
    }
```

Listing 11 presents an exemplary NF registration of AMF, which contains all the necessary information as specified in 3GPP Release 15 [11].

The PUT HTTP/2 request issued by AMF shows the modified PLMN and TAC values (mentioned in Section 5.3.1).

SMF1 and SMF2 registration requests to the common NRF should include slice information (Table 4.3).

Listing 12 presents the procedure for SMF1 to provision a PDU session establishment for a slice through registration [SST: 1, SD:0x000001]. The PUT request reflects the necessary information that is to be used by AMF in SMF selection for PDU Session establishment request (the S-NSSAI and supported DNNs).

Listing 12. SMF1 registration to the NRF.

```
11/06 15:17:09.480: [sbi] DEBUG: [PUT]
http://127.0.0.10:7777/nnrf-nfm/v1/nf-instances/14f578cc-5de6-41ed-a909-6f79ebdf738b
(../lib/sbi/client.c:678)
11/06 15:17:09.480: [sbi] DEBUG: SENDING...[1000] (../lib/sbi/client.c:396)
11/06 15:17:09.486: [sbi] DEBUG: {
    "nfInstanceId":    "14f578cc-5de6-41ed-a909-6f79ebdf738b",
    "nfType":    "SMF",
    "nfStatus":    "REGISTERED",
    "ipv4Addresses":    ["127.0.0.4"],
    "allowedNfTypes":    ["AMF", "SCP"],
    "priority":    0,
    "capacity":    100,
    "load":    0,
    "smfInfo":    {
        "sNssaiSmfInfoList":    [{
```

```
                "sNssai":    {
                    "sst":    1,
                    "sd":    "000001"
                },
                "dnnSmfInfoList":    [{
                        "dnn":    "internet"
                    }]
            }],
        "taiList":    [{
                "plmnId":    {
                    "mcc":    "666",
                    "mnc":    "01"
                },
                "tac":    "000001"
            }]
    },
    "nfServiceList":    {
        "1503e75e-5de6-41ed-a909-6f79ebdf738b":    {
            "serviceInstanceId":    "1503e75e-5de6-41ed-a909-6f79ebdf738b",
            "serviceName":    "nsmf-pdusession",
            "versions":    [{
                    "apiVersionInUri":    "v1",
                    "apiFullVersion":    "1.0.0"
                }],
            "scheme":    "http",
            "nfServiceStatus":    "REGISTERED",
            "ipEndPoints":    [{
                    "ipv4Address":    "127.0.0.4",
                    "port":    7777
                }],
            "allowedNfTypes":    ["AMF"],
            "priority":    0,
            "capacity":    100,
            "load":    0
        }
    },
    "nfProfileChangesSupportInd":    true
} (../lib/sbi/client.c:398)
```


The SMF2 configuration is also properly provided in NF registration request - Listing 13 shows the supported S-NSSAI and DN in the field *smfInfo*.

Listing 13. SMF2 registration to the NRF, with the outline of provided services.

```
11/06 15:17:14.518: [sbi] DEBUG: [PUT]
http://127.0.0.10:7777/nnrf-nfm/v1/nf-instances/17fbb676-5de6-41ed-a777-83634a1d1a33
(../lib/sbi/client.c:678)
11/06 15:17:14.518: [sbi] DEBUG: SENDING...[997] (../lib/sbi/client.c:396)
11/06 15:17:14.524: [sbi] DEBUG: {
    "nfInstanceId":    "17fbb676-5de6-41ed-a777-83634a1d1a33",
    "nfType":    "SMF",
    "nfStatus":    "REGISTERED",
    "ipv4Addresses":    ["127.0.0.24"],
    "allowedNfTypes":    ["AMF", "SCP"],
    "priority":    0,
    "capacity":    100,
    "load":    0,
    "smfInfo":    {
        "sNssaiSmfInfoList":    [{
                "sNssai":    {
                    "sst":    2,
                    "sd":    "000002"
                },
                "dnnSmfInfoList":    [{
                        "dnn":    "ims"
                    }]
            }],
        "taiList":    [{
                "plmnId":    {
                    "mcc":    "666",
                    "mnc":    "01"
                },
                "tac":    "000001"
            }]
    },
    "nfServiceList":    {
        "18049642-5de6-41ed-a777-83634a1d1a33":    {
            "serviceInstanceId":    "18049642-5de6-41ed-a777-83634a1d1a33",
            "serviceName":    "nsmf-pdusession",
            "versions":    [{
                    "apiVersionInUri":    "v1",
                    "apiFullVersion":    "1.0.0"
                }],
            "scheme":    "http",
            "nfServiceStatus":    "REGISTERED",
            "ipEndPoints":    [{
```

```
                "ipv4Address":    "127.0.0.24",
                "port":    7777
            }],
        "allowedNfTypes":    ["AMF"],
        "priority":    0,
        "capacity":    100,
        "load":    0
    }
},
"nfProfileChangesSupportInd":    true
```

After the 5G core network control plane functions are registered, the 5GCN Userplane Functions can be deployed. Before calling the UPF1/UPF2 executable on the *vm_5GCN-UPF VM*, it is necessary to ensure that there is a port forwarding capability and TUN interfaces for data tunneling. Listing 14 presents the commands that must be executed on the Debian operating system in order to achieve this.

Listing 14. Necessary network configuration of the UPF VM.

```
ip tuntap add name ogstun mode tun
ip tuntap add name ogstun2 mode tun

ip addr add 10.45.0.1/16 dev ogstun
ip addr add 10.46.0.1/16 dev ogstun2

ip link set ogstun up
ip link set ogstun2 up

iptables -t nat -A POSTROUTING -s 10.45.0.0/16 ! -o ogstun -j MASQUERADE
iptables -t nat -A POSTROUTING -s 10.46.0.0/16 ! -o ogstun2 -j MASQUERADE
```

Upon running the commands presented in Listing 14, both UPF1 and UPF2 NFs have been deployed.

The last step in ensuring that the virtualized 5G network has been fully deployed is executing the software emulating gNB provided by UERANSIM. Then, the gNB establishes N2 link with the 5GCN AMF. Listing 15 shows an AMF view of gNB registration - the SCTP interface between 5G-AN and 5G core has been set up.

Listing 15. gNB issued NG link (N2) setup request to the AMF.

```
11/05 14:13:56.827: [amf] INFO: gNB-N2 accepted[192.168.58.130]:59305 in ng-path module
(../src/amf/ngap-sctp.c:113)
11/05 14:13:56.828: [amf] DEBUG: amf_state_operational(): AMF_EVENT_NGAP_LO_ACCEPT
(../src/amf/amf-sm.c:78)
```

```
11/05 14:13:56.828: [amf] INFO: gNB-N2 accepted[192.168.58.130] in master_sm module
(../src/amf/amf-sm.c:664)
11/05 14:13:56.828: [amf] INFO: [Added] Number of gNBs is now 1 (../src/amf/context.c:973)
11/05 14:13:56.828: [amf] DEBUG: SCTP_ASSOC_CHANGE:[T:32769, F:0x0, S:0, I/O:10/10]
(../src/amf/ngap-sctp.c:152)
11/05 14:13:56.828: [amf] DEBUG: SCTP_COMM_UP (../src/amf/ngap-sctp.c:161)
11/05 14:13:56.829: [amf] DEBUG: NGSetupRequest (../src/amf/ngap-handler.c:128)
11/05 14:13:56.829: [amf] DEBUG:     IP[192.168.58.130] GNB_ID[0x1]
(../src/amf/ngap-handler.c:176)
11/05 14:13:56.829: [amf] DEBUG:     PagingDRX[2] (../src/amf/ngap-handler.c:179)
11/05 14:13:56.829: [amf] DEBUG:     TAC[1] (../src/amf/ngap-handler.c:202)
11/05 14:13:56.829: [amf] DEBUG:     PLMN_ID[MCC:666 MNC:1] (../src/amf/ngap-handler.c:230)
11/05 14:13:56.829: [amf] DEBUG:     S_NSSAI[SST:1 SD:0x1] (../src/amf/ngap-handler.c:269)
11/05 14:13:56.829: [amf] DEBUG:     S_NSSAI[SST:1 SD:0x2] (../src/amf/ngap-handler.c:269)
11/05 14:13:56.829: [amf] DEBUG:     TAC[1] (../src/amf/ngap-handler.c:40)
11/05 14:13:56.829: [amf] DEBUG:     PLMN_ID[MCC:666 MNC:1] (../src/amf/ngap-handler.c:41)
11/05 14:13:56.829: [amf] DEBUG:     SERVED_TAI_INDEX[0] (../src/amf/ngap-handler.c:46)
11/05 14:13:56.829: [amf] DEBUG:     PLMN_ID[MCC:666 MNC:1] (../src/amf/ngap-handler.c:68)
11/05 14:13:56.829: [amf] DEBUG:     S_NSSAI[SST:1 SD:0x1] (../src/amf/ngap-handler.c:73)
11/05 14:13:56.829: [amf] DEBUG: NG-Setup response (../src/amf/ngap-path.c:254)
11/05 14:13:56.829: [amf] DEBUG: NGSetupResponse (../src/amf/ngap-build.c:36)
11/05 14:13:56.829: [amf] DEBUG:     IP[192.168.58.130] RAN_ID[1]
(../src/amf/ngap-path.c:70)
11/05 14:14:03.402: [amf] DEBUG: amf_state_operational(): OGS_EVENT_NAME_SBI_TIMER
(../src/amf/amf-sm.c:78)
```

The AMF logs contain the gNB's provided data on supported slices, TAC, and PLMN, which match the configured values as mentioned in Section 5.3.1. The gNB console also confirms that an SCPT connection through the N2 link has been established with the AMF, as shown on Figure 15.

```
vm5G-UERANSIM:~/UERANSIM/build# ./nr-gnb -c ../../VM5GC/2slice_topo/ran_config/gnb_2slices.yaml
UERANSIM v3.2.6
[2022-11-06 15:24:56.167] [sctp] [info] Trying to establish SCTP connection... (192.168.58.110:38412)
[2022-11-06 15:24:56.199] [sctp] [info] SCTP connection established (192.168.58.110:38412)
[2022-11-06 15:24:56.199] [sctp] [debug] SCTP association setup ascId[3]
[2022-11-06 15:24:56.200] [ngap] [debug] Sending NG Setup Request
[2022-11-06 15:24:56.206] [ngap] [debug] NG Setup Response received
[2022-11-06 15:24:56.206] [ngap] [info] NG Setup procedure is successful
```

Figure 15. The UERANSIM gNB console, which shows a successful N2 link establishment.

The process of 5G network deployment is complete when the gNB registers and establishes an N2 connection. This process is successful even when the various parts of the 5G network are distributed across multiple virtual machines.

6.2. Slice-aware PDU Session establishment

Since the 5G network deployment process has been successfully completed, a proper verification scenario was carried out. The execution order of the UE in this scenario mirrors the SMF execution

order described in Section 6.1: SMF2 is called after SMF1 has registered itself to the NRF. UE1 is called before the UE2 on UERANSIM.

```
[nas] [info] UE switches to state [MM-DEREGISTERED/PLMN-SEARCH]
[rrc] [debug] New signal detected for cell[1], total [1] cells in coverage
[nas] [info] Selected plmn[666/01]
[rrc] [info] Selected cell plmn[666/01] tac[1] category[SUITABLE]
[nas] [info] UE switches to state [MM-DEREGISTERED/PS]
[nas] [info] UE switches to state [MM-DEREGISTERED/NORMAL-SERVICE]
[nas] [debug] Initial registration required due to [MM-DEREG-NORMAL-SERVICE

[nas] [debug] UAC access attempt is allowed for identity[0], category[MO_si

[nas] [debug] Sending Initial Registration
[rrc] [debug] Sending RRC Setup Request
[rrc] [info] RRC connection established
[rrc] [info] UE switches to state [RRC-CONNECTED]
[nas] [info] UE switches to state [MM-REGISTER-INITIATED]
[nas] [info] UE switches to state [CM-CONNECTED]
[nas] [debug] Authentication Request received
[nas] [debug] Security Mode Command received
[nas] [debug] Selected integrity[2] ciphering[0]
[nas] [debug] Registration accept received
[nas] [info] UE switches to state [MM-REGISTERED/NORMAL-SERVICE]
[nas] [debug] Sending Registration Complete
[nas] [info] Initial Registration is successful
```

Figure 16. A complete view of the UERANSIM UE1 console which outlines a positive attachment to the 5G-AN.

The UE attachment was successfully achieved, as shown in Figure 16 - the UE1 is in registered state. This suggests that the UE1 setup parameters match the subscriber profile in the Open5GS database. The AMF logs also confirm UE1 registration (Figure 17), as the registration process for its IMSI (666010000000001) is complete.

```
[gmm] DEBUG: gmm_state_initial_context_setup(): OGS_EVENT_NAME_SBI_CLIENT (../src/amf/gmm-sm.c:962)
[amf] DEBUG: [imsi-666010000000001] Registration accept (../src/amf/nas-path.c:85)
[gmm] DEBUG: [imsi-666010000000001]    5G-S_GUTI[AMF_ID:0x20040,M_TMSI:0xd600ab53] (../src/amf/gmm-build.c:75)
[gmm] DEBUG: [imsi-666010000000001]    TAI[PLMN_ID:66f610,TAC:1] (../src/amf/gmm-build.c:89)
[gmm] DEBUG: [imsi-666010000000001]    NR_CGI[PLMN_ID:66f610,CELL_ID:0x10] (../src/amf/gmm-build.c:91)
[gmm] DEBUG: [imsi-666010000000001]    SERVED_TAI_INDEX[0] (../src/amf/gmm-build.c:96)
[amf] DEBUG: InitialContextSetupRequest(UE) (../src/amf/ngap-build.c:477)
[amf] DEBUG:     RAN_UE_NGAP_ID[2] AMF_UE_NGAP_ID[2] (../src/amf/ngap-build.c:626)
[amf] DEBUG:     IP[192.168.58.130] RAN_ID[1] (../src/amf/ngap-path.c:70)
[amf] DEBUG: amf_state_operational(): AMF_EVENT_NGAP_MESSAGE (../src/amf/amf-sm.c:78)
[amf] DEBUG: ngap_state_operational(): AMF_EVENT_NGAP_MESSAGE (../src/amf/ngap-sm.c:53)
[amf] DEBUG: InitialContextSetupResponse (../src/amf/ngap-handler.c:788)
[amf] DEBUG:     IP[192.168.58.130] RAN_ID[1] (../src/amf/ngap-handler.c:808)
[amf] DEBUG:     RAN_UE_NGAP_ID[2] AMF_UE_NGAP_ID[2] (../src/amf/ngap-handler.c:840)
[amf] DEBUG: amf_state_operational(): AMF_EVENT_NGAP_MESSAGE (../src/amf/amf-sm.c:78)
[amf] DEBUG: ngap_state_operational(): AMF_EVENT_NGAP_MESSAGE (../src/amf/ngap-sm.c:53)
[amf] DEBUG: UplinkNASTransport (../src/amf/ngap-handler.c:546)
[amf] DEBUG:     IP[192.168.58.130] RAN_ID[1] (../src/amf/ngap-handler.c:568)
[amf] DEBUG:     RAN_UE_NGAP_ID[2] AMF_UE_NGAP_ID[2] TAC[1] CellID[0x10] (../src/amf/ngap-handler.c:647)
[amf] DEBUG: amf_state_operational(): AMF_EVENT_5GMM_MESSAGE (../src/amf/amf-sm.c:78)
[gmm] DEBUG: gmm_state_initial_context_setup(): AMF_EVENT_5GMM_MESSAGE (../src/amf/gmm-sm.c:962)
[gmm] INFO: [imsi-666010000000001] Registration complete (../src/amf/gmm-sm.c:1127)
[amf] INFO: [imsi-666010000000001] Configuration update command (../src/amf/nas-path.c:430)
```

Figure 17. A successful UE1 access to the implemented 5G network from AMF viewpoint.

6.3. Analysis and resolution of crash prone scenario

During the initial registration, the UE also requests the establishment of a PDU session. Although the registration to the network was successful, the session establishment failed.

After multiple attempts to establish a PDU session, the UE1 fails to receive any response to its *InitialUEMessage* and switches to the IDLE state. The AMF logs allowed tracing the cause of this issue in the implemented 5GCN. Listing 16 shows the AMF's inability to connect to a HTTP/2 server at the 127.0.0.24 address.

Listing 16. AMF logged warning related to NF HTTP/2 server inaccessibility.

```
11/05 14:14:45.431: [sbi] WARNING: [7] Failed to connect to 127.0.0.4 port 7777: Connection
refused (../lib/sbi/client.c:585)
11/05 14:14:45.431: [amf] WARNING: client_cb() failed [-1] (../src/amf/sbi-path.c:56)
```

The address mentioned in the Listing 16 belongs to the SBI interface of SMF2 (as configured in Section 5.3.3). The AMF requested session establishment in an invalid SMF, as it would be expected to query SMF1. Since SMF2 is located in a different logical network, the UE1 PDU session establishment request should not have been forwarded to this specific network function.

Further investigation unveiled that the SMF2 crashed due to an unsupported subnet: [family: 2, dnn: internet], as shown on Listing 17. The error suggests that the process of selecting the SMF might be flawed.

Listing 17. A SMF2 crash backtrace.

```
11/05 14:16:32.760: [smf] DEBUG: smf_state_operational(): OGS_EVENT_NAME_SBI_CLIENT
(../src/smf/smf-sm.c:83)
11/05 14:16:32.760: [smf] DEBUG: smf_gsm_state_wait_5gc_sm_policy_association():
OGS_EVENT_NAME_SBI_CLIENT (../src/smf/gsm-sm.c:480)
11/05 14:16:32.760: [pfcp] ERROR: CHECK CONFIGURATION: Cannot find subnet [family:2,
dnn:internet] (../lib/pfcp/context.c:1636)
11/05 14:16:32.760: [pfcp] ERROR: Please add FALLBACK subnet as below.
(../lib/pfcp/context.c:1638)
11/05 14:16:32.760: [pfcp] ERROR:     subnet: (../lib/pfcp/context.c:1639)
11/05 14:16:32.760: [pfcp] ERROR:       - addr: 10.50.0.1/16 (../lib/pfcp/context.c:1641)
```

During further investigation, it was revealed that the SMF2 has received a *Nsmf_PDUSession_CreateSMContext* request for a DN with unsupported slice combination: [SST: 1, SD:0x000001] - Listing 18.

Listing 18. An invalid PDU Session establishment request posted to SMF2.

```
11/05 14:16:32.754: [sbi] DEBUG: [POST] /nsmf-pdusession/v1/sm-contexts
(../lib/sbi/nghttp2-server.c:800)
11/05 14:16:32.754: [sbi] DEBUG: RECEIVED: 1142 (../lib/sbi/nghttp2-server.c:804)
11/05 14:16:32.755: [sbi] DEBUG: --=-KepEqNu8K4rVigBUOK4YlQ==
Content-Type: application/json

{
    "supi":    "imsi-666010000000001",
    "pei":    "imeisv-4370816125816151",
    "pduSessionId":    1,
    "dnn":    "internet",
    "sNssai":    {
```

```
        "sst":    1,
        "sd":    "000001"
    },
```

Upon further verification, the AMF has been found to cause the SMF2 to crash by issuing a request to establish PDU session to a DN (Listing 19), which is unsupported by the targeted SMF.

Listing 19. AMF failing to select an adequate SMF for a slice-aware PDU session establishment.

```
11/05 15:10:15.173: [gmm] INFO: UE SUPI[imsi-666010000000001] DNN[internet] S_NSSAI[SST:1
SD:0x1] (../src/amf/gmm-handler.c:1062)
11/05 15:10:15.173: [sbi] DEBUG: [POST]
http://127.0.0.24:7777/nsmf-pdusession/v1/sm-contexts (../lib/sbi/client.c:678)
11/05 15:10:15.173: [sbi] DEBUG: SENDING...[1142] (../lib/sbi/client.c:396)
11/05 15:10:15.173: [sbi] DEBUG: ---edDbhslS4YBpe4hFtxQp8A==
Content-Type: application/json

{
    "supi":    "imsi-666010000000001",
    "pei":    "imeisv-4370816125816151",
    "pduSessionId":    1,
    "dnn":    "internet",
    "sNssai":    {
        "sst":    1,
        "sd":    "000001"
    },
```

At this point, the investigation required a deeper look into the ”UE Requested PDU Session Establishment” process, as presented in 3GPP 5G Systems procedures [28]. As shown in Figure 11, before issuing a faulty *Nsmf_PDUSession_CreateSMContext* request, the AMF performs an internal SMF selection procedure. This procedure, described in Section 4.3.2.2.3 of 3GPP 5G Systems procedures [28], may be skipped if the AMF has local access to SMF information.

This explains the lack of the actual SMF selection procedure, as both SMF1 and SMF2 are located within one NSI and are registered with a common NRF (see Section 5.3.4). The AMF obtains SMF information through a notification sent by the NRF to the */nnrf-nfm/v1/nf-status-notify* endpoint and stores it until its deregistration.

A reverse engineering approach was adopted to verify why AMF fails to anchor the PDU Session in the adequate SMF. Upon analyzing the AMF source code (Listing 20) responsible for handling UL NAS Transport, it appears that if the network function doesn't have any information about the UE SMF session, it searches through all known NF instances that provide the *Nsmf_PDUSession* service.

If an NF is found, it is stored in the AMF session data and is assigned to the *nf_instance* variable. Since the 5G network NFs registering has been verified, both SMFs information is recorded in the AMF session.

Listing 20. Open5GS AMF source code: gmm_handle_ul_nas_transport.c.

```
if (!SESSION_CONTEXT_IN_SMF(sess)) {
                ogs_sbi_nf_instance_t *nf_instance = NULL;
                ogs_sbi_service_type_e service_type =
                    OGS_SBI_SERVICE_TYPE_NSMF_PDUSESSION;

                nf_instance = sess->sbi.
                    service_type_array[service_type].nf_instance;
                if (!nf_instance) {
                    OpenAPI_nf_type_e requester_nf_type =
                                NF_INSTANCE_TYPE(ogs_sbi_self()->nf_instance);
                    ogs_assert(requester_nf_type);

                    nf_instance = ogs_sbi_nf_instance_find_by_service_type(
                                    service_type, requester_nf_type);
                    if (nf_instance)
                        OGS_SBI_SETUP_NF_INSTANCE(
                                sess->sbi.service_type_array[service_type],
                                nf_instance);
                }

                if (nf_instance) {
                    ogs_assert(true ==
                        amf_sess_sbi_discover_and_send(
                            OGS_SBI_SERVICE_TYPE_NSMF_PDUSESSION, NULL,
                            amf_nsmf_pdusession_build_create_sm_context,
                            sess, AMF_CREATE_SM_CONTEXT_NO_STATE, NULL));
                } else {
                    ogs_assert(true ==
                        amf_sess_sbi_discover_and_send(
                            OGS_SBI_SERVICE_TYPE_NNSSF_NSSELECTION, NULL,
                            amf_nnssf_nsselection_build_get, sess, 0, NULL));
                }
```

The cause of the faulty SMF selection may be related to the *nf_instance* itself. Since SMF2 was called after SMF1, it may have been chosen as the desired node for requesting the session establishment. To address this issue, the execution order of the Session Management Functions was reversed.

Contrary to the expectations, a reversed execution of SMFs did not solve the issue regarding SMF selection mismatch. AMF continued to issue *Nsmf_PDUSession_CreateSM- Context* Requests for UE1 PDU session establishment to an invalid SMF2.

A number of combinations regarding NFs/UEs execution order were checked:

- SMF1 → SMF2 → UE1 PDU session → UE2 PDU session
  - UE1 PDU session requested on SMF2, resulting in SMF2 crash.
- SMF1 → SMF2 → UE2 PDU session → UE1 PDU session
  - UE2 PDU session was successfully established, however UE1 request has been issued to an invalid SMF2, resulting in its crash.
- SMF2 → SMF1 → UE2 PDU session → UE1 PDU session
  - UE2 PDU session was issued to SMF1, which caused it to crash.

After several attempts, it became clear that it is impossible to resolve the issue by changing the sequence of the network functions deployment.

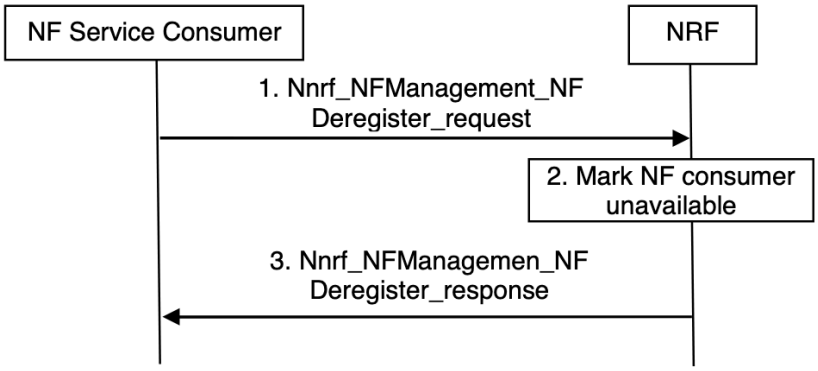

Figure 18. NF deregistration procedure upon a graceful exit to the NRF.

Source: 3GPP Release 15 5G Systems Procedures [28]

Due to the malfunction of the internal SMF selection switch within the AMF, it became impossible to achieve the desired multi-slice configuration, as specified in verification scenario (Section 4.3) with adopted NFs configuration (see Section 5.3).

Unfortunately, the documentation of Open5GS does not provide the possible fix for the SMF selection issue. SMF selection issue has not been reported by the Open5GS community. Therefore, a complete set of log files has been gathered to inform about a potential vulnerability in the AMF code

in a specific multi-slice scenario. The issue might be related to the case when multiple S-NSSAIs are associated with one NSI, however this has not been proven.

Therefore, I experimented with procedures related to the UE-triggered service request (Section 4.2) and NF deregistering (as in Figure 18). A resolution was proposed to address the issue:

1. Ensure that NFs have registered to the NRF, with an exclusion of SMF2.
2. Trigger a registration with PDU Session establishment for the UE1.
3. Verify that the PDU Session has been anchored in UPF1 and the NF is capable of tunneling UE traffic to an outbound DN.
4. Utilize the UERANSIM provided *command line interface* (CLI) for UE1 and trigger the UE-initiated PDU session request.
5. Invoke the NF deregister procedure (Figure 18) of SMF1 by stopping its process on vm_5GC-CP VM.
6. Call SMF2 executable and trigger UE2 registration.
7. Verify that PDU Session for UE2 has been anchored in the correct slice.
8. Deploy SMF1 NF.
9. Verify that PDU Session for UE1 has been reestablished (i.e. AMF issued the request to SMF1 instead of SMF2), avoiding the crash scenario.

6.4. Successful PDU session establishment

The workaround proposed in Section 6.3 would not be suitable for use in an actual 5GN deployment, as it involves dynamic phasing in and out of network functions to provide the requested service. Although this approach is not effective, it allowed to address the SMF crash incident without changing the NFs configuration (Section 5.3).

As the UE1 successfully accessed the 5GN, the *Nsmf_PDUSession_CreateSM- Context* Request has been issued to the SMF1, which registers the PDU session in PCF [28] (Listing 21), proving a positive workflow of procedure regarding PDU Session establishment (as per Figure 11).

```
11/06 15:24:59.110: [smf] DEBUG: UE using UPF on IP[192.168.58.121] (../src/smf/context.c:1067)
11/06 15:24:59.110: [smf] INFO: UE SUPI[imsi-666010000000001] DNN[internet] IPv4[10.45.0.2] IPv6[] (../src/smf/npcf-handler.c:495)
11/06 15:24:59.110: [pfcp] DEBUG: [43] LOCAL  Create  peer [192.168.58.121]:8805 (../lib/pfcp/xact.c:112)
```

Figure 19. Correct UPF selection for UE1 requested DN.

Furthermore, the correct UPF was selected for the DN requested by UE1 (refer to Table 4.3). N4 link has been established between SMF1 and UPF1, fulfilling the requirements for a slice-aware session establishment, as shown in Figure 19.

Listing 21. A SMF1 log exert showing registering PDU context for UE1 data session on PCF.

```
11/06 15:24:59.090: [sbi] DEBUG: [POST]
http://127.0.0.13:7777/npcf-smpolicycontrol/v1/sm-policies
11/06 15:24:59.091: [sbi] DEBUG: {
    "supi":    "imsi-666010000000001",
    "pduSessionId":    1,
    "pduSessionType":    "IPV4",
    "dnn":    "internet",
    "notificationUri":    "http://127.0.0.4:7777/nsmf-callback/v1/sm-policy-notify/1",
    "ipv4Address":    "10.45.0.2",
    "subsSessAmbr":    {
        "uplink":    "1048576 Kbps",
        "downlink":    "1048576 Kbps"
    },
    "subsDefQos":    {
        "5qi":    9,
        "arp":    {
            "priorityLevel":    8,
            "preemptCap":    "NOT_PREEMPT",
            "preemptVuln":    "NOT_PREEMPTABLE"
        },
        "priorityLevel":    8
    },
    "sliceInfo":    {
        "sst":    1,
        "sd":    "000001"
    },
    "suppFeat":    "4000000"}
[...]
11/06 15:24:59.110: [sbi] DEBUG: [201:POST]
http://127.0.0.13:7777/npcf-smpolicycontrol/v1/sm-policies
11/06 15:24:59.110: [sbi] DEBUG: {
    "sessRules":    {
        "1":    {
            "authSessAmbr":    {
                "uplink":    "1048576 Kbps",
                "downlink":    "1048576 Kbps"
            },
            "authDefQos":    {
                "5qi":    9,
                "arp":    {
                    "priorityLevel":    8,
```

```
                    "preemptCap":    "NOT_PREEMPT",
                    "preemptVuln":    "NOT_PREEMPTABLE"
                },
                "priorityLevel":    8
            },
            "sessRuleId":    "1"
        }
    },
    "suppFeat":    "4000000"
}
```


UE1 has been assigned an IP address of 10.45.0.2/16, which is reflected in a tunnel interface created on 5G-AN VM (Figure 20) To verify the successful service establishment by the virtualized 5GCN, a PING to an external site has been issued from the PDU specific tunnel interface created on *vm_5G-UERANSIM - uesimtun0*.

Below, indicators of a positive execution of *UE triggered PDU Session establishment* are presented:

- Connection setup and tunnel interface creation (*uesimtun0*) on UERANSIM UE1 console - Figure 20.
- PDU resource setup for UE1 on UERANSIM gNB console - Figure 21.
- Connected status of UE1 on UERANSIM UE1 CLI - Figure 22.
- A successful data transmission over the 5GCN UP to a remote site: kt.agh.edu.pl - Figure 23.

Additionally, GTP-U packets transmitting user data (i.e. the ICMP Ping data) are visible in UPF1 logs (Figure 24).

```
[2022-11-06 15:24:59.044] [nas] [info] Initial Registration is successful
[2022-11-06 15:24:59.044] [nas] [debug] Sending PDU Session Establishment Request
[2022-11-06 15:24:59.048] [nas] [debug] UAC access attempt is allowed for identity[0], category[MO_si
g]
[2022-11-06 15:24:59.250] [nas] [debug] Configuration Update Command received
[2022-11-06 15:24:59.300] [nas] [debug] PDU Session Establishment Accept received
[2022-11-06 15:24:59.301] [nas] [info] PDU Session establishment is successful PSI[1]
[2022-11-06 15:24:59.329] [app] [info] Connection setup for PDU session[1] is successful, TUN interfa
ce[uesimtun0, 10.45.0.2] is up.
```

Figure 20. UE1 UERANSIM console with status for PDU session.

```
[2022-11-06 15:24:58.960] [rrc] [debug] UE[1] new signal detected
[2022-11-06 15:24:58.965] [rrc] [info] RRC Setup for UE[1]
[2022-11-06 15:24:58.966] [ngap] [debug] Initial NAS message received from UE[1]
[2022-11-06 15:24:59.043] [ngap] [debug] Initial Context Setup Request received
[2022-11-06 15:24:59.300] [ngap] [info] PDU session resource(s) setup for UE[1] count[1]
```

Figure 21. 5G-AN view of UE1 PDU service access.

```
vm5G-UERANSIM:~/UERANSIM/build# ./nr-cli  imsi-666010000000001
--------------------------------------------------------------------
$ status
cm-state: CM-CONNECTED
rm-state: RM-REGISTERED
mm-state: MM-REGISTERED/NORMAL-SERVICE
5u-state: 5U1-UPDATED
sim-inserted: true
selected-plmn: 666/01
current-cell: 1
current-plmn: 666/01
current-tac: 1
last-tai: PLMN[666/01] TAC[1]
stored-suci: no-identity
stored-guti:
 plmn: 666/01
 amf-region-id: 0x02
 amf-set-id: 1
 amf-pointer: 0
 tmsi: 0xca0061a5
has-emergency: false
```

Figure 22. The connected status for UE1.

Next, the connectivity between UE1 and the DN via PING program has been confirmed (Figure 23).

```
vm5G-UERANSIM:~/UERANSIM/build# ping -I uesimtun0 kt.agh.edu.pl
PING kt.agh.edu.pl (149.156.114.51) from 10.45.0.2 uesimtun0: 56(84) bytes of data.
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=1 ttl=61 time=21.3 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=2 ttl=61 time=28.2 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=3 ttl=61 time=22.1 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=4 ttl=61 time=23.1 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=5 ttl=61 time=27.5 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=6 ttl=61 time=25.7 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=7 ttl=61 time=28.2 ms
```

Figure 23. PING to external network from UE1.

```
[gtp] DEBUG: SEND GTP-U[255] to Peer[192.168.58.130] : TEID[0x1] (../lib/gtp/v2/p

[upf] DEBUG: [RECV] GPU-U Type [255] from [192.168.58.130] : TEID[0x2] (../src/up

[upf] DEBUG:    QFI [0x1] (../src/upf/gtp-path.c:299)
[pfcp] DEBUG: PROTO:1 SRC:0a2d0002 959c7233 0800e75e 829700c0 (../lib/pfcp/rule-m

[pfcp] DEBUG: HLEN:20  DST:959c7233 0800e75e 829700c0 40d36763 (../lib/pfcp/rule-

[pfcp] DEBUG: PROTO:58 SRC:0-0 DST:0-0 (../lib/pfcp/rule-match.c:172)
[pfcp] DEBUG: SRC:00000000 00000000 00000000 00000000/00000000 00000000 00000000
p/rule-match.c:178)
[pfcp] DEBUG: DST:ff020000 00000000 00000000 00000002/ffffffff ffffffff ffffffff
p/rule-match.c:187)
[upf] DEBUG: PAA IPv4:10.45.0.2 (../src/upf/rule-match.c:61)
```

Figure 24. GTP-U data related to ICMP Ping to kt.agh.edu.pl, issued by UE1.

In the next step, the UE1 PDU session has been released using UERANSIM CLI for UE1. The 5GCN responded correctly and a result of PDU session release is visible in AMF logs, as shown on Figure 25.

```
11/06 15:52:49.357: [smf] DEBUG: smf_state_operational(): OGS_EVENT_NAME_SBI_SERVER (../src/smf/smf-sm.c:83)
11/06 15:52:49.357: [smf] DEBUG: smf_gsm_state_wait_5gc_n1_n2_release(): OGS_EVENT_NAME_SBI_SERVER (../src/smf/gsm-sm.c:1412)
11/06 15:52:49.357: [smf] DEBUG:     TAI[PLMN_ID:66f610,TAC:1] (../src/smf/nsmf-handler.c:265)
11/06 15:52:49.357: [smf] DEBUG:     NR_CGI[PLMN_ID:66f610,CELL_ID:0x10] (../src/smf/nsmf-handler.c:267)
11/06 15:52:49.357: [smf] DEBUG: smf_state_operational(): SMF_EVT_5GSM_MESSAGE (../src/smf/smf-sm.c:83)
11/06 15:52:49.358: [smf] DEBUG: smf_gsm_state_wait_5gc_n1_n2_release(): SMF_EVT_5GSM_MESSAGE (../src/smf/gsm-sm.c:1412)
11/06 15:52:49.358: [sbi] DEBUG: STATUS [204] (../lib/sbi/nghttp2-server.c:357)
11/06 15:52:49.358: [sbi] DEBUG: STREAM closed [5] (../lib/sbi/nghttp2-server.c:852)
11/06 15:52:49.358: [sbi] DEBUG: [POST] http://127.0.0.5:7777/namf-callback/v1/imsi-666010000000001/sm-context-status/1 (../lib/sbi/client.c:612)
11/06 15:52:49.358: [sbi] DEBUG: SENDING...[54] (../lib/sbi/client.c:396)
11/06 15:52:49.359: [sbi] DEBUG: {
       "statusInfo":   {
              "resourceStatus":       "RELEASED"
       }
} (../lib/sbi/client.c:398)
11/06 15:52:49.360: [smf] DEBUG: smf_gsm_state_wait_5gc_n1_n2_release(): EXIT (../src/smf/gsm-sm.c:1412)
11/06 15:52:49.360: [smf] DEBUG: smf_gsm_state_session_will_release(): ENTRY (../src/smf/gsm-sm.c:1581)
11/06 15:52:49.360: [smf] INFO: Removed Session: UE IMSI:[imsi-666010000000001] DNN:[internet:1] IPv4:[10.45.0.2] IPv6:[] (../src/smf/context.c:1596)
11/06 15:52:49.360: [smf] DEBUG: smf_gsm_state_session_will_release(): EXIT (../src/smf/gsm-sm.c:1581)
11/06 15:52:49.360: [sbi] DEBUG: ogs_sbi_client_remove() [127.0.0.5:7777] (../lib/sbi/client.c:145)
11/06 15:52:49.360: [sbi] DEBUG: [UNREF] 3 (../lib/sbi/client.c:150)
11/06 15:52:49.360: [sbi] DEBUG: ogs_sbi_nf_instance_remove() (../lib/sbi/context.c:875)
11/06 15:52:49.360: [sbi] DEBUG: [UNREF] 2 (../lib/sbi/context.c:878)
11/06 15:52:49.360: [sbi] DEBUG: ogs_sbi_nf_instance_remove() (../lib/sbi/context.c:875)
11/06 15:52:49.360: [sbi] DEBUG: [UNREF] 2 (../lib/sbi/context.c:878)
11/06 15:52:49.360: [sbi] DEBUG: ogs_sbi_nf_instance_remove() (../lib/sbi/context.c:875)
11/06 15:52:49.361: [sbi] DEBUG: [UNREF] 2 (../lib/sbi/context.c:878)
11/06 15:52:49.361: [sbi] DEBUG: ogs_sbi_nf_instance_remove() (../lib/sbi/context.c:875)
11/06 15:52:49.361: [sbi] DEBUG: [UNREF] 2 (../lib/sbi/context.c:878)
11/06 15:52:49.361: [smf] INFO: [Removed] Number of SMF-Sessions is now 0 (../src/smf/context.c:2982)
11/06 15:52:49.361: [smf] INFO: [Removed] Number of SMF-UEs is now 0 (../src/smf/context.c:957)
```

Figure 25. PDU session release reported by SMF1 to AMF.

As specified in the resolution (see Section 6.3), both SMF2 and UE2 executables have been called on respective VMs. Simultaneously, SMF1 process on 5GCN CP platform has been stopped to trigger NF deregistration procedure (Figure 18).

The UE2 request for a PDU service has been correctly handled. Since AMF session cache had been cleared of the SMF1 entry upon being notified of its deregistration by the NRF, the internal SMF selection process (Listing 20) has returned the only one SMF present in the network: SMF2.

The workaround (described in 6.3) allowed for a positive PDU Session establishment for UE2 in slice [SST:2, SD:0x000002]. Listing 22 presents successful PDU session creation in the 5G-AN. The gNB forwards a resource reservation for UE2 PDU session to the AMF. The AMF confirms that the data session is available to SMF via the [*/nsmf-pdusession/v1/sm-contexts/1/modify*] resource.

Listing 22. N2 message presenting an accepted PDU session request and resource reservation in 5G-AN for UE2.

```
11/06 16:20:07.405: [amf] DEBUG: PDUSessionResourceSetupRequest(Session)
(../src/amf/ngap-build.c:1329)
11/06 16:20:07.405: [amf] DEBUG:     RAN_UE_NGAP_ID[3] AMF_UE_NGAP_ID[3]
(../src/amf/ngap-build.c:1374)
11/06 16:20:07.405: [amf] DEBUG:     IP[192.168.58.130] RAN_ID[1]
(../src/amf/ngap-path.c:70)
11/06 16:20:07.405: [sbi] DEBUG: STATUS [200] (../lib/sbi/nghttp2-server.c:357)
11/06 16:20:07.405: [sbi] DEBUG: SENDING...: 40 (../lib/sbi/nghttp2-server.c:365)
11/06 16:20:07.405: [sbi] DEBUG: {
    "cause":    "N1_N2_TRANSFER_INITIATED"}
11/06 16:20:07.408: [amf] DEBUG: amf_state_operational(): AMF_EVENT_NGAP_MESSAGE
(../src/amf/amf-sm.c:78)
11/06 16:20:07.408: [amf] DEBUG: ngap_state_operational(): AMF_EVENT_NGAP_MESSAGE
(../src/amf/ngap-sm.c:53)
```

```
11/06 16:20:07.408: [amf] DEBUG: PDUSessionResourceSetupResponse
(../src/amf/ngap-handler.c:1515)
11/06 16:20:07.408: [amf] DEBUG:     IP[192.168.58.130] RAN_ID[1]
(../src/amf/ngap-handler.c:1540)
11/06 16:20:07.408: [amf] DEBUG:     RAN_UE_NGAP_ID[3] AMF_UE_NGAP_ID[3]
(../src/amf/ngap-handler.c:1572)
11/06 16:20:07.408: [sbi] DEBUG: [POST]
http://127.0.0.24:7777/nsmf-pdusession/v1/sm-contexts/1/modify (../lib/sbi/client.c:678)
11/06 16:20:07.408: [sbi] DEBUG: SENDING...[291] (../lib/sbi/client.c:396)

Content-Type: application/json

{
    "n2SmInfo":    {
        "contentId":    "ngap-sm"
    },
    "n2SmInfoType":    "PDU_RES_SETUP_RSP"
}

11/06 16:20:07.413: [sbi] DEBUG: [204:POST]
http://127.0.0.24:7777/nsmf-pdusession/v1/sm-contexts/1/modify (../lib/sbi/client.c:562)
```

A data service has been positively provided, which has been verified on the UE2 UERANSIM console - Figure 26. As shown, the UE2 has been assigned an IP address from SMF2/UPF2 pool for the requested DN (IMS), equaling 10.46.0.2/16.

```
[2022-11-06 16:20:07.290] [nas] [info] Initial Registration is successful
[2022-11-06 16:20:07.290] [nas] [debug] Sending PDU Session Establishment Request
[2022-11-06 16:20:07.291] [nas] [debug] UAC access attempt is allowed for identity[0], category[MO_si
g]
[2022-11-06 16:20:07.495] [nas] [debug] Configuration Update Command received
[2022-11-06 16:20:07.540] [nas] [debug] PDU Session Establishment Accept received
[2022-11-06 16:20:07.541] [nas] [info] PDU Session establishment is successful PSI[1]
[2022-11-06 16:20:07.562] [app] [info] Connection setup for PDU session[1] is successful, TUN interfa
ce[uesimtun0, 10.46.0.2] is up.
```

Figure 26. An overview of 5G-AN data related to UE2 PDU service establishment for IP data.

Additionally, a remote site (kt.agh.edu.pl) is accessible from UE2 via dedicated tunnel interface (*uesimtun0*) with Ping program - Figure 27.

```
vm5G-UERANSIM:~/UERANSIM/build# ping -I uesimtun0 kt.agh.edu.pl
PING kt.agh.edu.pl (149.156.114.51) from 10.46.0.2 uesimtun0: 56(84) bytes of data.
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=1 ttl=61 time=32.0 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=2 ttl=61 time=29.3 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=3 ttl=61 time=22.9 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=4 ttl=61 time=22.1 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=5 ttl=61 time=28.1 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=6 ttl=61 time=27.7 ms
64 bytes from mimas.tele.agh.edu.pl (149.156.114.51): icmp_seq=7 ttl=61 time=27.9 ms
```

Figure 27. UE2 access to a remote network through 5G user plane.

Having surpassed another step in the resolution scenario (Section 6.3), the SMF1 instance has been recalled, which availed for reestablishing UE1 PDU session. The reestablishment is visible as the AMF is notified of any SMF that registers to the NRF. While UPF2 was constantly tunneling UE2 ICMP data to the internet, UE1 successfully registered to the 5GN (Figure 28) and established data session in slice [SST: 1, SD: 0x000001].

In Figure 28, UE1 sends a NAS request to establish a PDU session. The AMF then sends an HTTP/2 POST request to SMF1 via the [*nsmf-pdusessions/v1/sm-contexts*] service to anchor the session.

Figure 28. AMF logs regarding a SMF1 profile update (IP address of 127.0.0.4), together with UL NAS message containing UE1 request to establish data service.

After reestablishing the PDU session, UE1 is able to regain access to the internet, indicating that the resolution described in Section 6.3 was successful in addressing the SMF issue in the deployed 5G network.

7. A study on extedability of the selected CN implementation

This chapter discusses the possibility of adding a 3GPP-defined module to the chosen 5G Core Network implementation. Section 7.1 provides an analysis of selected services provided by Open5GS, while Section 7.2 describes the attempts to access those services utilizing SBI interfaces. Conclusively, Section 7.3 presents the final result of an implemented NF.

7.1. Overview of available NF

The Open5GS implementation of the 5G Core Network (5GCN) includes a sufficient number of NFs to meet the specification of Release 16 [30]. However, complex scenarios (e.g. network data analysis or exposing network events to third-party applications) may require the use of additional NFs [28].

Figure 29 presents the fundamental NFs of the 5G networks architecture. Yet, Open5GS does not provide all of them. The Network Exposure Function, Application Function (AF), and Network Slice Specific Authentication and Authorization Function (NSSAAF) are not implemented.

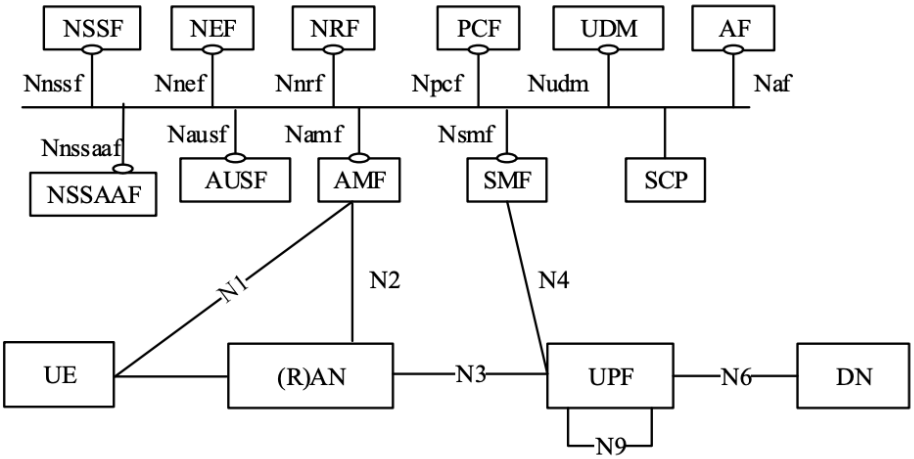

Figure 29. A SBI-based representation of NFs relations in 5G systems.

Source: 3GPP Procedures for 5G Systems [28]

Apart from these three NFs, the 5GCN defined in [30] also includes a module responsible for data analysis, NWDAF, which is not implemented as well.

Due to the lack of the mentioned NFs, it was impossible to perform some procedures defined for 5G systems. Thus, supplementing the Open5GS with one of the missing network functions would be beneficial for the community and present a visible contribution to the field of open source 5G implementations.

In order to create a functional module, it is necessary to have a strong foundation of essential services. After evaluating the dependencies between existing and planned NFs, it was determined that the NEF would be the best target for implementation because many of the planned NFs rely on its services. The remaining unimplemented network functions were excluded as they either require NEF for its operation, or would not provide useful services to the rest of Open5GS NFs. Below, the reasoning for rejecting other unimplemented NFs is presented.

The Open5GS implementation of the AMF does not support slice authorization, so the NSSAAF cannot be chosen for this project.

The NWDAF relies heavily on data related to the network operations. Since the module itself cannot obtain the information necessary to conduct network state summary, it requires a NEF southbound API to access the required data [24, 9], i.e. the *Nnef_EventExposure* service.

Application Functions can be used to influence 5G network behaviour. However, the 5GCN CP functions and services cannot be directly accessed by the AFs due to security policies. Instead, access is provided through the northbound API exposure of the NEF, which helps to prevent unauthorized access [10]. Therefore, implementing an AF would require NEF as well.

Network Exposure Function, as defined in 3GPP 23.501 [30], provides external exposure of Control Plane NFs capabilities to third party applications. These capabilities can be related to Monitoring, Provisioning or Policy/Charging [24].

Policy/Charging capability is related to enforcing QoS policies and enabling charging for a UE in 5G network. This capability involves a direct interaction of NEF with a PCF in order to forward policy/charging request from an external application.

Monitoring capability concerns exposing events regarding UEs mobility, reachability or loss of connectivity. It allows identifying en emerging event, together with a NF that produces it. NEF enables an eligible external application to access such events (Figure 30).

Provisioning capability allows an authorized external application to provide the 5G NFs with data related to a given UE. Such data migh be then used to influence the UE behaviour in the network. For instance, NEF might change Session Management data of an UE upon request from an external application.

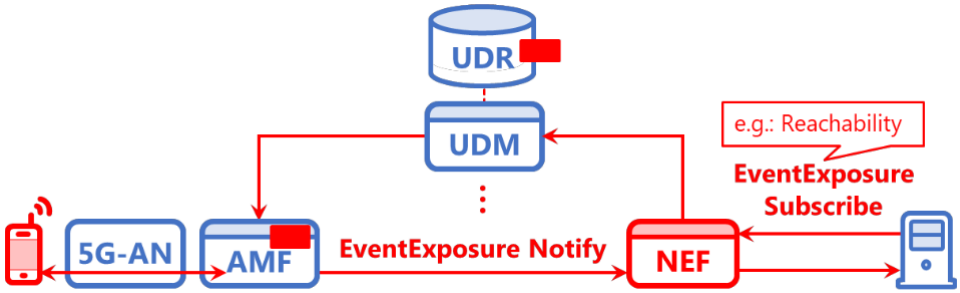

Figure 30. A representation of NEF EventExposure to a 3rd party application.

Source: Sploty 6.0 5G Systems Overview [24]

In this thesis, an approach to deliver a monitoring capability has been attempted.

7.2. Verification of the accessibility of required NF services

The first steps into delivering a NEF application are related the EventExposure capability of Open5GS. As noted in 3GPP TS 29.508 [12], NEF relays SMF events to 3rd party applications via the *Nsmf_EventExposure service*.

A code analysis of the Open5GS SMF implementation showed that this service unavailable. In addition, when accessing the *Namf_EventExposure* service with a proper POST request, an *Invalid Resource* code is returned. The above steps were repeated for the AMF-related *Namf_EventExposure* service to check if Mobility information could be exposed. Unfortunately, the outcome was the same.

Since it is not possible to access the necessary resources to implement the NEF function, a different approach has been taken. The desired events are related to UE mobility and session management, so an alternative solution is to provide on-demand access to this information instead of using the Subscribe-Notify model. From the perspective of a mobile network operator, such a module would be very useful.

Provided information should include the number and type of established PDU sessions, the anchoring AMF per UE, policy enforcement, QoS statistics, and overall slice capacity. Data related to the active PDU sessions and QoS policies is provided by the PCF and can be extracted through the

Npcf_SMPolicyControl service API. The data regarding the overall PDU sessions can be accessed through UDR. *SessionManagementSubscriptionData* is the session management data storage. The mobility information of the UE can also be accessed via UDR resource: *AccessAndMobilitySubscriptionData*.

The accessibility of the service has been tested on a deployed 5GCN as described in Section 6.1.

7.2.1. PCF SMPolicyControl accessibility verification

The SMPolicyControl service should be accessible for extraction of an individual Session Management (SM) Policy that is provided by the PCF. The data of a specific SM Policy is supposed to be accessible by SMF at the *[apiRoot/npcf-smpolicycontrol/v1/sm-policies/smPolicyId]* resource.

Before trying to extract any SMPolicy Data from the PCF, the following steps were taken:

1. Assuring that the 5GCN is up and running.
2. Issuing UE triggered PDU Session establishment.
3. Ensuring that the PDU Session has been anchored to a proper DN.
4. Intercepting the PDU Session ID in PCF logs.
5. Executing a standardized HTTP/2 GET request to the PCF API responsible for SMPolicyControl with the intercepted Session ID.
    - curl -X GET –http2-prior-knowledge 'http://192.168.58.110:16003/npcf-smpolicycontrol/v1/sm-policies/1'
6. Results analysis.

The curl HTTP/2 request in Step 5 returned an unexpected error code: '403 Invalid HTTP method', which is not mentioned in the PCF specification. However, the error is not relevant as the GET method is supposed to be used to access a selected SM Policy.

Further investigation revealed the PCF code (Listing 23) responsible for handling the Npcf interface. The internal switch case for the *Npcf_SMPolicyControl* resource returns an error for any request that does not comply with the syntax [*apiRoot/npcf-smpolicycontrol/v1/sm-policies/smPolicyId/delete*].

apiRoot/npcf-smpolicycontrol/v1/sm-policies/smPolicyId/delete

The PCF allows only for deletion of a given SM policy. Any attempt to extract the SM policy data with GET method causes the PCF SBI server to return an error.

Listing 23. Open5GS PCF source code handling Npcf SBI interface.

```
SWITCH(message->h.resource.component[2])
CASE(OGS_SBI_RESOURCE_NAME_DELETE)
    handled = pcf_npcf_smpolicycontrol_handle_delete(
            sess, stream, message);
    if (!handled) {
        ogs_error("[%s:%d] "
            "pcf_npcf_smpolicycontrol_handle_delete() failed",
            pcf_ue->supi, sess->psi);
        OGS_FSM_TRAN(s, pcf_sm_state_exception);
    }
    break;

DEFAULT
    ogs_error("[%s:%d] Invalid HTTP URI [%s]",
            pcf_ue->supi, sess->psi, message->h.uri);
    ogs_assert(true ==
        ogs_sbi_server_send_error(stream,
            OGS_SBI_HTTP_STATUS_FORBIDDEN, message,
            "Invalid HTTP method", message->h.uri));
END
```

7.2.2. UDR Nudr SBI resource access

As defined in UDR specification [5], the Access and Mobility Subscription Data, along with the Session Management Subscription Data, should be accessible at the designated location:

- AccessAndMobilitySubscriptionData → */subscription-data/ueId/servingPlmnId/provisioned-data/am-data*
- SessionManagementSubscriptionData → */subscription-data/ueId/servingPlmnId/provisioned-data/sm-data*

The verification process of the following resource accessibility is a mirrored version of the one mentioned in Section 7.2.1. The curl request is designed to get the Subscription Data of UE1 and includes its IMSI in the URL.

The curl HTTP/2 GET requests to the respective resource are constructed as follows:

- AccessAndMobilitySubscriptionData:

  curl -X GET –http2-prior-knowledge 'http://192.168.58.110:16005/nudr-dr/v1/subscription-data/imsi-666010000000001/66601/provisioned-data/am-data

- SessionManagementSubscriptionData:

  curl -X GET –http2-prior-knowledge 'http://192.168.58.110:16005/nudr-dr/v1/subscription-data/imsi-666010000000001/66601/provisioned-data/sm-data

The attempt to access the AM data was successful using the given query. As shown in Fig. 31, the response returned was in the form of JSON file and contained the subscribed AMBR data and NSSAI information.

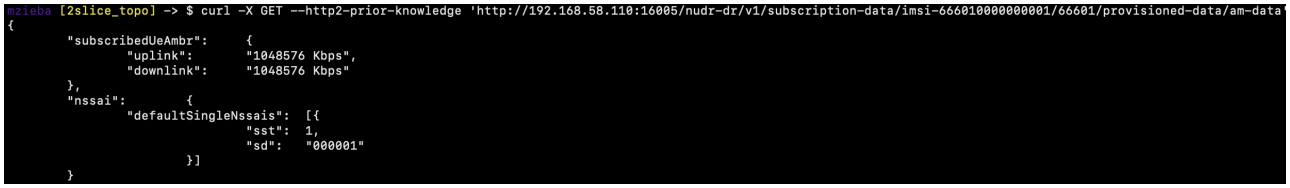

Figure 31. A successful curl query to a Nudr SBI interface of UDR.

However, the SM query failed to return SM data, with the error code of '400 [imsi-666010000000001] NO_SNSSAI' (Figure 32). According to the 3GPP standard for Nudr SBI [5], the S_NSSAI for the PDU Session is a mandatory query parameter in the HTTP/2 URL for a given SM resource.

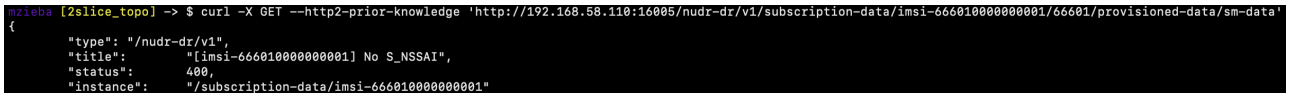

Figure 32. A failed curl query to a Nudr SBI interface of UDR.

The modified part included an URL encoded single-nssai parameter of the pair of [SST: 1, SD: '000001'] for UE1 slice:

```
single-nssai={
"sst":    {sst},
"sd":    "{sd}"
}
```

After specifying the slice to retrieve the SM subscribed information, the curl request returned a JSON containing the desired Session Management data, as shown in Figure 33.

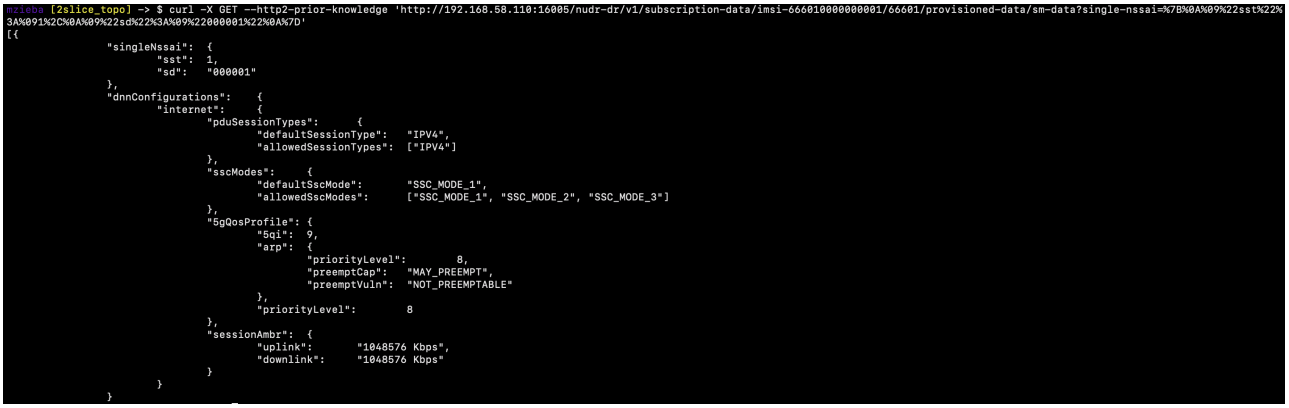

Figure 33. A successful curl retrieval of SM data for UE1 default S-NSSAI.

7.3. Implementation of a quasi-NEF

The implementation of the NEF has been hindered due to the lack of resources that are readily accessible from NFs, such as the PCF (as mentioned in Section 7.2.1). The initial plan to replace event exposure with the scraping of real-time and on-demand information from the NFs involved in Session Management and Policing has been abandoned. The current release of Open5GS does not allow for access to the current state of the network. However, it is possible to access UE-specific data, such as the Subscriber data stored in the MongodB, through the UDR (as verified in Section 7.2.2)

Due to the misaligned implementation of the PCF, it is currently impossible to retrieve on-demand PDU session policies. However, aggregating raw data from Subscription Profiles may provide insight into the state of the 5GCN.

With that assumption in mind, a conceptual quasi-NEF has been implemented. The NF is written in Python, a general purpose scripting language that proved useful during the initial prototyping phase. Additionally, the *pycurl* module accurately replaced the curl CLI program, assuring that the SBI services examined in the previous sections would behave in the same manner upon accessing them via Python scripts.

The quasi-NEF's goal is to aggregate 5G related data that would be useful for a mobile network operator or a private 5G slice tenant. The implemented function is expected to:

- Summarize slice data,
- List all preemption vulnerable UEs,
- List all preemption capable UEs.

Summarizing slice-specific data can be useful when predicting the resources that need to be deployed to ensure uninterrupted service provisioning. For example, aggregated data that shows the total aggregated maximal bitrate for a given DN may suggest increasing the number of UPF instances in the User Plane to reduce pressure on the existing NFs. If the 5GCN CP needs to be scaled, information about the SSCs supported by UEs for a given DN can help to ensure PDU Session continuity as the

generated traffic is offloaded onto newly deployed UPFs [24]. The total bitrate per DN may also be a factor to consider when adjusting virtualized resources to the network state.

Such slice summary consists of the following parts:

- ALL_SUPPORTING_IMSIS, which is a list of UEs that support a given S-NSSAI.
- PER_DNN_AGGREGATED_BITRATE, which denotes the total Uplink/Downlink aggregated bitrate for a DN in a given slice.
- PER_DNN_COMMON_SUPPORTED_SSC, a parameter that outlines all Session and Service Continuity (SSC) [24] modes present in a DN.
- PER_DNN_DEFAULT_REQUIRED_SSC, a parameter which names the default SSC mode that is common for the UEs within a slice for a DN.
- TOTAL_AMBR_UL, a parameter denoting a total Aggregated Maximal Bitrate for Uplink in the slice.
- TOTAL_AMBR_DL, a parameter denoting a total Aggregated Maximal Bitrate for Downlink in the slice.
- TOTAL_IMSIS, which is a number of UEs that support a given slice.

If the 5GCN infrastructure is at risk of being overloaded, additional network resources must be deployed. The pressure on the physical hardware should be reduced to ensure that 5G critical services can continue to be delivered. However, physical constraints may prevent from increasing the number of NFs on a given platform. In such cases, the last resort may be to release a subset of non-critical services, such as low priority PDU sessions, in order to reduce the load on the network.

Listing 24 presents a summarization of Session Management data for a [SST:1 SD:000001] slice returned by the implemented quasi-NEF:

Listing 24. A slice summary for S-NSSAI: [SST:1, SD:0x000001].

```
{'S_NSSAI[SST:1 SD:000001]': {
    'ALL_SUPPORTING_IMSIS': ['666010000000001',
     '666011',
     '666012',
     '666013'],

    'PER_DNN_AGGREGATED_BITRATE': {'ims': {'TOTAL_AMBR_DL': '260096Kbps',
    'TOTAL_AMBR_UL': '260096Kbps'},
    'internet': {'TOTAL_AMBR_DL': '1506304Kbps',

    'TOTAL_AMBR_UL': '1506304Kbps'}},
```

```
'PER_DNN_COMMON_SUPPORTED_SSC': {'ims': ['SSC_MODE_3',
                                         'SSC_MODE_1',
                                         'SSC_MODE_2'],
                                 'internet': ['SSC_MODE_3',
                                              'SSC_MODE_1',
                                              'SSC_MODE_2']},
                'PER_DNN_DEFAULT_REQUIRED_SSC': {'ims': {'SSC_MODE_1'},
                'internet': {'SSC_MODE_1'}},
                'TOTAL_AMBR_DL': '1766400Kbps',
                'TOTAL_AMBR_UL': '1766400Kbps',
                'TOTAL_IMSIS': 4}}
```

7.3.1. Code overview

The main class that acts as an intermediary between UDR and the rest of quasi-NEF components is *PycurlClient*. It is a Singleton implementation of an overloaded *Curl* class provided by the *pycurl* module. Although the class may seem as a simple replacement of curl CLI utlitity, it provides additional features:

- Buffering the HTTP responses.
- Altering the pycurl-defined parameters.
- Custom exception handling.

Listings 25 and 26 highlight the core part of PycurlClient, which interacts with UDR and extracts data.

Listing 25. A PycurlClient method to perform HTTP/2 request with custom Exception handling.

```
@override
def perform(self):
    """Method to execute http/2 query towards NF SBI servers"""
    self._buffer = BytesIO()
    assert 'URL' in self._curl_options, f'url unset'
    self.setopt(pycurl.WRITEDATA, self._buffer)
    try:
        super().perform()
    except pycurl.error as err:
        _message = f'Unable to perform http/2 request to
        ↪ {self._curl_options.get("URL")}\n\tError: {err}'
        raise InvalidQueryException(message=_message, errors=err)
```

The overridden *perform* method allows for on-demand buffer initialization, as Pycurl does not have a built-in data storing mechanism - Listing 25.

While CLI curl by default delivers the query result to the standard output, PyCurl requires defining a buffer to store the returned data. It has been implemented by assigning a BytesIO object that the PyCurl writes the query results to. The *setop* method (Listing 26) ensures that any entered configuration parameters (usually the URL) are of the correct type. It also stores the accepted parameters in order to facilitate easy access to the Pycurl configuration (e.g., for retrieving the type of NFs or the SBI resource).

Listing 26. An overridden Pycurl's *setop* method featuring option logging.

```
@override
def setopt(self, option, value):
    """Sets an option of Curl client"""

    assert option in self.__libcurl_acceptable, f'invalid option to setopt:
    {option}'
    try:
        super().setopt(option, value)
        self._curl_options.update({self.__libcurl_acceptable.get(option):
        value})
    except TypeError as err:
        print(f'invalid value: {value} to option: {option}; {err}')
    except pycurl.error as err:
        print(f'libcurl rejected option || value: {err}')
```

BaseClass uses both *setopt* and *perform* in its class *send_query* method. The BaseClass *send_query* method allows (Listing 27) for dynamic generation of HTTP/2 requests via a common handle (PycurlClient Singleton) and decoding the response into a JSON format. The method is inherited by remaining NFs specific agents: AmfClient, UdrClient and NrfClient.

Listing 27. The use-case of PycurlClient Singleton for HTTP/2 requests issuing.

```
class BaseNfClient:
    @classmethod
    def send_query(cls, http_proxy: PycurlClient, url) -> Union[dict, None]:
        http_proxy.setopt(pycurl.URL, url)
        try:
            http_proxy.send_get_event()
        except InvalidQueryException as e:
            print(e)
```

```
            return None

        _response = http_proxy.get_http_response()

        if not _response:
            print(f'didnt receive any response from sbi:
            {http_proxy.get_targeted_nf()}')
        elif isinstance(_response, str):
            try:
                _response = json.loads(_response)
            except json.decoder.JSONDecodeError:
                _message = f'Failed to compile response into json: {_response}'
                return None

        return _response
```

The BaseClass *send_query* method provides an acceptable format of data to undergo a feature extraction process. The JSON-encapsulated data is aggregated with the help of *map/filter/reduce* Python functions (Listing 28), and presented in both human and further-processing friendly format.

Listing 28. The slice data aggregation enabled by Pythons built-in data processing functions.

```
_aggregated_bitrate = self._aggregate_bandwidth(self._get_all_ambr_items(_data))
        _per_dnn_bitrate = self._get_per_dnn_ambr_items(_data)
        _per_dnn_aggregated_bitrate = list(
            map(lambda x: {x: self._aggregate_bandwidth(_per_dnn_bitrate[x])},
            _per_dnn_bitrate))
        _per_dnn_aggregated_bitrate =
        self._unpack_dicts(_per_dnn_aggregated_bitrate)

        _all_ssc_modes = self._get_all_ssc_modes(_data)

        _default_required_ssc_modes = list(map(lambda x: {x:
        set(_all_ssc_modes[x]['DEFAULT_SSCS'])}, _all_ssc_modes))
        _default_required_ssc_modes = dict(reduce((lambda x, y: {**x, **y}),
        _default_required_ssc_modes))

        _common_supported_ssc_modes = list(map(
```

```
            lambda x: {x:
            self._get_union_of_supported_ssc(_all_ssc_modes[x]['ALL_SSCS'])},
            _all_ssc_modes))
        _common_supported_ssc_modes = dict(reduce((lambda x, y: {**x, **y}),
        _common_supported_ssc_modes))
```

The quasi-NEF data aggregation methods are called by invoking a Singleton of *AfService* class, which mimics an Northbound API of an actual NEF [10]. Then a selection of methods can be called on the object. Figure 34 presents a successful request to provide the collection of preemption capable UEs for slice [SST: 2, SD:Ox000002].

```
In[3]: nb_api = AfService()
In[4]: nb_api.get_all_preemption_capable_ues(sst=1, sd='000001')
Out[4]:
{'PREEMPTIVE_UES': {'internet': {'666010000000001': {'preemptCap': 'MAY_PREEMPT',
    'arp': 8},
   '666012': {'preemptCap': 'MAY_PREEMPT', 'arp': 8}},
  'ims': {}}}
In[5]: nb_api.get_all_preemption_capable_ues(sst=1, sd='000002')
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/IPython/core/interactiveshell.
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "<ipython-input-5-4df0536bd5d2>", line 1, in <cell line: 1>
    nb_api.get_all_preemption_capable_ues(sst=1, sd='000002')
  File "/Users/mzieba/Dropbox/praca_inzynierska/5G_git/NEF/af_service.py", line 129, in get_all_preemption_capable_ues
    _data = self.get_aggregated_slice_data(sst=sst, sd=sd, imsis=imsis)
  File "/Users/mzieba/Dropbox/praca_inzynierska/5G_git/NEF/af_service.py", line 56, in get_aggregated_slice_data
    raise InvalidQueryException(message=_message, errors='')
nef_proxy_aux.InvalidQueryException: Unsupported slice: S_NSSAI [SST: 1, SD: 000002]
```

Figure 34. A successful query to UDR and aggregation of preemption-capable UE IMSIs, together with a custom exception after requesting a nonexistent resource.

The implemented quasi-NEF is capable of aggregating Session and Access management data that is extracted from UDR. This way of accessing data is not necessarily the case that would happen in an actual 5G core network, as the UDR is not supposed to be directly interfaced by a NEF. However, this was the only possible approach to implement a NF that would mimic a NEF functionality using the Open5GS software.

Such implementation is only an intermediary step in providing an actual NEF, as it could be extended with event exposure. An actual NEF would enable an external application to subscribe to a 5G network event (e.g. UE deregistering). Then the NEF would subscribe itself in the name of the external application to the NF that produces a given event. Further, if the NEF is notified of a subscribed event, it would forward it to the external application.

8. Conclusion

In conclusion, the deployment of 5G core networks in cloud environments offers significant potential benefits for operators, including improved scalability, flexibility, and cost-efficiency. However, the adoption of this approach also presents a number of challenges and limitations that must be carefully considered. The considerations and challenges that have been addressed in this thesis might successfully aid the deployment of 5G core networks into cloud environments.

The thesis showed a successful deployment of 5G core network based on Open5GS software on top of virtual machines. One of the encountered challenges involved resolving a deployment-specific issue regarding the way Open5GS handles slicing. The Open5GS caused problems related to slice selection for data session establishment. The thesis presented a resolution to the issue by utilizing the Network Function deregistering mechanism together with data session releasing. The description of an issue with all necessary data would be signaled to the Open5GS community.

Furthermore, research into the Open5GS showed what services and procedures can be executed using the Network Functions provided by the framework. These findings resulted in implementing a quasi-NEF client that exposes subscriber information to the 3rd party applications (external to the 5G core network). The implemented module provides data that could be beneficial for mobile carriers while managing the 5G network.

The further research might involve transferring the virtual machines used for Network Functions deployment to a public cloud environment. An extension to the second part of the thesis would involve enabling the Open5GS NFs to support the services required for a proper implementation of a Network Exposure Function.

References

[1] ITU-R. "IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond". In: 2015.

[2] Mayur M Patil et al. "A qualitative analysis of the performance of MongoDB vs MySQL database based on insertion and retriewal operations using a web/android application to explore load balancing — Sharding in MongoDB and its advantages". In: *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. 2017, pp. 325–330.

[3] Martin Sauter. *From GSM to LTE-Advanced Pro and 5G - An Introduction to Mobile Networks and Mobile Broadband, Third Edition, WirelessMoves Cologne*. John Wiley & Sons Ltd, 2017.

[4] Duc-Hung Luong et al. "Cloudification and Autoscaling Orchestration for Container-Based Mobile Networks toward 5G: Experimentation, Challenges and Perspectives". In: *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. 2018, pp. 1–7.

[5] 3rd Generation Partnership Project. *5G; 5G System; Usage of the Unified Data Repository services for Subscription Data; Stage 3*. 2018.

[6] 3rd Generation Partnership Project. *5G; System Architecture for the 5G System*. 2018.

[7] 3rd Generation Partnership Project. *Study on Enhancements to the Service-Based Architecture*. 2018.

[8] Ying Zhang. *Network Function Virtualization Concepts and Applicability in 5G Networks*. Wiley, 2018.

[9] 3rd Generation Partnership Project. *5G; 5G System (5GS); Network exposure function southbound services; Stage 3*. 2019.

[10] 3rd Generation Partnership Project. *5G; 5G System; Network Exposure Function Northbound APIs; Stage 3*. 2019.

[11] 3rd Generation Partnership Project. *5G; 5G System; Network function repository services; Stage 3*. 2019.

[12] 3rd Generation Partnership Project. *5G; 5G System; Session Management Event Exposure Service; Stage 3*. 2019.

[13] Dejana T. Vojnak et al. "Performance Comparison of the type-2 hypervisor VirtualBox and VMWare Workstation". In: *2019 27th Telecommunications Forum (TELFOR)*. 2019, pp. 1–4.

[14] AWS. *5G Network Evolution with AWS Design scalable, secure, reliable, and cost-efficient cloud-native core and edge network on AWS*. 2020. https://d1.awsstatic.com/whitepapers/5g-network-evolution-with-aws.pdf (visited on 11/26/2022).

[15] Leonardo Bonati et al. "Open, Programmable, and Virtualized 5G Networks: State-of-the-Art and the Road Ahead". In: *Computer Networks* 182 (2020), p. 107516. https://www.sciencedirect.com/science/article/pii/S1389128620311786.

[16] Wan Lei et al. *5G System Design. An End to End Perspective*. Springer, 2020.

[17] Young Jung Christopher Adigun. *Open source mobile core network implementation on Amazon Elastic Kubernetes Service*. 2021. https://aws.amazon.com/blogs/opensource/open-source-mobile-core-network-implementation-on-amazon-elastic-kubernetes-service/ (visited on 11/26/2022).

[18] Francisco Joaquim De Souza Neto et al. "Analysis for Comparison of Framework for 5G Core Implementation". In: *2021 International Conference on Information Science and Communications Technologies (ICISCT)*. 2021, pp. 1–5.

[19] Kuan-Lin Lee, Chung-Nan Lee, and Ming-Feng Lee. "Realizing 5G Network Slicing Provisioning with Open Source Software". In: *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. 2021, pp. 1923–1930.

[20] Microsoft Azure. *Create a virtual network using the Azure portal*. 2022. https://learn.microsoft.com/en-us/azure/virtual-network/quick-create-portal (visited on 12/13/2022).

[21] Sergio Barrachina-Muñoz, Miquel Payaró, and Josep Mangues-Bafalluy. "Cloud-native 5G experimental platform with over-the-air transmissions and end-to-end monitoring". In: *2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*. 2022, pp. 692–697.

[22] free5GC. *free5GC description*. 2022. https://www.free5gc.org/.

[23] Sukchan Lee. *Open5GS Documentation*. 2022. https://open5gs.org/open5gs/docs/ (visited on 11/26/2022).

[24] © Sploty Sp. z o.o. *5G System Overview*. Sploty Sp. z o.o., 2022.

[25] RedHat. *VNF and CNF, what's the difference?* 2022. https://www.redhat.com/en/topics/cloud-native-apps/vnf-and-cnf-whats-the-difference (visited on 12/13/2022).

[26] Oana-Mihaela Ungureanu and Călin Vlădeanu. "Leveraging the cloud-native approach for the design of 5G NextGen Core Functions". In: *2022 14th International Conference on Communications (COMM)*. 2022, pp. 1–7.

[27] VirtualBox. *6.1. Virtual Networking Hardware*. 2022. https://www.virtualbox.org/manual/ch06.html (visited on 12/13/2022).

[28] 3GPP 2018. *5G; Procedures for the 5G System*. 3GPP TS 23.502 version 15.2.0 Release 15.

[29] 3GPP 2019. *Binding Support Management Service*. 3GPP TS 29.521 version 15.3.0 Release 15.

[30] 3GPP 2020. *System architecture for the 5G System (5GS)*. TS 23.501 version 16.6.0.

[31] Massimo Condolucia and Toktam Mahmood. "Softwarization and virtualization in 5G mobile networks: Benefits, trends and challenges". In.

[32] Docker. *Use containers to Build, Share and Run your applicatio*. https://www.docker.com/resources/what-container/ (visited on 12/13/2022).

[33] ALİ GÜNGÖR. *UERANSIM*. https://github.com/aligungr/UERANSIM.

[34] *RESTful APIs of main Network Functions in the 3GPP 5G Core Network*. https://github.com/jdegre/5GC_APIs/tree/Rel-16 (visited on 12/13/2022).

[35] s5uishida. *Open5GS 5GC UERANSIM UE / RAN Sample Configuration - Select UPF based on S-NSSAI*. https://github.com/s5uishida/open5gs_5gc_ueransim_snssai_upf_sample_config (visited on 12/13/2022).

[36] 3GPP 2021 Technical Specification Group Services and System Aspects. *Security architecture and procedures for 5G system*. TS 33.501 V16.6.0 (2021-03).

[37] VMWare. *Bare-metal hypervisors*. https://www.vmware.com/topics/glossary/content/bare-metal-hypervisor.html (visited on 12/13/2022).

[38] VMWare. *What are Containers?* https://www.vmware.com/topics/glossary/content/containers.html (visited on 12/13/2022).

[39] VMWare. *What is a hypervisor?* https://www.vmware.com/topics/glossary/content/hypervisor.html (visited on 12/13/2022).