António Manuel Martins Borges

5GLAN: Redes de Área Local baseadas em tecnologias 5G para Mission Critical Services

Network Exposure Function as Microservices

Dissertation in the context of the Master's in Informatics Engineering, Specialization in CSI,
Oriented by the professor Tiago Cruz and presented to Professors Fernando Boavida and Catarina Silva
Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2021

Acknowledgments

I would like to say thank you to my parents for all the support throughout my life and in specific regarding the support during the production of this document.

Thank you, Professor Luís Cordeiro for the reality checks and for the suggestions and comments to improve my work constantly.

A big thanks to Professor Tiago Cruz for being always available to clarify my doubts and for your helpful suggestions.

To André Gomes, a big thank you for the technical discussions and for the helpful suggestions and advice.

Finally, I would like to thank to João Henriques for his valuable guidance on the production of this document.

Abstract

In the last few years, the new Fifth Generation (5G) of mobile networks has been introduced and has brought the potential of extend the cloud capabilities to the network. This means that the virtualized environments will become part of the network. Beyond that, 5G also brings a new architecture and one of the key changes is that the 5G Core (5GC) network functions (NFs) become loosely coupled. However, as the network evolves and new features are added, the loosely coupled NFs grow in complexity, impacting their performance. A microservice architecture is a better choice for implementing the 5GC NFs.

Therefore, this works presents the standardized 3rd Generation Partnership Project (3GPP) 5G architecture, with the focus on the Network Exposure Function (NEF) and its decomposition on micro services and the tools that will enable the management of those micro services. With the standards and the tools, the work described in this document aims to carry out the NEF decomposition into micro services and compare its performance with a monolithic NEF implementation.

Keywords: 5G, 5G Core, Network Functions, Network Exposure Function, Microservices.

Glossary

1G -First Generation of Mobile Telecommunications

2G – Second Generation of Mobile Telecommunications

3G – Third Generation of Mobile Telecommunications

3GPP - 3rd Generation Partnership Project

4G – Fourth Generation of Mobile Telecommunications

4K - 4000-pixel Screen Resolution

5G – Fifth Generation of Mobile Telecommunications

5G-VN – 5G Virtual Network

5GC – Fifth Generation of Mobile Telecommunications Core

5GLAN – 5G Local Area Network

5GLAN-VN – 5G Local Area Network Virtual Network

5GNR – 5G New Radio

5GPPP – 5th Generation of Mobile Telecommunications Infrastructure Public Private Partnership

5GS – 5G System

AF – Application Function

AMF – Access and mobility management function

API -Application Programming Interface

AUSF – Authentication Server function

BDTP – Background Data Transfer Policy

CLI – Command Line Interface

CP – Control Plane

CPU – Central Process Unit

DHCP – Dynamic Host Configuration Protocol

DNN – Data Network Name

DNS – Domain Name System

E2E NSO – End to End Network Service Orchestrator

EPC – Evolved Packet Core

eSBA – Enhanced Service Based Architecture

GMLC – Gateway Mobile Location Centre

HPLMN – Home Public Land Mobile Network

HTTP – Hypertext Transfer Protocol

ICN - Information -Centric network

LAN – Local Area Network

MAC – Media Access Control Address

MANO – Management and Orchestration

NaaS – Network as a Service

NAS – Non Access Stratum

NEF - Network Exposure Function

NF – Network Function

NFV – Network Function Virtualization

NFV MANO – Network Function Virtualization MANO

NGMN – Next Generation Mobile Networks

NIDD – Non-IP Data Delivery

NIDD – Non-IP Data Delivery

NPN – Non-Public Network

NRF – Network Function Repository

NRF – Network Repository Function

NSI – Network Slice Instance

NSSAI – Network Slice Selection Assistance Information

NSSF – Network Slice Selection Function

NSSI – Network slice subnet instance

NWDAF – Network Data Analytics Function

ONAP – Open Network Automation Platform

OS – Operative System

OSI layer – Open System Interconnection layer

OSM – Open Source Mano

PCF – Policy Control Function

PDU Session - Protocol Data Unit Session

PEI – Permanent Equipment Identifier

PFD – Packet Flow Description

PFDF – Packet Flow Description Function

PLMN – Public Land Mobile Network

PNI-NPN – Public Network Integrated Non-Public Network

QoS – Quality of Service

RAN – Radio Access Network

S-NSSAI – Single Network Slice Selection Assistant Information

SBA – Service Based Architecture

SBI – Service Based Interface

SCP – Service Communication Proxy

SCS/AS – Service Capable Server / Application Server

SDN – Software Defined Network

SFM – Service Function Monitor

SM Context – Session Management Context

SMEs - Small Medium-sized Enterprises

SMF – Session Management's Function

SNPN – Standalone Non-Public Network

SUCI – Subscription Concealed Identifier

SUPI – Subscription Parameter Identifier

TCP – Transmission Control Protocol

TSN – Time Sensitive Networks

UCMF – UE Radio Capability Management Function

UDM – Unified Data Management

UDR – Unified Data Repository

UE – User Equipment

UP – User Plane

UPF – User Plane Function

VA – Vertical Application

VAO – Vertical Application Orchestrator

VIM – Virtualized Infrastructure Manager

VM – Virtual Machine

VNF – Virtual Network Function

VPLMN – Visited Public Land Mobile Network

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 TOPIC AND CONTEXT

5G is one of the hot acronyms of the present time but is nothing more than the fifth generation of mobile telecommunication networks. 5G brings huge improvements in three areas: bandwidth, latency and density. The most well-known by the general public is the massive improvement of the bandwidth that derives from the use of the so-called millimetre electromagnetic waves (mmWave). The bandwidth improvement will enable the generalization of technologies that are now limited by current speeds, 4000-pixel (4K) screen resolutions will become generalized, video streaming will become the norm and other technologies will grow, cloud gaming or virtual reality for example. Another 5G improvement is the ultra-connectivity capabilities that will allow for greater density of devices connected by area unity. It will boost IoT enabling smart houses or smart cities. But the most important improvement resides on the reduction of communication latency, and it is probably the area were 5G will impact the most. Autonomous driving is the first area that comes to mind, but the low latency will enable a revolution to industry and can even impact medicine.

These three major improvements provided by 5G are leading to Cloudification. Cloudification is the process of extending cloud platforms, technologies, and virtualization capabilities to the network. The network becomes more agile, flexible and scalable. 5G will lead to the network softwarization. A network based on software opens doors for new key technologies like software-defined networks, network slicing, network virtualization and intelligent orchestration.

Network virtualization is at the present time a very well standardized field. Third Generation Partnership Project (3GPP) releases provide standards that have detailed the virtualized network functions and its Application Programming Interfaces (APIs). It provides a standard from where researchers can build and implement features to fulfil the 5G potential to the telco world. Intelligent orchestration is one of those features and allows the network to orchestrate herself adjusting dynamically to the user, device or use case for optimal performance.

Taking in consideration the 5G potential, not only for the telco providers, but also for different size companies, an international consortium was created joining companies and research institutions from different countries in Europe to develop the FUDGE-5G [1] project. FUDGE-5G stands for "FUlly DisinteGrated private nEtworks for 5G verticals". One of the project partners is OneSource and it is on the context of the participation of OneSource in the FUDGE-5G project that the work described in this document is being performed and developed. The work is part of the network function disintegration, a goal of FUDGE-5G, in particular the decomposition of the Network Exposure Function into micro services.

1.2 Host Entity

OneSource [2] is high-tech Small and medium-sized enterprise (SME) based in Coimbra and specialized in 4 areas of activity:

- Provision of consulting and auditing services in communication networks, computer infrastructures and information systems.
- Design, implementation and administration of computer networks and systems in an outsourcing regime, being currently responsible for the administration of networks and systems that serve, in total, more than 12,000 users.
- Applied research and technological development services in the areas of information and telecommunications technologies.
- Design, development and maintenance of information systems, including intranets, corporate websites and integration of information systems.

The participation of OneSource in FUDGE-5G is due to its large expertise integrating its Situational Awareness platform, Mobitrust [3], with other projects and platforms. Integration and development of its features were supported by projects Mobitrust (Eureka/CATRENE), Mobilizador 5G (national project) and 5GINFIRE (H2020).

1.3 Objectives

There are a lot of research possibilities on the 5G world. Those possibilities encompass the applications for the 5G technology, wave spectrum challenges, battery life and storage for 5G equipment’s, security research, small cell research, 5G core implementations and much more with a lot of different research groups addressing challenges in all of those possibilities.

One of the core tasks of FUDGE-5G is to ensure that appropriate network function disintegration is carried out both for the 5GC control and user plane. In the end, the project aims to provide a rich set of disintegrated network functions with the corresponding descriptors for service instantiation. The work described in this document is part of the above task and is focused on the disintegration of the Network Exposure Function (NEF), one of the network functions to be disintegrated within the FUDGE-5G project.

The first goal is to identify and disintegrate the NEF services into individual and independent NEF components, micro services. The second goal is to assure that the NEF micro services are able to provide the same service as the monolithic NEF in a transparent way for the NEF consumer. To achieve that, the micro services need to be instantiated and lifecycle managed, the third goal. At the end of the master thesis, it is expected that the work carried out shows how a implemented microservice NEF performs in terms of latency and availability when compared with a monolithic NEF implementation.

1.4 Methodology

The work presented this document followed an adapted Scrum [4] framework for development. The 3GPP NEF requirements provided the product backlog and each planned activity a sprint, that addressed the requirements belonging to the activity, the sprint backlog. The adaptation comes from the fact that there is no team, so the author played the role of product owner, scrum master and developer.

After each sprint, a retrospective was carried out. The goal was to identify what went well and not well and to make adaptations to the next sprint, in order to improve the time efficiency and the delivered work quality.

1.5 Overview of the Structure of the Document

In this section the document structure is introduced. Firstly, Chapter 2, covers the 3GPP standardization work regarding the 5G Core components in general and with a focus on the Network Exposure Function, in section 2.1. Moreover, 5GLAN concept is explained, as well as the 3GPP defined requirements. Section 2.2 describes some of the possible computational infrastructure providers. In section 2.3, orchestration tools for verticals, virtual machine and container level are presented, and finally, section 2.4 details the micro service architectural approach. Chapter 3 presents the FUDGE-5G project, the European project behind the work presented in this document. The NEF disintegration work and the micro service deployment is described in Chapter 4. Chapter 5 illustrates and describes the work plan and Chapter 0 presents the final conclusions.

2 State of the Art and Implementation Tools

This chapter covers the state of the art of the technologies that will support the development of the work described in this document. It starts by detailing the core technology behind 5G presenting the needed background regarding the 5th Generation Core (5GC) architecture and components, as well as their role into the 5th Generation System (5GS). This section also contains the description of the Network Exposure Function (NEF) services. Some relevant 5G concepts are also introduced, encompassing Non-Public Networks (NPNs), Network slicing, 5th Generation Local Area Network (5GLAN) and the Service Communication Proxy (SCP).

The second section of the chapter encompasses the infrastructure providers studied: AWS, an on-demand cloud computing resources provider, OpenStack, an opensource cloud computing software to control pools of computational resources, VMWare, a cloud computing and virtualization, and Docker, an operative system level virtualization provider, that delivers packed software units, containers.

The third section includes the studied orchestration tools. It starts with Matilda, a vertical application orchestrator. Then moves to ONAP and OSM that are opensource automation and orchestration frameworks for Virtual Networks Functions (VNFs). At the end, Kubernetes, that is a system for automatic deployment, scaling and management of containerized applications.

The fourth section provides the information related to the Microservice architectural paradigm. It defines a microservice, presents the relevance for the mobile telecommunication networks implementation and, finally, enumerates the decomposition criteria to be considered when disintegrating the NFs.

At the end, a small section resuming the relevance of all the state-of-the-art sections for the work presented in this document.

2.1 5G System

The basis for this work is 5G, the 5th generation mobile network. It is a new worldwide wireless standard after 1G, 2G, 3G, and 4G networks. 5G enables a network that is designed to connect virtually everyone and everything as one, including machines, objects, and devices.

5G wireless technology is meant to deliver higher multi-Gbps peak data speeds, ultra-low latency, more reliability, massive network capacity and increased availability. Higher performance and improved efficiency enable and provides new user experiences and connects new industries.

Most of these 5G capabilities came from the 5G New Radio [5] (5G NR). The 5G NR was developed to enhance the flexibility, scalability and efficiency of the radio both for spectrum and power consumption. One of the key differences from 5G NR to the previous radio generations radio is that new areas of the spectrum are being made available for 5G, namely the mmWave. It allows to overcome the physical limitations for bandwidth and throughput of the lower frequencies used in the previous generations.

Another important step forward in 5G is the softwarisation of the network. The network functions, that on the previous generations were hardware based, are now implemented as software running in cloud environments. The 5GC core is now software based and a new set of functionalities emerged, like Network Slicing and 5GLAN.

2.1.1 5GC Architecture and Network Functions

This section describes the 5G system architecture and its different components, Network Functions (NFs). It is a standardized architecture, and its standard can be found in the 3GPP release 23.501 [6]. Each NF has both functional behaviour and APIs defined. An NF can be implemented either as a network element on a dedicated hardware, as a software instance running on a dedicated hardware (Virtual Machine or Container), or as a virtualized function instantiated on an appropriate platform, e.g., a cloud infrastructure.

The 5G architecture enables two main characteristics: **separation of control and user planes** and **modularization**. On the first one, the User Plane (UP) carries user traffic while the Control Plane (CP) carries signalling in the network. This allows each traffic plane resource to be scaled independently. It also enables the deployment of User Plane Functions (UPFs) separated from the Control Plane functions meaning that the UPFs can be deployed closer to the users, ultimately reducing the communication latency. The second one, modularization, is achieved by having the 5GC NFs deployed in independent modules allowing for independent scaling and evolution.

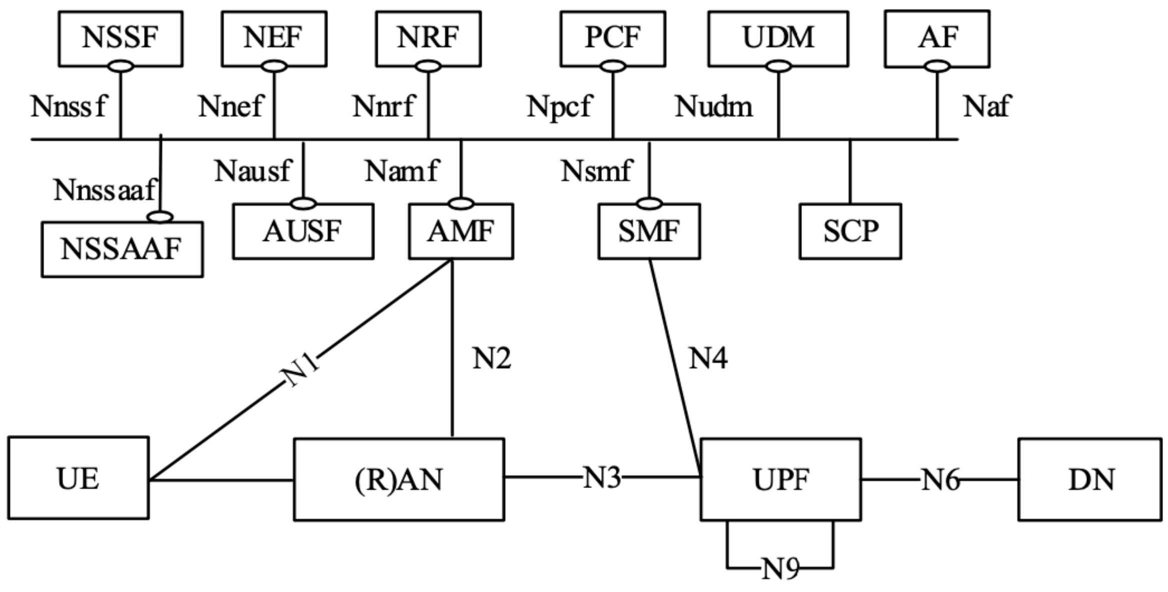


FIGURE 1 - 5G SYSTEM ARCHITECTURE (FROM [6])

The control plane functions use Service-Based Interfaces (SBI) to communicate. The user plane functions use point-to-point connections for communication. The called “N” interfaces. The 5GC NFs are described below.

Access and Mobility Management Function - AMF

The AMF is responsible for Radio Access Network (RAN) CP interface (N2) and Non Access Stratum (NAS), (N1), ciphering and integrity protection. The AMF performs registration, connection, reachability and mobility management for the User Equipment (UE). The AMF deals with the UE Access authentication and authorization, provides transport for session management messages between the UE and the Session Management Function (SMF) by being a transparent proxy for routing them. The AMF raises event notifications of UE mobility and supports network slice specific authentication and authorization.

Session Management Function - SMF

The SMF deals with session establishment and management. It deals with UE IP allocation and management (can receive it from an UPF), DHCP messages and selection of UPF by configurating the traffic steering rules. The SMF, also, enforces the policy control function, and takes care of charging data collection for charging interfaces, controlling and coordinating it at the UPF.

User Plane Function - UPF

The UPF is an anchor point between inter-/intra-radio-access (RAT) mobility and allocates the UE IP in a response to a SMF request. It is an external PDU session point to interconnect to the data network, deals with packet routing, forwarding and inspection playing policy rules enforcer on gating, redirection and traffic steering. On the downlink, the UPF buffers and marks packets triggering the subsequent notifications and performs packet duplication. On the uplink, it removes the duplicated packets and performs the packet marking.

Policy control function - PCF

The PCF supports a unified policy framework, to rule the 5G network. The PCF provides policy rules to control plane functions enforcement. It uses the Unified Data Repository (UDR) to access subscription information for policy decisions.

Network Exposure Function - NEF

The NEF securely exposes NF capabilities and events that are retrieved/stored as structured data in the UDR using the standardized interface (Nudr), this information may be used for analytics. It allows the AFs to securely provide information to 3GPP network by translating internal-external information, in particular, the NEF handles masking of network and user sensitive information to external AFs according with the network policy.

Network Repository Function - NRF

The NRF supports the service discovery function by responding to discovery request of NFs or Service Communication Proxy (SCP) with information of the discovered NF or SCP instances. It maintains NF and SCP instances profiles, when one of these instances is registered/updated/deregistered, the NRF raises notifications to the appropriate subscriptors.

In the context of network slicing, multiple NRFs can be deployed at Public Land Mobile Network (PLMN) level, shared-slice level (configured with information regarding the set of network slices) and slice-specific level (has information belonging to a Single-Network Slice Selection Assistance Information (S-NSSAI)).

Unified Data Management - UDM

The UDM is responsible for the generation of 3GPP Authentication and Key Agreement credentials and handles user authentication by storing and managing the Subscription Permanent Identifier (SUPI). It also supports the de-concealment of the Subscription

Concealed Identifier (SUCI). The UDM also handles the 5G Virtual Network (5G-VN) management.

Authentication Server Function - AUSF

The AUSF supports authentication for 3GPP access and untrusted non-3GPP access by performing authentication and authorization verifications for UEs. It is done by connecting the UDM to retrieve the authentication information and by providing the results to the AMF.

Application Function - AF

The AF interacts with the core in order to provide services that support the application to influence the traffic routing, to access NEFs and to interact with the policy framework. AFs that are considered trusted by the operator, are allowed to communicate directly with relevant NFs, otherwise, the AF uses the NEF to interact with the 3GPP NFs.

Unified Data Repository - UDR

The UDR is responsible by storing and retrieval of subscription data to the UDM, policy data to the PCF, data for exposure, application data (packet flows description, AF request information for multiple UEs, 5G-VN group information) and NF group ID matching a subscriber identifier.

Network Data Analytics Function - NWADF

NWDAF represents the operator managed network analytics function and supports data collection from NFs, AFs and from Operations, Administration and Maintenance (OAM). It exposes is own service registration and metadata to AFs/NFs and supports analytics information provision to them.

2.1.2 Network Exposure Function Services

This section describes the services provided by the NEF that will become micro services at the end of the work described in this document. The services as well as its operations are described in 3GPP release 16, technical specification (TS) 23.502 [7].

A NEF is deployed in a 5GC to expose standardized APIs for external services. The AF is the main consumer of services from NEF, but there are also services that are consumed internally to the 5GC, namely by the NWDAF and by the SMF. All the NEF services are explained bellow.

Event Exposure

The NEF here is receiving information from monitoring events, inside the 5G system, and reports them to the requesting party. These events may be detected by the AMF (e.g., UE loss of connectivity, UE location reporting, UE communication failure, number of UEs in a specific geographical area), by the SMF (UE PDU session status and downlink data delivery status) or by the UDM (e.g., UE roaming status, UE change of SUPI-PEI association, etc).

The service follows a Subscribe/Notify semantic and the consumers are the AF and NWDAF.

PFD Management

This service allows for create, update and delete Packet Flow Descriptors (PFDs). An example of use is when an AF requires that all its devices start to send information to a new application server. The AF will use this NEF service to create or update a PFD with the new IP, port and protocol to be used.

Parameter Provisioning

In this service the NEF is used by the AF to provision UE related information (e.g., network configuration parameters, location and privacy parameters) or 5G Virtual Network group information (e.g., group data or membership management).

Trigger

The trigger service allows the AF to request that a trigger is sent to an application on a UE and to receive the status of the trigger delivery.

BDTP Negotiation

This service allows the AF to request a creation or update of a Background Transfer Policy (BDTP) [8]. It allows for providers to favour specific time windows to specific UEs in a geographical area, with the intent to use non-busy hours that are cheaper and provide larger communication bitrates.

Traffic Influence

The AF uses the NEF to request traffic influence, i.e., routing decisions. This service is used when the AF is not authorized to make request directly to the SMF, so the request goes through the NEF. If valid, the request may influence the routing decisions for a PDU session and UPF selection for serving a specific UE or a group of UEs.

Chargeable Party

By using this service, the AF requests to become the chargeable party of a data session for a specific UE.

AF Session with QoS

This service enables the AF to request the NEF for an AF session with a specific quality of Service (QoS). The AF must indent the specific UE and the QoS parameters and it may, also, identify a time window and a traffic volume for the specific session. It supports create, update and revoke operations.

MSISDN-less MO SMS

This service allows the NEF to deliverer a SMS without Mobile Station International Subscriber Directory Number (MSISDN-less). The SMS comes from a specific UE connected do the 3GPP system, i.e., mobile originated.

Service Parameter

The AF uses the service to store service specific parameters in the UDR via the NEF. The NEF stores the information in the UDR under the “Application data” category. This information may be used by a UE that subscribes to notifications on the data. It supports the Create, Update and Delete operations as well as Get.

API support capability

This service is intended to inform the AF about the level of support in the API associated with a UE. These changes happen due to the possible support of interoperability with the Evolved Packet Core (EPC), i.e., 4G. If some service becomes unsupported, the AF is notified by the NEF of it. It supports the Subscribe and Unsubscribe operations (by the AF) and the Notify operation (by the NEF).

NIDD configuration

In this service, the AF is able to configure the NEF with necessary information for data delivery via the Non-IP Data Delivery (NIDD).

NIDD

This service allows for NIDD data to be delivered from the UE or group of UEs to the AF or from the AF to the UE. It supports the delivery service (AF -> UE) and the delivery notify service (UE(s) -> AF).

SM Context

This service provides the capability to manage the SMF-NEF connection that is established for the NIDD via the NEF when a UE requests to send “unstructured” data . It supports Create and Delete operations as well as the delivery operation itself. The consumer of the service is the SMF.

Analytics Exposure

The NEF is able to provide analytics to the AF. These analytics are from the UEs being served by the AF. The service provides Subscribe/Unsubscribe and Fetch operations to the AF and the Notify operation that is performed by the NEF to the subscriber AF.

UCMF provisioning

By using this service, the AF is able to provide the UE radio Capability Management Function (UCMF) with Manufacturer-assigned UE Radio Capability ID via the NEF. The UE Radio Capability information contains information on radio access technologies (RATs) that the UE supports (e.g., power class, frequency bands, etc). The service supports the Create, Delete and Update operations.

EC Restriction

This service is for allowing the AF to query the status of Enhanced Coverage (EC) Restriction or enable/disable EC Restriction per individual UE.

Apply Policy

This NEF service provides the capability to apply a previously negotiated Background Data Transfer Policy to a UE or a group of UEs.

Location

With this service, the NEF is able to provide the UE location to the AF.

2.1.3 Network Slicing

A network slice is an entire logical network that provides a set of network capabilities required by the service provider. Network slices range depending on the features of the service they are supporting.

The network slicing concept proposed by Next Generation Mobile Networks (NGMN) Alliance consists of three layers: (1) service instance layer, (2) network slice instance layer, and (3) resource layer [9]. The service instance layer represents the services supported by the network. Each service is represented by a single service instance that is provided by network operators or third parties. The network slice instance gives the network characteristics required by the service instance. A single network slice instance can be shared by multiple service instances. The network slice instance can be composed or not by sub-network instances shared by other network slice instances. The sub-network instance is a group of NFs, which run on top of logical or physical resources. The NSSF works with the Network Slice Selection Assistance Information (NSSAI) to take care of the selection and attribution of a slice to the UE on a 5G network.

2.1.4 Non-Public Networks

3GPP defines Non-Public Networks (NPNs) as: “Non-public networks intended for the sole use of a private entity such as an enterprise, and may be deployed in a variety of configurations, utilising both virtual and physical elements. Specifically, they may be deployed as completely standalone networks, they may be hosted by a PLMN, or they may be offered as a slice of a PLMN.” [10]

An NPN is a 5G private network deployed for the solo use of private entity (e.g., a vertical costumer, a government agency). The NPN is designed to support non-public use that includes infrastructural and communication services.

NPNs are divided in two categories. An NPN can be deployed as a standalone NPN (SNPN) or as a public network integrated NPN (PNI-NPN). On the SNPN type, the NPN does not rely on NFs provided by a PLMN. On the PNI-NPN type the NPN is deployed with the support of a PLMN. The PNI-NPN deployment can be done in two ways. The first way uses a PLMN dedicated Data Network Name (DNN) with a mobile pipe to forward the traffic to the NPN mobile gateway, the UPF. The second PNI-NPN deployment option uses a dedicated slice of the PLMN with exclusive resources allocated to the NPN.

5G PPP Technology Board [11], defines the following components for an NPN in a 5G system:

- **3GPP 5G Radio Access Network (RAN):** The set of base stations that provide New Radio (NR) connectivity to the UEs.
- **3GPP 5GC user plane (UP):** A set of UPFs
- **3GPP 5GC control plane (CP):** The 5GC cloud native network functions that provide signalling and packet core functionality. The SMF, AMF, PCF and the NSSF.
- **3GPP 5GC data management:** The 5GC cloud native network functions that provide authorization and authentication of UEs. They include the UDM and the UDR.
- **Service/Enterprise applications:** The applications that provide the service logic and that are hosted in cloud environments.

PNI-NPNs, normally, have the UP, CP and data management functions provided and managed by the PLMN, while the RAN is kept on the customer premises but with public spectrum. The isolation of the CP, UP and data management traffic is achieved with a DNN or a dedicated PLMN slice, as mentioned before.

In the SNPN scenarios, all components are deployed on the private customer premises, providing the required isolation. However, some SNPN owners, may deploy NPN components into hyperscalers (AWS, Azure, Google cloud) or even share the RAN with other mobile network operators.

2.1.5 5GLAN

5G Local Area Network (5G LAN) provides the same functionalities as Local Area Networks (LANs) and VPN's but improved with 5G capabilities (e.g., high-performance, long-distance access, mobility and security). Another important aspect about 5G LAN is that it will enable private communication (IP or non-IP) with seamlessly integration of 5G with fixed and wireless LAN.

5G LAN aims to provide connectivity between two or more UEs connected to a 5G network. The protocol registration and discovery, that allows a UE to communicate to another UE on the same 5G LAN group is provided by the SMF. After each UE has registered identifier information with the SMF, responsible by managing the paths across UEs in the 5GLAN group, the SMF is able to provide the necessary forwarding information. UE register and discover identifiers, during the establishment of a PDU Session, can be MAC or IP addresses. After the registration/discovery process, the SMF, managing the paths over the 5GLAN group, provides the forwarding between the UPFs ensuring the LAN connectivity between the UEs. The forwarding happens (1) between UPFs and (2) between UPFs and UEs. In the first, if the N9 interface is used, the forwarding is done using a tunnel-based approach. If the Nx interface is used, the forwarding is path-based. In the second, LAN based forwarding is used between the final UPF and the UE through the N3 interface.

Trossen [12] proposes a path base forwarding, thus using the Nx interface. A bit field is used to encode the destination UPF and provided in the packet header. Each bit position on the bitfield represents a unique link in the network. When a UPF receives a packet, checks (binary AND and CMP operation) the bitfield to find any local link being server by one of its ports, if the UPF cannot find a link, the packet is dropped. By using a binary OR for combining two or more path identifiers, multicast relations can be created. Another important feature of the path identifier is that it can used in both directions (request/response), thus not requires any path computation for the return path.

3GPP has defined the requirements for 5GLAN [13], below are listed the requirements to 5GLAN virtual network (5GLAN-VN) and to the discovery within it:

- A UE is able to select a 5GLAN-VN, that the UE is a member of, for private communication.
- A 5G system supports 5GLAN-VNs with a few to tens of thousands of member UEs.
- The 5GLAN-VN supports member UEs that are subscribed to different PLMNs, e.g., a 5GLAN-VN may span across multiple countries and have member UEs with a subscription to a PLMN in their home country.
- The 5G system supports on-demand establishment of UE to UE, multicast, and broadcast private communication between members UEs of the same 5GLAN-VN. At least IP and Ethernet data types should be supported.
- The 5G network ensures that only member UEs of the same 5GLAN-VN are able to establish or maintain private communications between each other using 5GLAN.
- The 5G system allows member UEs of a 5GLAN-VN to join an authorized multicast session over that 5GLAN-VN.
- The 5G system is able to restrict private communications within a 5GLAN-VN based on UE’s location.
- The 5G network enables member UEs of a 5GLAN-VN to use multicast/broadcast over a 5G LAN-type service to communicate with a required latency.
- The 5G system supports a mechanism to provide consistent QoE to all the member UEs of the same 5GLAN-VN. The 5G system supports routing based on a private addressing scheme within the 5GLAN-VN.

- The 5G system enables a member UE to discover other member UEs within the same 5GLAN-VN.
- The 5GLAN-type service is able to support existing non-3GPP service discovery mechanisms

2.1.6 SERVICE COMMUNICATION PROXY

This section describes how 3GPP core network functions can communicate between themselves by using a Service Communication Proxy (SCP). 3GPP describes and allows three deployment options SCP [6]:

1. SCP based on a service mesh
2. SCP based on independent deployment units
3. SCP based on Name-based Routing

SCP based on a service mesh

On this deployment the SCP follows a distributed model in where the SCP end points are co-located with the 5GC NF. The SCP end points are called Service Agents [6]. The Service Agents are responsible to implement the necessary peripheral tasks, end despite being co-located, the Service Agents and the 5GC NF are separated components. With Service Based Interfaces (SBI) communication between two 5GC functionalities (A and B), the consumer (A) communicates through his Service Agent via SBI. The Service Agent chooses a target producer based on the request and routes to the producer’s (B) Service Agent. The selection and routing policies, that are applied by the Service Agent are determined by the service mesh controller.

SCP based on independent deployment units

The 5GC NF and the SCP are deployed in independent deployment units. The SCP agents implement the HTTP intermediaries between service consumers and service producers. The SCP Agents are controlled by an SCP controller.

SCP based on Name-based Routing

In this SCP topology the SCP is based on a name-based routing mechanism that provides IP over Information - Centric Network (ICN). The IP-over-ICN [14] routing intends to establish islands of networks that, while maintaining IP-based protocols at the ingress and egress of the network, internally route packets based on Information-Centric Networking.

The core part of the SCP is a Path Computation unit, that performs Name-Based Routing. The 5GC services run as microservices in a cluster. The SCP contains a Service Deployment Cluster that encloses a Service Router. The Service Router is the communication node between the 5GC services, and the SCP and it is responsible for mapping IP based messages onto ICN publications and subscriptions, acting as a proxy. For direct communication between 5GC services within the cluster, the Service Router is not used, instead it serves multiple 5GC services when they need to communicate to other 5GC services within a different cluster.

In order to have two 5GC functionalities communication, service registration and discovery needs to take place. The service registration can be done by the 5GC Service function via the Nnrf interface. The registration request is forwarded to the operator's NRF as well to the internal Registry.

2.2 INFRASTRUCTURE

In this section are described possible infrastructure providers and infrastructure management tools. The providers can be cloud and on premisses. The management tools provide mechanisms to instantiate and administrate computational and network resources at the virtual machine or container level. AWS with AWS CloudFormation provide a tool to instantiate AWS resources, to the scope of this work, the relevant ones are the computational and network AWS resources. Kubernetes and Docker, with the container concept, supply tools and building blocks to instantiate network functions with the microservice concept behind. OpenStack is normally used in NFV deployments in the VIM layer, is opensource and gives standard interfaces that then are used by the orchestrator to access, manage and monitoring pools of resources.

2.2.1 AWS BASICS

This section describes the basic features and components of the AWS infrastructure that can be combined to provide powerful, secure and high available systems. These components include user management, network, computational instances and storage.

IAM

Identity and Access Management (IAM) is the AWS service to manage access to AWS services and resources on a secure way. In IAM there are different entities. The first one is the *user*, that represents an individual person or entity. Then we get the *groups* which englobes a set of users that will get the same permissions described on the third entity, the *policies*. The policies are a set of rules that applies to a user, a group or a role, they define the permissions and access rules to a set of resources. Finally, we have the *roles.* The role entity allows the same user to move from role to role and also a role can be assumed by different users.

EC2 Instances

Amazon EC2 provides a wide selection of VM instance types optimized to fit different use cases. Different instances provide different combinations of CPU, memory, storage, and networking capacity and that provide the flexibility to choose the appropriate mix of resources for needed for the application. Each instance type includes one or more instance sizes, allowing the scaling of resources to the requirements of the target workload. The distinct types of EC2 instances are divided into categories [15]:

- **General purpose:** These ones have a balanced mix of resources (network, computation and memory), and can be used to variety of workloads that uses these resources in equal proportions.
- **Compute optimized:** Compute optimized instances are made to be used in compute intensive applications that require high performance processors.
- **Memory optimized:** With big quantities of memory available, this type of instance suits workloads that process large datasets in memory.
- **Accelerated computing:** In these instances, hardware accelerators or co-processors are used to perform functions more efficiently then in software running CPUs.
- **Storage optimized:** When a workload performs high and sequential reads and writes to very large datasets in local storage, the storage optimized instances are the one to use, since they are optimized to provide tens of thousands of input/output operations per second to applications.

S3 – Object Storage

Amazon Simple Storage Service is storage for the Internet. It is designed to make web-scale computing easier for developers. Amazon S3 has a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable and fast data storage infrastructure that Amazon uses to run its own global network of web sites [16]. It is called an S3 bucket. An S3 bucket allows encryption when the files are in transit and when they are at rest. The bucket has a backup policy that saves versions of it along the time, and it provides a lock mechanism that can be used to fulfil legal requirements or just to ensure that the files cannot be changed.

Route 53 – AWS DNS

Route 53 is the DNS service from AWS, it allows for domain registration and the definition of a routing policy from a set offered by AWS. Route 53 connects user requests to the AWS infrastructure [17] and can also route to outside of AWS infrastructure. Another feature of AWS DNS is the ability to do health checks on the DNS end points. The routing policies offered by Route 53 are [18]:

- **Simple Routing policy:** Here there is only a name registered but with multiple IP addresses that are returned to the recursive resolver in a random order.
- **Weighted routing policy:** The traffic is routed accordingly with the weights that were attributed to the resources.
- **Latency routing policy:** Route 53 finds the resource with the lowest latency and then routes the traffic to there. This policy is mostly used when resources are in different AWS Regions with different latencies.
- **Failover routing policy:** It is used when there is an active-passive failover configuration. With the AWS health checks the traffic is routed to the active, if he fails then Route 53 routes the traffic to the passive.
- **Geo location routing policy:** The traffic is routed based on the location of the users.
- **Geo proximity routing policy:** Used when the traffic should be routed based on the location of the resources.
- **Multivalue answer routing policy:** AWS responds to DNS queries with up to eight healthy records selected randomly.

Virtual Private Cloud – VPC

VPC is a virtual network logically isolated that is used to launch AWS resources. It gives complete control over the virtual networking environment, including selection of IP address range, creation of subnets, for example a public-facing subnet for web servers that have access to the internet, and configuration of route tables and network gateways. It also allows the placement of backend systems, such as databases or application servers, in a private-facing subnet with no internet access. A VPC consists in Virtual private gateways, route tables, network access control lists (stateless), subnets and security groups (stateful).

Elastic Load Balancer

An Elastic Load Balancing automatically distributes incoming application traffic to multiple targets, such as EC2 instances, containers, IP addresses, Lambda functions, and virtual appliances. Elastic Load Balancing offers four types of load balancers that feature the high availability, automatic scaling, and robust security necessary to make the applications fault tolerant, they are [19]:

- **Classic load balancer:** it provides basic load balancing across multiple Amazon EC2 instances and operates at both the request level and connection level.
- **Application load balancer**: It is the load balancer suited for modern architectures like micro-services and container-based applications since it operates at OSI layer 7. The routing is based on the content of the request.
- **Network load balancer:** The faster load balancer because it is able to deal with millions of requests with a very low latency, it routes based on IP protocol data, and it is ideal for routing TCP and UDP traffic.
- **Gateway load balancer:** this load balancer makes it easy to deploy, scale, and manage third-party virtual appliances by providing one gateway for distributing traffic across multiple virtual appliances, while scaling them up, or down, based on demand.

2.2.2 AWS CLOUDFORMATION

AWS CloudFormation gives a way to model a collection of related AWS and third-party resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code. A CloudFormation template describes the desired resources and their dependencies so they can be launched and configured together as a stack. A template can be used to create, update, and delete an entire stack as a single unit, instead of managing resources individually.

As written before, CloudFormation treats infrastructure as code. That code is written in templates that are YAML, as in Figure 2, or JSON formatted text files. A CloudFormation template is a blueprint for building AWS resources [21]. From the template a Stack is created. A CloudFormation Stack is collection of AWS resources that are managed as a unit. A stack can be created, updated and deleted. When updating a stack, AWS CloudFormation allows the generation of a Change set that shows how the updates will impact the already running infrastructure.

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template
Resources:
  MyEC2Instance: #An inline comment
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: "ami-0ff8a91507f77f867" #Another comment -- This is a Linux AMI
      InstanceType: t2.micro
      KeyName: testkey
      BlockDeviceMappings:
        -
          DeviceName: /dev/sdm
          Ebs:
            VolumeType: io1
            Iops: 200
            DeleteOnTermination: false
            VolumeSize: 20
```

FIGURE 2 - CLOUDFORMATION YAML TEMPLANTE (FROM [20])

2.2.3 OpenStack

OpenStack is a cloud software composed of software components that enable the control of large pools of compute, network and storage in a datacentre, Figure 3 presents the OpenStack components. Besides the infrastructure as a service, OpenStack also contains components that provide fault management, service management and orchestration.

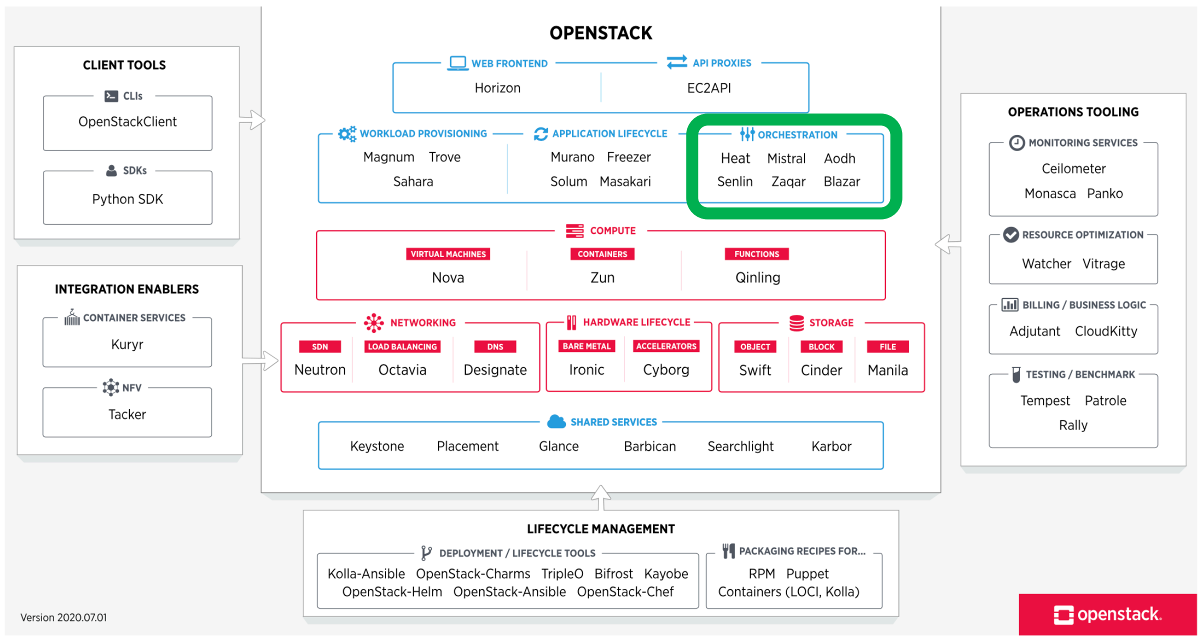
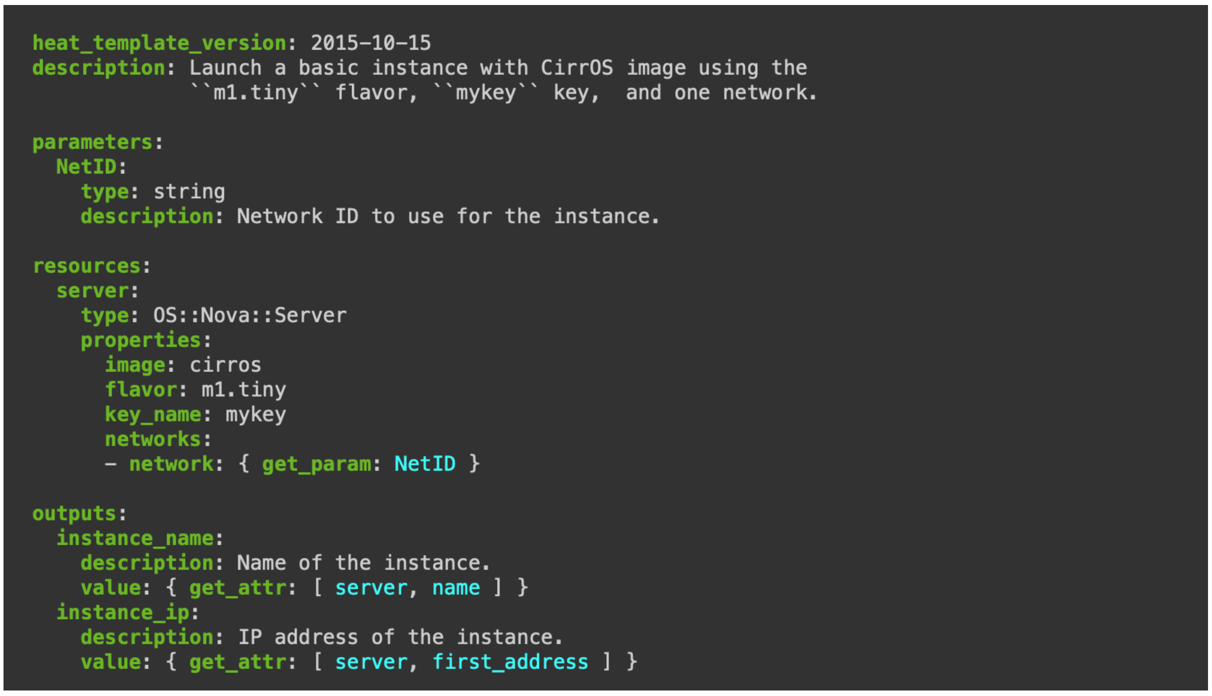


FIGURE 3 - OPENSTACK LANDSCAPE (FROM [22])

For orchestration, OpenStack provides six components listed below [23]:

- **HEAT:** HEAT uses templates to orchestrate resources for a cloud application based on templates in text files. These templates describe the application infrastructure that HEAT, by executing OpenStack API calls, generates. The templates specify the resources, that include instances, volumes, security groups, etc., and the relations between them. On Figure 4 an example of an OpenStack YAML template for the creation of HEAT stack with one instance.
- **SENLIN:** With SENLIN, OpenStack offers the capability of creation and operation of clusters of homogenous recourses, facilitating the orchestration of those collections.
- **MISTRAL:** Using YAML templates, a workflow can be described by a set of tasks and task relations. By uploading such template to MISTRAL, it can than take care of the steps described in the template. MISTRAL is a workflow service.
- **ZAQAR:** ZAQAR provides a multi-tenant messaging service bases on a RESTful API that can be used to exchange messages between the components of the application.
- **BLAZAR:** BAZAR gives the ability for reservation of services or an amount of it for a designated time period, that then it provides to users accordingly with the reservations. I can be used to provide instantiation of network slices.
- **AODH:** AODH is an alarm service that can trigger actions based on metrics collected by the data collected by the OpenStack data collection service, CEILOMETER.



```
heat_template_version: 2015-10-15
description: Launch a basic instance with CirrOS image using the
             ``m1.tiny`` flavor, ``mykey`` key,  and one network.

parameters:
  NetID:
    type: string
    description: Network ID to use for the instance.

resources:
  server:
    type: OS::Nova::Server
    properties:
      image: cirros
      flavor: m1.tiny
      key_name: mykey
      networks:
      - network: { get_param: NetID }

outputs:
  instance_name:
    description: Name of the instance.
    value: { get_attr: [ server, name ] }
  instance_ip:
    description: IP address of the instance.
    value: { get_attr: [ server, first_address ] }
```

FIGURE 4 - HEAT YAML TEMPLATE FOR CREATION OF ONE STACK (FROM [24])

2.2.4 VMWARE

VMWare [25] offers solutions for cloud computing and virtualization both for bare metal deployments and for deployments on top of an operative system. The VMWare hypervisors allow for the deployment of Virtual Machines with completely virtualized hardware. The disadvantage of VMWare is that it is not an opensource thus a license needs to be purchased. On the other end is more friendly user, thus enabling the fast deployment and instantiation of virtual environments.

A VMWare virtual environment provides the tools to manage the cluster network, storage, memory and CPU. It enables the creation of virtual networks emulating the LAN concept. At the same time, it can bridge the physical network adapter, providing connectivity for the VM. Moreover, the software also allows the sharing of the physical devices (USB, disk drive) between the hosting machine and the VM.

2.2.5 Docker

It all started with applications running in physical servers and this caused resource allocation issues [26]. One application could consume all the resources and let the others, on the same server, starving. On the other end, the resources could be iddle but still allocated. So, the technology evolved to the virtualized world. Virtualization is a technology that allows the creation of multiple Virtual Machines (VMs) on a single physical hardware system [27]. In order for this system to work, a software called hypervisor is deployed in the physical server and connects directly to the hardware. The hypervisor is responsible for distribute the resource to the different VMs. Virtualization allows better usage of the resources and at the same time to a reduction of down time of the service deployed on the VM [28]. This behaviour results from the easiest provision, deployment, cloning and replication of a VM when compared to a physical machine. The arising is: Are the VMs small enough? No, they are not. They still use a lot of resources to boot the Operative System (OS). So, the container concept was born. A container is far more lightweight that a VM [29]. Containers share the OS kernel, use much less memory and boot way faster. So, containers are the ideal tools to bundle and run applications.

Containers

A container is a standard unit of software that packages up code and all its dependencies. A container image is a standalone, executable package of software that includes everything needed to run the application: code, runtime, system tools, system libraries and settings. A container provides isolation and security allowing to run many containers simultaneously on a given host. Containers are lightweight because they don’t need a hypervisor and run directly within the host machine’s kernel. It is even possible to run containers within host machines that are actually virtual machines.

Docker

With Docker it is possible to develop, run and test applications in an isolated environment, the container [30]. Docker gives the tools and the platform to deploy, and lifecycle manage containers.

Docker architecture rests on a client-server application called the Docker Engine. This Engine has three main components. First, the docker daemon, it is the server and does all the hard work of building, running and distributing the containers, a daemon can also communicate with other daemons to manage Docker services. The second component is a Rest API that exposes interfaces that can be used to pass instructions to the daemon. The last component is the command line interface (CLI) that acts has the client and allows that instructions or scripts get to the docker daemon. The Docker client can communicate with more than one daemon.

Docker uses objects to provide the service. Networks, volumes, plug-ins are the standard across multiple platforms. Docker then extends the object library with its own objects. Starting with Images. A docker image is a template with instructions to create a container, usually they are based in other images but with additional instructions. Images can be created from scratch or found published in a registry. To build an image, a Dockerfile needs to be written with a simple syntax. The Dockerfile describes the steps needed to create the image and run it. Each instruction in a Dockerfile creates a layer in the image. When the Dockerfile is changed and the image rebuild, only the changed layers are rebuilt. This is what makes images so lightweight, small, and fast, when compared to other virtualization technologies. Another object is the container which is a runnable instance of an image. By using the Docker API or CLI, the container can be created, started, stopped, moved, or deleted. A container can be connected to one or more networks, attached to storage, or even be used to create a new image based on its current state. By default, a container is relatively well isolated from other containers and from the hosting machine. However, the isolation configurations can be changed. Finally, the service object. Docker services provide containers with scaling capabilities. The containers can be replicated across multiple Docker daemons, which all work together as a swarm of multiple workers nodes [31]. Each member of a swarm is a Docker daemon, and all the daemons communicate using the Docker API. A service allows the definition of the desired state, such as the number of replicas of the service that must be available at any given time. By default, the service is load-balanced across all worker nodes.

2.3 Orchestration Tools

The capability to automatically configure, coordinate and manage computing resources is called orchestration. It allows for suited attribution and provisioning of resources to meet the performance goals for a service. At the same time, orchestration enables the most effective utilization of the resources available. This chapter provides an overview of some tools and its building blocks that provide solutions service orchestration.

The section encompasses vertical application orchestration, NFVs orchestration and containerized services orchestration.

2.3.1 Matilda Vertical Applications Orchestrator

MATILDA vertical applications orchestrator (VAO) is responsible for the lifecycle management of cloud-native applications based on their deployment over a 5G programmable infrastructure. The MATILDA VAO is formed by the following components [32]: (i) the deployment and execution manager that produces optimal deployment plans and manages the overall execution of the application, (ii) a set of data monitoring mechanisms which collect metrics from the applications and from the network, (iii) a data fusion, real-time profiling and analytics toolkit, that use machine learning mechanisms to produce advanced insights and provide real-time profiling of the deployed components, application graphs and Virtual Network Functions, (iv) service discovery mechanisms, that follow a service mesh approach, for supporting registration and consumption of application- oriented services,(v) a context awareness engine in order to provide inference over the acquired data and support runtime policies enforcement, and (vi) mechanisms supporting interaction among the VAO and the OSS.

Deployment and Execution Manager

This Orchestrator component is responsible to materialize a vertical application (VA) placement plan, that incorporates the different components of the VA and the way that they connect to each other. The VA provider introduces some constraints regarding his VA. The telco provider interprets these constraints in a constraint satisfaction problem, when getting to a solution, a slice is created that facilitates the requirements of the VA provider. That slice is passed to the VAO, that is responsible to trigger the VIMs and to monitor the proper instantiation of the vertical components.

Matilda Agent

The main duty of the MATILDA Agent is done at the application layer (OSI layer 7). It handles the signalling between the VAO and the core components. When an VA is deployed into a MATILDA enabled provider, there is a VM to each component of the VA and all of these VM are spanned before a seven step are executed, after all the seven steps the VA component is operational. The seven steps are:

1. Agent booted.
2. Check executable prerequisites.
3. Fetch Image of Vertical Component.
4. Block until dependencies are resolved.
5. Spawn container of Vertical Component.
6. Register component to Software Defined Storage (SDS) server when health-check passes.
7. Register to a pub/sub queue.

From the previous steps, it is possible to understand that the SDS server, that acts as a key/value store, is a very important component, since it is accessible by all Agents that are booted. Besides that, it stores all information about the VA components, Agent arguments and Docker image's locations.

Monitoring Mechanisms

The MATILDA monitoring system is responsible by acquisition and management of metrics, the management of alerts and events based on these metrics, and the visualization of the available data that comes from a variety of domains that fall in four categories:

- Network function virtualization infrastructure that includes computation, network and storage resources both from virtual and physical resources.
- Software defined networks (SDN) elements, also physical and virtual.
- Physical devices that do not belong to the previous categories, like routers and switches non-SDN
- Linux containers, that form the 5G VA components, deployed.

Data Analytics Toolkit

The data analytics toolkit provides a set of native analyses processes/scripts that processes the metrics collected and presents the results in different ways. This processes/scripts include:

- Correlation Analysis.

- Time Series Decomposition and Forecasting.
- Resource Efficiency Analysis.
- Clustering.
- Filter healthy metrics.

Runtime Policies Enforcement

MATILDA follows a continuous match-resolve-act approach to provide policy enforcement. On the match phase, the alerts from the monitoring infrastructure, allow to infer the set of applied rules that then are mapped to the application deployed graphs. On the resolve phase, the conflicts between different rules that may be valid and were triggered at the same time. In the act phase, the provision of a set of suggested actions by the policy manager to the orchestration components Deployment Manager (application graphs placement) and the Execution Manager (application graphs management) is done. Policies enforcement is realized through a rule-based framework that attempts to derive execution instructions based on the current set of data and the active rules, rules associated with the deployed application graphs at each point of time.

The Policy manager consists of: (i) the working memory (WM); facts based on the provided data, (ii) the production memory (PM); set of defined rules, and (iii) an inference engine (IE) that does reasoning and conflict resolution as well as triggering of the appropriate actions.

The collection and consumption of the metrics data is based on a publish/subscribe model. That uses application graphs-oriented topics to provide the correspondent data to the Policy Manager, that converts the data into facts that can be matched to the rules on the active policies. Each rule has two parts: the conditions and the actions to meet those conditions. If the conditions are met, the policy is enforced, if the conditions are not met, the IE publishes the actions on the message broker, so the orchestration consumes them. The image below shows the interaction between the Policy manager and the monitoring agents as well with the orchestration and slice management mechanisms.

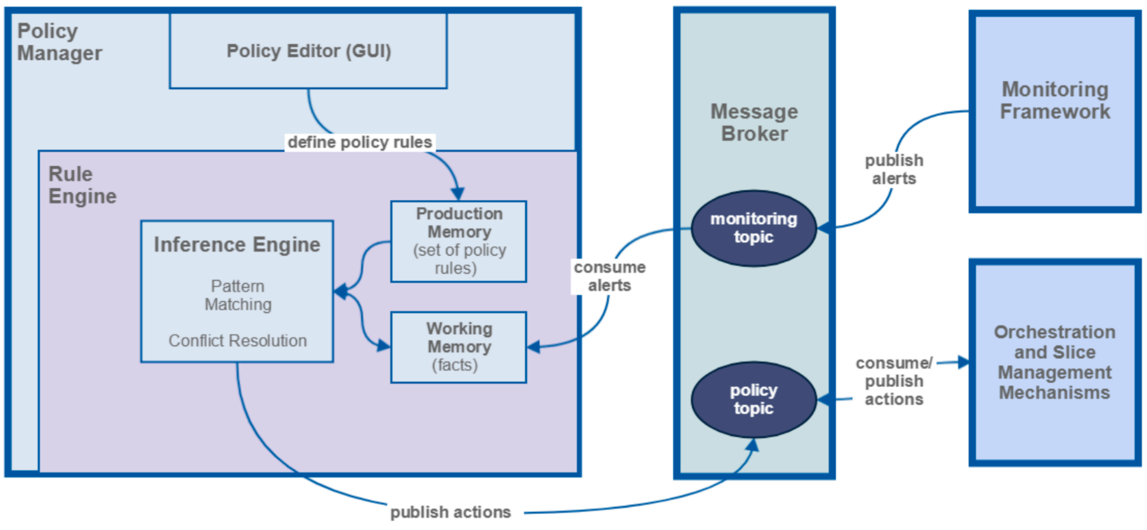


FIGURE 5 - POLICY MANAGER AND MONITORING MECHANISMS INTERACTION (FROM [32])

North bound APIs for Communication Service Providers

The Separation Support System (OSS) provides Northbound APIs towards the VAO. MATILDA specified and partially implemented to interfaces to support the Northbound APIs:

1. Interface for accepting a slice intent from the orchestrator by asking the telco provider to create the slice with the specifications of the Application graph.
2. Interface to inform the orchestrator if the slice can be created or not, so the deployment of the application can begin or not.

2.3.2 ONAP

ONAP [33] is a platform for orchestration, management, and automation of network and edge computing services. It provides real-time, policy-driven orchestration and automation of physical, virtual, and cloud native NFs that enables rapid automation of new services and complete lifecycle management, a fundamental need of 5G networks. Below some of the relevant ONAP components for authentication and authorization, inventory management, VNFs lifecycle management, monitoring and analyses and communication with the underling infrastructure.

AAF - Application Authorization Framework

AAF [34] consists of a set of Client Libraries (CADI Framework) and RESTful Services to support multiple Authentication Protocols. The AAF provides consistent authentication, authorization and security to various ONAP components.

AAI - Active and Available Inventory

Active and Available Inventory [35] (AAI) is the ONAP component that contains all the network information. AAI captures references to all the service components (network, data centre resources) and their relations. It shows the components state lively.

APPC - Application Controller

The APPC [36] API allows the management and control of VNFs's lifecycle. The APPC receives commands from external ONAP components to manage the lifecycle of virtual applications and their components.

DCAE - Data Collection, Analysis and Events

Data Collection Analytics and Events [37] (DCAE) is the main data collection and analysis system of ONAP. DCAE offers services that provide data collection and analytics.

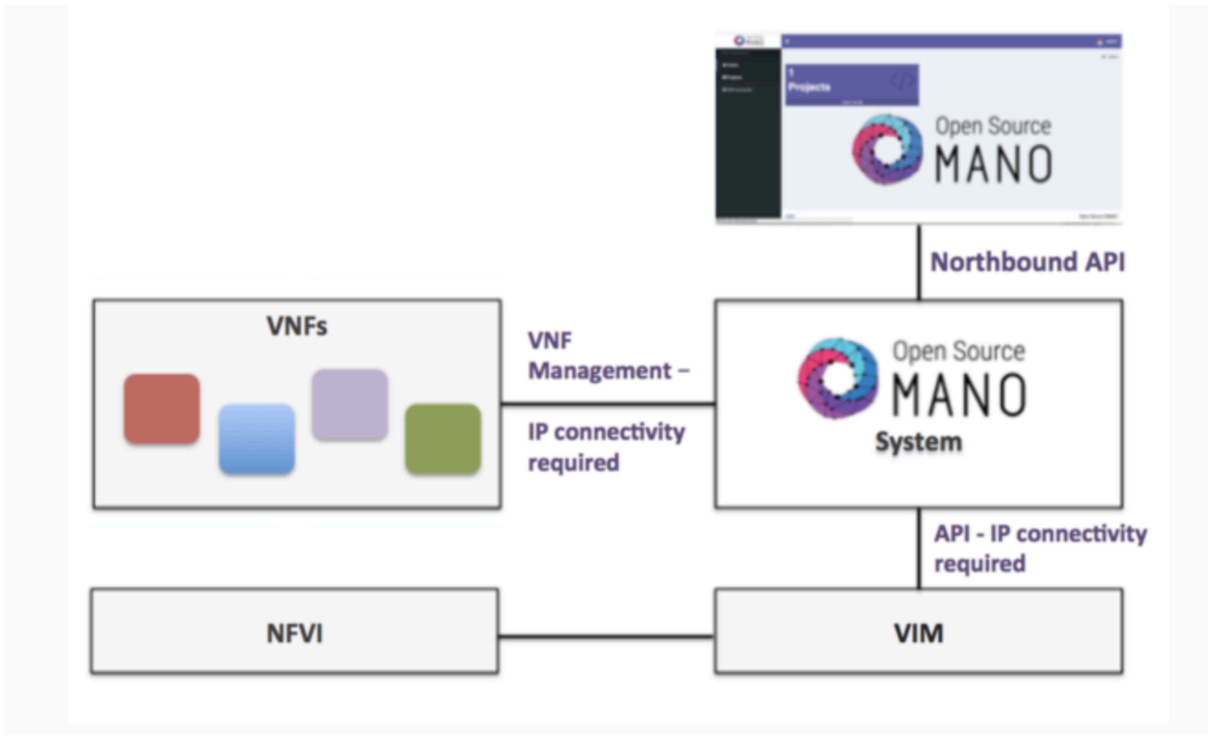
MULTICLOUD - MultiCloud Framework

ONAP MultiCloud [38] is the mediator for the communication with VIM or cloud infrastructure to:

- enable ONAP to deploy and run-on multiple infrastructure environments
- Offers decupling between the ONAP and the infrastructure, minimizing the impact of update it.
- Enables infrastructure providers to expose infrastructure's resources to ONAP allowing for optimization of placement of VNFs.

2.3.3 OPEN-SOURCE MANO - OSM

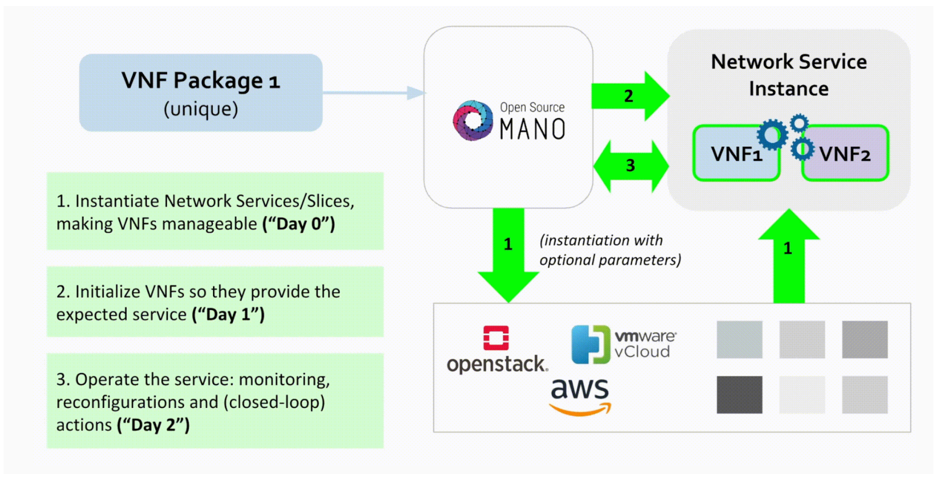
OSM is an end-to-end network service orchestrator that allows modelling and automation of telco services. With the OSM orchestrator it is possible to create network resources that are identified by a service object ID that then can be manage throw the OSM northbound API in order to control and monitor the network resource through its lifecycle. Those network resources can belong to two categories, network services and network slices (a composition of various network services). The OSM orchestrator is a consumer from two entities, the entity responsible by providing the computational resources (VMs or containers) and the entity responsible by the software defined networks.



Open Source MANO
Northbound API
VNFs
VNF Management – IP connectivity required
Open Source MANO System
API - IP connectivity required
NFVI
VIM

FIGURE 6 - OSM INTERATION WITH VIMS AND VNFS (FROM [39])

In OSM a VNF lifecycle goes through a three stage, day 0, day 1 and day 2 (see Figure 7). Day 0 stage is where the VNF is instantiated and the management channels are established and for that is required that the VNF is described and the NFVI requirements (CPU, RAM and disk) are defined. In Day 1 stage, the VNF is configured so it can start to provide the expected service, for it, it is required that the dependencies between components are identified. Finally, Day 2 which is related to modifying the VNF during runtime, it requires the definition of the dependencies, all possible configurations for runtime operations and KPIs, with these three sets it is possible to configure closed-loop operations like Auto-scaling or Auto-healing.



VNF Package 1 (unique)
Open Source MANO
2
3
Network Service Instance
VNF1
VNF2
1. Instantiate Network Services/Slices, making VNFs manageable ("Day 0")
2. Initialize VNFs so they provide the expected service ("Day 1")
3. Operate the service: monitoring, reconfigurations and (closed-loop) actions ("Day 2")
1
(instantiation with optional parameters)
1
openstack.
vmware vCloud
aws

FIGURE 7 – MANO VNF INSTANTIATION PROCESS (FROM [40])

2.3.4 KUBERNETES

Kubernetes [41] is an opensource platform, for managing containerized services and workloads. It allows declarative configuration and automation. Kubernetes automates rollouts and rollbacks, monitoring the health of the services to prevent bad rollouts. It also continuously runs health checks against the services, restarting containers that fail or have stalled, and only advertising services to clients when it has confirmed they've started up successfully. Additionally, Kubernetes will automatically scale the services up or down based off of utilization.

In a Kubernetes cluster, a set of worker machines, called nodes, run the containers. A node hosts Pods, that are the components of the application workload. A Node provides Kubernetes his runtime environment and its first component is the *Kubelet* [42]. *Kubelet* takes a set of PodSpecs and makes sure that the containers described there are running and healthy. The second Kubernetes Node component is the *kube-proxy* [43]. As the name indicates, *kube-proxy* is a network proxy that maintains network rules on nodes. It allows Pods to communicate within the cluster or to outside of the cluster. *Kube-proxy* it can rely on the OS packet filtering layer rules or do the forwarding by himself. The last Node component is the container runtime, the software responsible to run the container, Docker is one of the possibilities.

In the cluster's control plane global decisions are made and when an event occurs, the control plane detects it and responds. The first component of the control plane is the *kube-apiserver* [44] that exposes the Kubernetes API. It is Kubernetes front end, and *kube-apiserver* is designed to scale horizontally. The second component is the *etcd*. It is a high availability key store to save all the cluster data. The third component is the *kube-scheduler* [45]. *Kube-scheduler* takes in consideration resource requirements (hardware, software, policy constraints, deadlines and data locality), when a new Pod needs to be assigned to a node. The fourth component is the *kube-controller-manager,* it runs controller processes that include the node controller, the replication controller, the endpoints controller and service and token controllers. Finally, the last component of the control plane is the *cloud-controller-manager,* that is responsible to embed cloud-specific control logic. *Cloud-controller-manager* gives the ability to link Kubernetes with a cloud provider API and separates the components that interact with the cloud provider from the ones that only interact with the Kubernetes cluster.

Kubeadm, kubelet and kubectl

These three Kubernetes tools are the ones that allow an administrator to deploy and manage a Kubernetes cluster.

The first one is *kubeadm* [46]. *Kubeadm* is the tools that allows for the cluster creation with two main instructions, *init* and *join. Init* is the instruction that bootstraps the cluster control plane. *Join* bootstraps a worker node and joins it to the cluster. The second tool is *kubelet* [47]*. Kubelet* is a node agent, running in all nodes of a Kubernetes cluster. Its job is to ensure that each pod is running with the specs provided by the PodSpec file and that the containers running there are healthy. Finally, the third Kubernetes tools is *kubectl* [48]. It is a command line tool that enables the cluster administrator to control the Kubernetes cluster. Instructions like *apply, delete, get* and *describe* allows the cluster administrator to deploy, delete, list and get the description of the cluster deployed units (pods, services, etc).

2.4 MICROSERVICES

The concept behind a Microservice architecture is that complex applications become easier to build and maintain if they are divided into smaller pieces that work together to provide the same service expected from the monolithic application, but with optimized performance and resource consumption.

2.4.1 DEFINITION

Microservices are an architectural method for building applications [49]. The application is segmented into smaller pieces focused on specific functionalities [50], the microservice.

A microservice is a self-contained segment of a data, business or function domain. It contains clear interfaces that allow for development, maintenance and operation of such applications, especially their scaling.

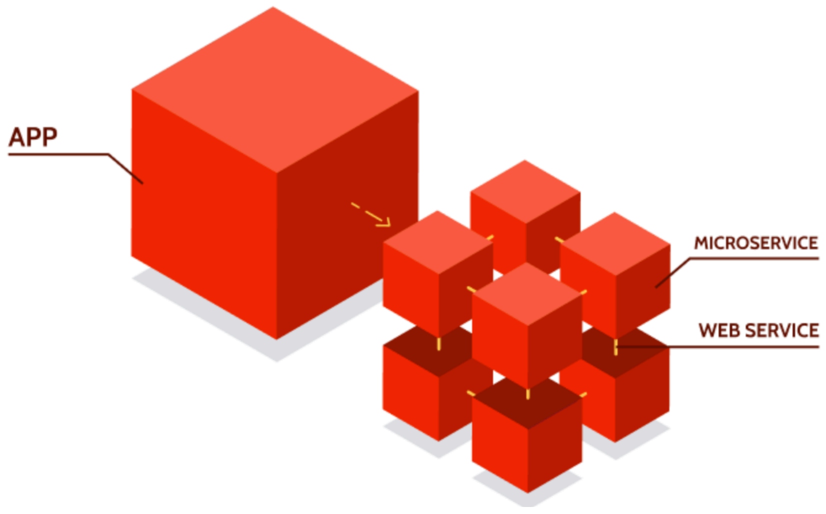


FIGURE 8 - MICROSERVICE ARCHITECTURE (FROM [49]).

In a Microservice architecture, services are:

- Processes that, over a network, communicate, interact and operate to fulfil an objective. These processes use technology agnostic communication protocols like HTTP.
- Structured around service functionalities, data pools or business capabilities.
- Capable of being implemented in distinct languages, databases, hardware and software environments.
- Decentralized, independently deployed and developed, and built and released using automatic processes.

It is a common practice to use a Microservice architecture in cloud-native applications. Cloud native applications are applications conceived and built for running in the cloud. They take advantage of the cloud resources, scaling capabilities, built-in resilience and self-healing mechanisms and are suitable for automated management, orchestration, and monitoring.

The technological enabler for the (cloud native) microservices is the container. Containers fit well on this architectural approach since they occupy a well-defined slice of the hosting infrastructure and are isolated from the other containers. Another important aspect is that containers package code, dependencies and runtime into a single binary image. These characteristics enable containers to be moved easily and to run in any environment, whether a Desktop, IT infrastructure or on the cloud.

2.4.2 MICROSERVICES ON MOBILE NETWORK SYSTEMS

Mobile network systems are based on network functions instead of generic business functions. By being part of the critical communication infrastructure, network functions have the need to satisfy requirements regarding latency and reliability. In the first three mobile network system generations, these requirements were met by using dedicated hardware implementing the network and its functionalities. In the fourth generation some industrial initiatives raised the idea of replacing the hardware-based network functions by software-based ones, the virtual network functions. With the fifth generation, the system was designed natively to be software based and modular, fostering the transition between virtual network functions (VNFs) to cloud-native network functions (CNFs).

3GPP defined the 5GS architecture, Figure 1, where the NFs are identified both for control and user plane of the 5GC. At the same time 3GPP provides the functional description, as well as the services produced by each NF to the distinct consumers [51].

In principle, each NF service shall be self-contained, reusable, and managed independently from other services offered by the same NF. These characteristics allow for agile and dynamic scaling, independent lifecycle management and data isolation, all suiting the microservice architectural paradigm.

2.4.3 Decomposition Criteria

In [51], 3GPP provides guidelines for a logical decomposition of the service producers, the NFs. The decomposition is done, in the first iteration, along the services that the producer offers. However, it is important to take in consideration other possible disintegration criteria. These criteria provides different approaches that may be beneficial for different service performance metrics that may be harmed by the strict service decomposition suggested by 3GPP. The criterium are:

- **Parallelizability**: NF services that can run in parallel should be decomposed. It saves time but at the same time requires more computational resources.
- **Bottleneck Approach**: It is good practise to isolate the NF services that may represent bottlenecks. On the other end, if a NF service executes rapidly or has a fast relation with other NF services, there may be no real advantage in splitting it from the rest of the NF services.
- **Privacy/Security**: In general, when decomposing the NF into independent services, the attack surface is increased. It may also be difficult to increase security into a heterogeneous set of containers. To reduce the threat level, it is important to configure the services in term authentication and authorization.
- **Criticality/Redundancy/Resilience**: isolating services that have strict dependability requirements will optimize the effect of their pluri-instantiation. Thus, strengthening the overall resilience of the NF against failures.
- **Dependencies**:
    - Data/State: There may exist dependencies between NF services within the same NF when they share some resources/data. There must be a criterion that is dependent on the data that a service needs to read/write/retrieve/process for running. It may be prejudicial to be performing these operations for multiple microservices, especially when there is the need to keep synchrony between multiple instances of the same NF.
    - Consumer/Producer: The lifetime of a consumer/producer relationship needs to be considered. This is especially relevant with HTTP/2 where multiplexing of

HTTP transactions in non-serial order over the same TCP connection is possible.

- **Portability**: It is beneficial to disaggregate a service from the NF when this allows the disintegrated service to run on a broader set of cloud resources with different types.
- **Automatability**: The disintegration of the NF into microservices may facilitate the automation of service provisioning.
- **Scalability**: With a disintegrated NF, its scalability capability should, naturally, improve.
- **Observability**: with the correct disintegration, the tracing, logging and monitoring of a (micro) service should be improved.
- **Upgradability**: the disintegration can be carried out with the purpose of optimizing the service update/upgrade operations easiness.

2.5 CONSIDERATIONS

The basis for this work is the new generation of mobile communication systems. From the first section of this chapter, 2.1, it was possible to identify and understand the role and interfaces of the 5GC NFs. Each of the NFs has a specific set of tasks that enable the 5GC to provide functionalities such as connectivity and mobility management, authentication and authorization, subscriber data management and policy management, among others. Special attention was given to the NEF. From the NEF section, it was possible to identify all the services provided by the NF, that will be disintegrated as microservices.

In section 2.1, three more 5G innovative concepts were explained, Network Slicing, NPNs and 5GLAN. Network slicing is the ability to provide isolated pieces of the network to specific consumers, providing isolation and a specific Quality of Service (QoS). NPNs are the focus of the FUDGE-5G project. The NPN section provided the knowledge of the deployment scenarios and components necessary to provide a mobile telecommunication service on top of a private network. 5GLAN is the transposal of the well-known concept of LAN to 5G networks. In the 5GLAN section were identified the 5GC NFs responsible for the communication establishment and management, as well as a routing protocol to forward 5GLAN packets between UE 5GLAN group members.

Section 2.2 is focused on the infrastructure providers. To run the 5GC NFs a computational infrastructure is necessary. This section gives the insight of the four possible infrastructure providers, a cloud infrastructure provider, AWS, and its management tool CloudFormation. An opensource local infrastructure provider, OpenStack, VMWare and a container infrastructure provider. AWS provides computational resources on the cloud. It has as advantage the “infinite pool” and the availability and resilience of those resources. On the other end, the cost associated has a high weight on the decision of using those resources. OpenStack provides the capability of managing the resources locally and it is mandatory the hardware is available on premises. At the same time, it is an opensource tool. Both of these tools were considered as the virtualization providers the Virtual Machine Level. Finally, Docker is the tool that delivers solutions to virtualization on the operative system level. The micro-service architecture is based on containerized applications that run on top of the operative system. Docker provides a platform to deliver the NF services as containers.

The third section, section 2.3, is focused on orchestration tools. Tools that allow for automatic instantiation, lifecycle management and scaling of virtualized computational blocks. It encompasses tools for the orchestration of Vertical applications, orchestration of VNFs and orchestration of containers. From the Matilda Vertical Application orchestrator, some of the concepts presented were applied by the OneSource team on the orchestration of Mobitrust, OneSource Vertical Application. ONAP and OSM provide the tools for orchestration in the Virtual Machine (VM) level both for the 5GC NFs and for the VMs that will host the containers of the disintegrated NEF. Kubernetes provides a platform for orchestration of containers, regarding this work, the containerized NEF services.

Finally, the last section, 2.4, describes the micro-service concept and enumerates the criteria to disintegrate the NEF services.

3 FUDGE-5G

The project presented in this document is integrated in the participation that OneSource has on the European project FUDGE-5G. FUDGE-5G stands for FUlly DisinteGrated private nEtworks for 5G verticals. The project started at September 2021 (M1) and is scheduled to finish by March 2023 (M30).

The project brings together four vendors of virtualized 5GC solutions: Athonet, Cumucore, One2many and Fraunhofer FOKUS. OneSource and Fivecomm are experts on 5G service applications. UBITECH develops a 5G vertical application orchestrator, Matilda, described in 2.3.1. InterDigital and Huawei, two major vendors of SBA platforms. Thales, a European leader in secured complex mission critical systems. And finally, Telenor that brings a state-of-the-art 5G facility.

FIGURE 9 - FUDGE-5G CONSORTIUM (FROM [52])

In a system, usually a functionality is linked with a particular system component. The cloud-native vision of SBA challenges this vision by providing the ability to transform these functionalities into decomposed, orchestrable and controllable resources. FUDGE-5G is based on the SBA vision and goes even further by committing to extend the SBA vision to both control and user plane. The to be implemented framework is called enhanced Service-Based Architecture and will decompose the 5G ecosystem into the following players: (i) NR radio access networks, (ii) eSBA 5G platforms, (iii) mobile 5G core solutions, (iv) 5G vertical application orchestration platforms, and (iv) 5G vertical service applications.

3.1 Project Objectives

The overall FUDGE-5G objective is to devise, access and demonstrate a novel cloud-based, unified and service based 5G architecture, solutions and systems for private networks. The

FUDGE-5G platform will enable extreme interoperability and customization for industry verticals. FUDGE-5G will provide solutions suitable for wired and wireless infrastructure, eSBA platform, mobile 5GC, service orchestration and vertical applications.

The overall objective is divided into seven objectives:

1. Define innovative vertical uses cases for 5G private networks.
2. Design the project platform in order to provide extreme deployment customization, both for control and user plan, for 5G private networks.
3. Devise 5G technological elements for 5G private networks, that include 5GLAN, 5GTSN, 5G multicast and the interconnection of private networks.
4. Develop and integrate the FUDGE-5G technological components into the project platform supporting a fully disintegrated 5G infrastructure for private 5G networks.
5. To carry out at least five field trials to validate the technology readiness of the project platform and its components.
6. Promote the project innovation on the relevant Standardization Development Organizations.
7. Demonstrate the value of the project innovation to relevant industry groups.

3.2 Project Work Plan

FUDGE-5G project is structured into six work packages (WP) that encompass the design, development, integration and trials of the platform and project technologies as well as the work regarding exploitation, dissemination and standardization activities and project management. The work packages are the following:

WP1 Ecosystem and Platform architecture will be focused into the design and definition both of the vertical use cases and the FUDGE-5G platform.

WP2 5G Core Technologies and Platform Development is the WP where all innovative technologies will be developed and integrated into the project platform.

WP3 5G-VINNI integration and Execution encompasses all the work required to integrate FUDGE-5G platform and verticals into the 5G-VINNI [53] facilities.

WP4 Demonstration of Products in real life conditions. The vertical use cases designed in WP1 will be trialled in order to validate the project technology components. The validation will be performed from the point of view of the infrastructure provider, as well as, from the vertical end user.

WP5 Exploitation, Standardization and Dissemination has activities that intent to improve the exploitation of the results and know-how obtained in the trials. On the other end, it intends to promote standardization of technologies on the relevant entities. Finally, WP5 covers the project results and achievements dissemination.

WP6 Project Coordination contains tasks that involve takin care of risks and implementing mitigation strategies, the communication between the project and the European Commission officers and the 5G-PPP association. Another important activity of the WP is the management of the ethics, privacy and data management of the vertical trials.

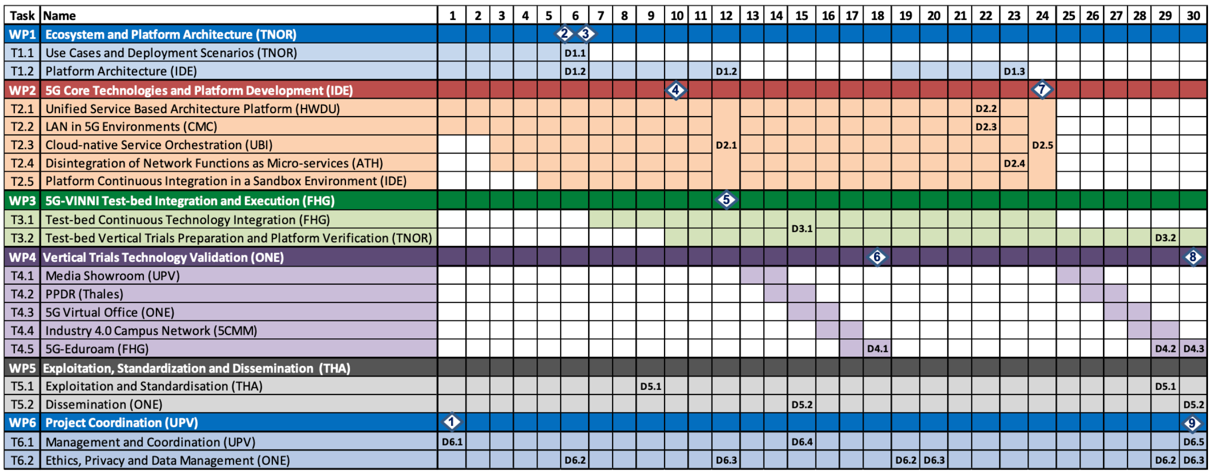


FIGURE 10 - FUDGE-5G GANTT CHART

3.3 USE CASES

The project realized five highly significative vertical use cases, namely: (i)Concurrent Media Delivery, (ii) a deployable 5G network for PPDR, (iii) a 5G Virtual Office, (iv) an Industry 4.0 Campus Network, and (v) Interconnected Non-Public Networks for university campus.

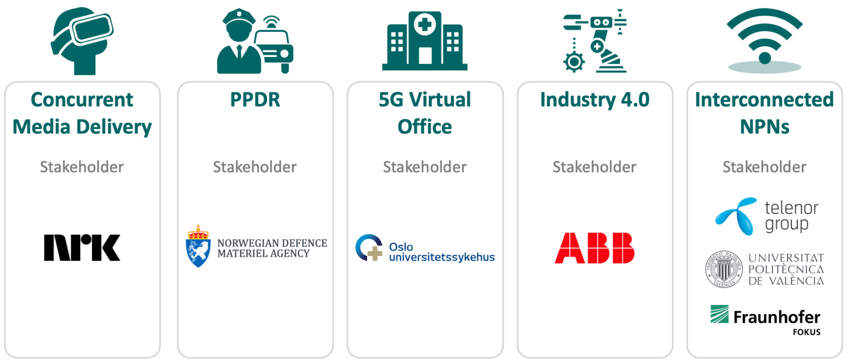


FIGURE 11 - FUDGE-5G USE CASES AND STAKEHOLDERS (FROM [54])

The **Concurrent Media Delivery** use case main goal is to use the 5G multicast capability in a 5G LAN group to delivery popular content to all mobile users belonging to that 5G LAN group. With this use case, FUDGE-5G will demonstrate how a mobile operator can optimize the radio resources when carrying out highly dynamic media distribution by showcasing a colossal video and audio media showroom. The stakeholder is NRK.

On the **PPDR** use case, FUDGE-5G will deploy Mission Critical Services with specific requirements in performance, reliability and security, executed as service applications on a pure 5G LAN. Those Mission Critical Services include push-to-talk and, video and data retrieved from operatives deployed on the field. The stakeholder is the Norwegian Defence Material Agency.

With the **5G Virtual Office** use case, FUDGE-5G will demonstrate that a set of corporate services can be accessed without any restriction regarding coverage range or proximity. This use case will be realized in an hospital and will enable hospital staff to access patients files, databases or office equipment independently of the location, allowing for the flexibility to work remotely. The stakeholder is the Oslo universitetssykehus Rikshospitalet.

In fourth use case, FUDGE-5G will bring 5G to the industry by showcasing an **Industry 4.0** campus with ultra-low latency, high reliability requirements and deterministic delivery of messages. The arrival of 5G will allow the substitution of the wired networks by wireless connectivity, enabling remote configuration and facilitating deployments. The stakeholder is ABB.

For last use case, **Interconnected NPNs**, FUDGE-5G proposes the deployment of 5G LAN across multiple independently administrated domains, a concept similar to Eduroam [55], thus providing connectivity of devices in multiple own administered domains. The use case stakeholders are Telenor, Universitat Politecnica de Valencia and Fraunhofer FOKUS.

3.4 5GLAN ON FUDGE-5G

Has described on section 2.1.5, 5GLAN aims to emulate the traditional LAN capabilities in the 5G network i.e., offer IP and non-IP communications seamlessly integration the 5G network with the fixed and wired networks. FUDGE-5G aims to go a step further and extend 5GLAN to support the so called "all-ethernet". The goal is to integrate multiple existing networks (5G, fixed, wireless) into a single converge 5G network under a unified access domain. All devices belonging to the network will be connected either by a "virtual ethernet cable" or by a real ethernet cable.

In order to provide the 5GLAN support, the (5GLAN) UPF needs to be adapted to integrate the mobile devices as part of the fixed LAN network. The adaptation will enable the connection of mobile devices to other devices part of the fixed LAN.

The (5GLAN) UPF will support ethernet PDU sessions. The SMF and the (5GLAN) UPF also need to support ARP proxying and IPV6 Neighbour solicitation proxying. As the session manager, the SMF will request that the (5GLAN) UPF, acting has the PDU session anchor, forwards the ARP/IPV6 Neighbour solicitation traffic to him.

Regarding the user plane traffic, the (5GLAN) UPF will perform packet modification. It will modify the:

- Ethernet Preamble.
- Start Frame Delimiter (SFD).
- Frame Check Sequence (FCS).

When packets enter the 5G network, the (5GLAN) UPF removes the ethernet preamble, the SFD and the FCS from the ethernet packet. On the other direction, the (5GLAN) UPF adds the three fields to the packets leaving the 5G network.

In a 5G network, the SMF is the NF responsible for allocation and attribution of IP addresses to the UEs. In the fixed LAN, the IP addresses are allocated and attributed by the Dynamic Host Configuration Protocol (DHCP) server. If not managed correctly, the external attribution would cause problems on the address management of the 5G network. The solution is to use, instead of the IP address, the UE's MAC address. The (5GLAN) UPF is responsible for storing the UE's MAC address and to associate it with the corresponding PDU session.

The other service required for the integration of LAN into the 5G system, is the creation and management of 5G VNs to provide 5GLAN UE groups. There are two options for the management. The groups can be dynamically created by the 5G operator, or they can be managed by an AF that uses the exposed services of the 5GC NEF. In order for the NEF to provide the service, it may include the Group Management Function (GMF). The GMF will implement and expose the service for create, update and remove a 5GLAN group. The service could them be used by an AF to manage the groups.

Once the 5G VN is created, a member UE will be accessing it on the specific created PDU session for that 5G VN.

3.5 ONESOURCE ROLE

OneSource has a role in all FUDGE-5G WPs. OneSource leads Use Case (UC) 3 – 5G Virtual Office. It means that the company produced the UC3 blueprint presented in the Derivable 1.1 [54]. For that deliverable, the Master Thesis author wrote the entire section 3, with the supervision of Dr. André Gomes. Yet for Deliverable 1.1, OneSource participates in UC2 - PPDR and UC5 – Interconnected NPNs. For UC2, the Master Thesis author contributions are related to the Vertical application, Mobitrust, description. All this works belongs to FUDGE-5G WP1.

Regarding WP2, OneSource contributes for Task(T) 2.4 – Disintegration of Network Functions as Micro-services. For the T2.4, each partner contributes with the disintegration of one of the NFs. OneSource got the NEF. The work presented in this document, at its submission date, reports OneSource contribution for T2.4, the disintegration of the NEF as Microservices.

For WP3, OneSource leads both tasks regarding UC3. The work encompasses the coordination and integration work, involving the remaining partners, of the different components for the Use Case testbed. The Master Thesis author participation here was on the management of the meetings, as well as the continuous monitoring of the execution of the planned work.

At the time of submission of this report, WP4 was not started yet.

With respect to WP5, OneSource leads T5.2 – Dissemination. The task comprises the management and content update of FUDGE-5G dissemination channels, which includes news and newsletter publications, as well as assuring that the project outputs are made available for the public. The Master Thesis author was responsible by doing the work described, with the supervision of the communication team leader. For T5.1, The Master Thesis author was responsible for describing OneSource exploitable assets and UC3 expected innovations. The work was developed with the supervision of Dr. André Gomes. Another important output of the participation on WP5, was the extended abstract - FUlly DisinteGrated private nEtworks for 5G verticals [56] and [57]– submitted and accepted for 2021 EuCNC [58]. The Master Thesis author took part on OneSource contributions for the extended abstract and was responsible by the presentation during the conference. The video is available here [59].

Finally, WP6 where OneSource is the leading partner for T6.2 – Ethics, Privacy and data management. For these tasks, the output was D6.2 – Data management plan. For that deliverable, the Master Thesis author contributions encompassed the consent template and information sheet template, as well as multiple contributions to different sections of the document.

When the Master Thesis was submitted, OneSource was, also, taking part of a Non-Public Networks white paper. The Master Thesis author was responsible by the identification of relevant uses cases, and their benefits of using NPNs, to be included in the white paper.

3.6 5G VERTICALS

A vertical application is an application designed for a particular market or industry. It is business specific software designed for a specific domain. On the 5G world, they are applications designed to address the needs of specific vertical sectors.

Despite the natural target of satisfying the common human communications (voice, data and internet), 5G has the goal of improving and accelerating the economy and global digital transformation. 5G aims to provide communication solutions for vertical sectors as automotive, manufacturing, media, energy, eHealth, public safety and smart cities.

In order to contribute to the digitalization of public awareness and eHealth, OneSource developed an application capable of providing situational awareness for different operational theatres.

3.6.1 MOBITRUST

Mobitrust [60] is an end-to-end platform that includes all components to provide situational awareness for mission critical services. The platform includes the devices and a control and command centre in completely integrated all-in-one platform.

The platform provides a variety of devices that field operators and patients wear. The devices provide a wide range of sensors and related equipment. The devices enable monitoring of a variety of indicators that include GPS positioning, man-down detection, electrocardiogram, respiratory rate as well as, environment monitoring to detect hazardous situations. Mobitrust also offers communication with on-demand real-time video and audio. All the collected data and communications are sent to command-and-control centre, enabling a continuous monitoring and situation awareness from the decision makers.

Mobitrust is implemented into two flavours, one designed for Public Protection and Disaster Relief (PPDR) operations and the other one for the eHealth vertical sector. The first one is be demonstrated on the FUDGE-5G PPDR use case and the second one showcased at the 5G Virtual Office use case.

Mobitrust Components

Mobitrust is composed by a set of components and technologies that working together provide the platform services represented in Figure 12. The components are:

Orchestrator: It is the brains of the platform. The orchestrator manages the integration of all components, and its tasks include the operations that provision and association of the WebRTC mount points, the End User device sensors' driver association and the fault tolerance mechanism.

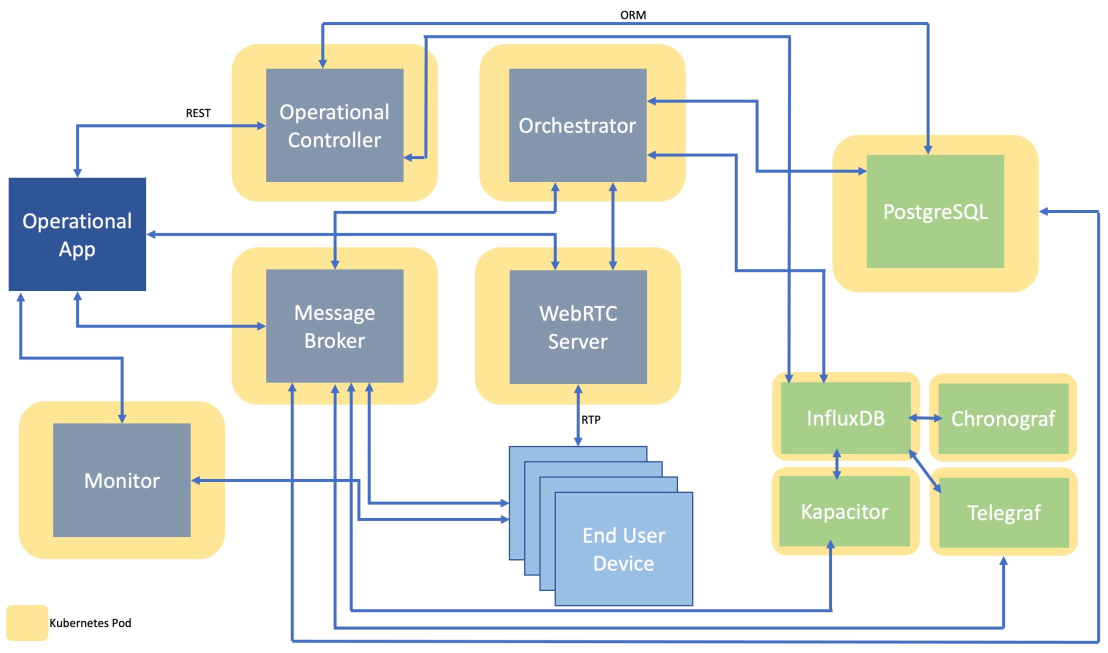


FIGURE 12 - MOBITRUST COMPONENTS AND ITERATIONS

Operational Controller: The operation controller is responsible for the services provided by the Command-and-Control Centre. It has all the backend operations that enable the visualization of the data collected by the platform. The component is also responsible by processing the request of the human operator.

Operational App: It is the frontend of the platform. Provides the visualization of all the data collected by the platform, including sensor data and live video. The Operational app also provides the interface for the human operator.

Message Broker: The component is the communication backhaul of the system. It follows the publish/subscribe model. The message broker is responsible for the communication, both control and data plane, between the components. It deals with configuration messages and with all the sensor collected data transmission.

WebRTC server: The WebRTC server is the component that deals with the voice and video transmission. It provides the mount points that enable the voice and video transmission between the End User Devices and the Command-and-Control Centre.

Monitor: The monitor is the micro-service responsible by watching and reporting on the state of the end-user devices. It uses the Message broker to ping the End User Devices and by that, monitor their state.

PostgreSQL Database: The relational database is the component of the system that stores the information regarding users, End user devices, mount points and their associations, as well as the access control policies.

Tick Stack: The component that stores the data associated with a timestamp. It is composed by influxDB, Telegraph, Kapacitor and Chronograph. It stores all the sensor data and system events. The component has a paramount importance when data forensic analyses is required.

End User device: The wearable that incorporates all the equipment that allows for the sensor data collection as well as for the real-time video and audio communications.

3.6.2 5G Virtual Office

A 5G Virtual Office provides secure access to a specific set of corporate services. This means that a 5G device can communicate with any other device member of the 5G Virtual Office, if there is any type of 5G coverage, including both indoors and outdoors. In an assumed hospital environment, a group of heterogeneous individuals, from different teams and with distinct responsibilities (doctors, nurses, paramedics), share a set of office resources and have the need to communicate both with each other, and with the hospital physical resources in a reliable way.

For the specific context of a hospital as considered in this use case, the main application is that hospital staff is not bound by location to access medical devices, electronic health records, or any office equipment, allowing the flexibility to work remotely (e.g., paramedics accessing patient health on the go, video conferencing diagnosis, remote operation of medical equipment), including from a patient premise or another off-site location. The Vertical includes scenarios that encompass Ward Remote Monitoring, Intra-Hospital Patient Transport Monitoring and Ambulance Emergency Response

Ward Remote Monitoring

Unlike Intensive Care Units (ICUs) where monitoring of patients is continuous and hospitals tend to have better equipment and more qualified staff, hospital wards have the capacity to accommodate big quantities of patients, and the amount of time that the medical staff has to monitor each of the patients individually is very short. Another important aspect is that wards have sparser monitoring resources. This leads to additional delays in the detection of symptoms, some of them life threatening, which increase the overall mortality rate. Thus, if more sophisticated monitoring can be expanded to wards and if it can be done with less human interaction/less need for qualified staff, the overall quality of healthcare can improve, and mortality can decrease.

In the 5G Virtual Office use case, every time a patient enters the ward (either from admission or from ICU), it receives a smart shirt to monitor cardiac and respiratory functions. All the data collected by sensors is centralized in the hospital network and sent over the 5G non-public network. This collection and the fact that it is done in real-time over 5G has multiple benefits: first, it has no wires and does not depend on sketchy Wi-Fi coverage, meaning the patient can be easily moved to another location without monitoring disruptions (see the next sub-scenario); second, it enables centralized data fusion and processing with machine learning algorithms that detect abnormal readings and dispatch alarms to the relevant staff. The latter benefit is extremely important towards an effective monitoring, and it further reduces the need for qualified staff to be constantly monitoring each patient. In fact, human intervention is reduced and only exists either when an abnormal reading is automatically detected or when a consultation is given to the specific patient.

While the patients are undergoing medical care at the ward, qualified staff can also remotely access all information and interact with patients in real time, even performing diagnostic tests and complex procedures (for example, ventilator configuration). This means that qualified staff and specialists do not need to move within the hospital at all times, reducing wait times, increasing efficiency and also reducing the probability of spreading contagious conditions.

Intra-Hospital Patient Transport Monitoring

A large percentage of patients need to be moved across the hospital building to perform exams, sometimes covering distances close to one kilometre. These patients are heavily monitored by multiple equipment types with wires, thus moving them around is complex.

This becomes a challenge when patients need to be monitored at all times, in particular ICU patients due to their underlying conditions and invasive monitoring equipment. Additionally, some of the sensors (including ECG, blood pressure, SpO2, respiration rate) may require very high sampling rates (> 100Hz) and will generate huge quantities of data that needs to be transmitted in real time.

With the 5G non-public coverage across the entire hospital, moving patients around, when necessary, becomes easier as coverage is highly dependable. Moreover, as it lacks wires and supports high bandwidths/low latencies, it is also a great alternative to existing wired solutions. As in the ward, such patients can have greater quality of monitoring delivered by the network solution, but also a better care overall because even if a qualified person is not monitoring vital signs at all times, an automated system is always performing a real time analysis to detect abnormal patterns and will send alerts as soon as that happens.

Ambulance Emergency Response

An ambulance is dispatched to an emergency call and uses an NPN on top of public 5G networks to remain connected, thus obtaining seamless connectivity with the required security and performance.

When an ambulance is called on site, paramedics aid the patient, where they usually receive only a brief report from the emergency call centre, identifying the patient with information obtained from the emergency contact. The paramedics can now check the patient’s medical history and information regarding underlying conditions (for example, medication being taken by the patient, allergies, recent hospital visits, chronic illnesses, etc.) from the emergency response vehicle. The information is readily available during the journey to the patient's location. Based on this patient information, paramedics apply the appropriate procedure to stabilize the patient and start moving towards the hospital. Along this path, the patient is monitored with cameras, microphones and biosensors. This information is uploaded, stored and viewed at the hospital to prepare for the patient's arrival. Still, additional diagnostic tests can now be performed to save time. If necessary, the doctor can instruct paramedics to apply specific procedures or medications.

This is particularly relevant as, typically, an ambulance has a crew of two paramedics, one being the driver, the other staying with the patient. The problem arises, when there are some tasks (bureaucracy, monitoring, reporting to the hospital), for which both are necessary. When having a doctor intervene remotely with the ambulance, as described above, the driver will be responsible only for making the journey to the hospital. The remote

support of the doctor allows the driver to focus on a specific task, enabling the redistribution of qualified paramedics to new ambulances to use resources more efficiently.

Upon arrival at the emergency room, everything is ready for the patient and the connectivity of any remaining monitoring equipment is assured by the hospital’s non-public 5G network.

Mobitrust is the Vertical Application that will enable all the monitoring and data access showcased into this vertical trial. OneSource is the leading partner of the Use Case, the content presented in chapter 4 of [61] was produced by the author with the supervision of Dr. André Gomes.

3.6.3 PPDR

Public Protection and Disaster Relief (PPDR) applications include software and hardware solutions that enable emergency agencies to improve their coordination activities to respond to threat events. Rescue operators, police forces, firefighters, ambulance services and civil defence are examples of verticals targeted by PPDR applications.

PPDR operations, nowadays, involve several and distinct entities with a multiplicity of actors deployed for different disaster scenarios in operational theatres (e.g., fires, earthquakes, war zones). In order to fulfil the common mission objectives, efficient coordination at operational, tactic and strategic level represent decisive factors. Thus, communication is a key enabler for information to transverse the PPDR hierarchy, allowing the actors to make correct and timely decisions. Advances in computing capabilities and cloud architectures have made possible new kinds of information sharing, including high quality audio/video communications, image recognition and classification, as well as, near real time telemetry (e.g., geopositioning tracking, vital signs, environmental and terrain monitoring). Likewise, the latest developments in telecommunications allow better support of communication in those environments (e.g., one-to-one calls, push-to-talk mechanisms, and multi-party conferences), which ultimately enable better collaboration models between law enforcement forces and first responders and improve the overall situation awareness. The Vertical trial encompasses scenarios that demonstrate different types of backhaul connectivity of the PPDR operational mobile telecommunications infrastructure: autonomous edge, intermittent connectivity to a remote cloud and the coexisting of the private and public mobile telecommunications network.

5G core and cloud applications deployed in FUDGE-5G mobile autonomous edge

In this scenario, a mobile autonomous edge is deployed, providing a communication bubble based on a 5G Non-public Network. A vehicle carrying the required infrastructure (5GC, Radio and Vertical Applications) provided 5G coverage for a mission critical operations theatre. With this deployment, the mobile telecommunication is assured in any circumstances. The scenario aims to demonstrate the capability of delivering 5G coverage anywhere, enabling the support of the PPDR operations [61].

Intermittent backhaul connectivity and heterogeneous deployment

The communication bubble showcased in the previous scenario, is based on limited computational capabilities. Those capabilities are the ones needed to support the minimal mobile telecommunication requirements. When on a PPDR scenario where the national/local communication infrastructure is destroyed or damage, eventually it will start to recover. This scenario aims to demonstrate the FUDGE-5G platform capabilities to leverage remote cloud capabilities provided by the nationwide telecommunications infrastructure that is now starting to recover from the disaster. The goal is to overcome the limited computational capabilities of the theatre deployed telecommunications bubble with enhanced performing applications, with fewer resource constraints. Without a full recovered national telecommunications infrastructure, only vertical applications will be considered to run on the remote cloud. Thus, only user plane traffic will be shifted. The control plane will stay at the bubble, that is the only capable of provided reliable performance.

Coexistence of public and non-public networks

On the previous scenarios it is assumed that the national/local mobile telecommunications infrastructure is destroyed or damage, however, it does not always happen. In this scenario is assumed that there are available other public and non-public networks and that the mobile devices are capable to connect to them, in different network slices. These networks can offer other services that may not be mission critical. Therefore, one slice offers connectivity to the public network, with best-effort performance. The other slice is the one that is served by the FUDGE-5G communication bubble and that assures connectivity to the local deployed mission critical services and PPDR applications. This slice provides a better QoS regarding prioritization, traffic isolation and security.

One of the PPDR applications providing mission critical services, on the three scenarios, is Mobitrust. Mobitrust will be responsible for the situational awareness service, enabling the monitor of the deployed forces as well as the surrounding environment. With those capabilities, the task leaders will be able to get near-real time information from the operations theatre.

The complete blueprint of the Use case can be found on section 3 of [61].

4 NEF AS MICRO SERVICES

The 5GC was designed following the 3GPP Service-Based reference architecture [62], allowing the exposure of the network capabilities to operators, application developers and service provides via standardized APIs. However, the 5GC NFs are standardized to be deployed as VNFs. VNFs are monolithic entities deployed on top of Virtual Machines.

The work, described below, is integrated into task 2.4 - Disintegration of Network Functions as Microservices - from FUDGE-5G Work Package 2 . For this task each partner selected the 5GC NF that they plan to disintegrate. OneSource chose the NEF to disintegrate due to its previous developed work with this particular NF, showcased in the project Mobilizador 5G [63].

4.1 PROBLEM STATEMENT

As the 5GC evolves, it needs to provide an even bigger variety of services. The increasing number of services, built on top of the monolithic NFs, increase exponentially the complexity of the code. Activities like bug resolution, interface modifications, adding capabilities impact the NF as a whole.

In the last version of 3GPP rel. 16, the NEF is standardized to support around 20 services with different development challenges, runtime requirements, maintenance needs and downtime impact. With the deployment of a monolithic NEF these service characteristics will propagate to all the NEF services, thus limiting the performance of the overall NF.

One can argue that the NF can be scaled, i.e., deployed with multiple instances. It is a possible approach, but with big inefficiencies. If the NF is replicated due to service A that faces a large load/traffic and at the same time, NF service B requires a big chunk of computational capabilities but faces a low number of requests, that do not require replication. By replicating the NF, those computational capabilities will be reserved but idle, increasing cost to the service provider, and ultimately to the service consumer.

With monolithic NFs, a single bug on one of its services can bring down the whole NF, the same is true for updates, even the smaller ones require the redeployment of the whole NF. Technologies that are well suited for a specific service may become difficult to adopt due to the implications on the remaining services.

From the point of view of the network operators [64], the challenge is to have an environment where they can deploy and upgrade services frequently, meeting the market demands.

Monolithic NFs do not suit these requirements. In reality, they do not benefit from the cloud capabilities (scaling, self-healing, intelligent orchestration, etc) since they do not offer the ideal deployment unit.

4.2 Proposed Solution

In order to overcome the problems identified for NFs deployed as monolithic entities, and in particular, the monolithic NEF, the solution is to follow a micro service architecture. The NEF services , described at section 2.1.2 are disintegrated, decoupling the NEF in small units that communicate using the same interfaces and by using HTTP, thus following the standardized APIs from 3GPP.

By deploying the NEF as a set of independent micro services a number of advantages arise:

- The services become easier to manage due to their smaller size
- When an update is needed, only the update service needs to be redeployed, maintaining the remaining services unchanged and on-line. The redeployed time is also smaller that the redeployment of the monolithic NF.
- The technology used for the service is the one that better fulfils the business requirements.
- The failing of the service does not affect the remaining NF services.

Instead of running the NF on top of a Virtual Machine, the services are containerised and deployed on top of a container engine. By transforming the monolithic NEF into a set of containers, the exploitation of the cloud capabilities becomes efficient, since a container is the ideal deployment unit for cloud deployments.

4.2.1 Solution Implementation

In order to implement the NEF services as micro services (Scenario 1), each one of them was written into a python version 3 executable. Since the 5GC is standardized as RESTful APIs, each executable runs a local instance of the Quart framework [65]. To perform the HTTP requests, python library Requests [66] was used. To transport the data inside the HTTP requests and responses, JSON format was used.

Regarding the architecture design(see Figure 13), a NEF gateway exposes all the NEF services to the consumers (AF, SMF and NWDAF) and performs the SCP role (see 2.1.6) based on independent deployment units. When a request arrives at the gateway, it is responsible to forward the request to the target micro-service and, once received the response, it forwards it to the service consumer. In order to avoid processing bottleneck at the gateway, no processing is made there, it just forwards the requests and replies.

Each NEF service container is deployed into a Kubernetes pod that runs in a node. A pod is not shared by multiple containers, but a node may host multiple pods. The micro services are implemented to support multiple instances but on separated pods.

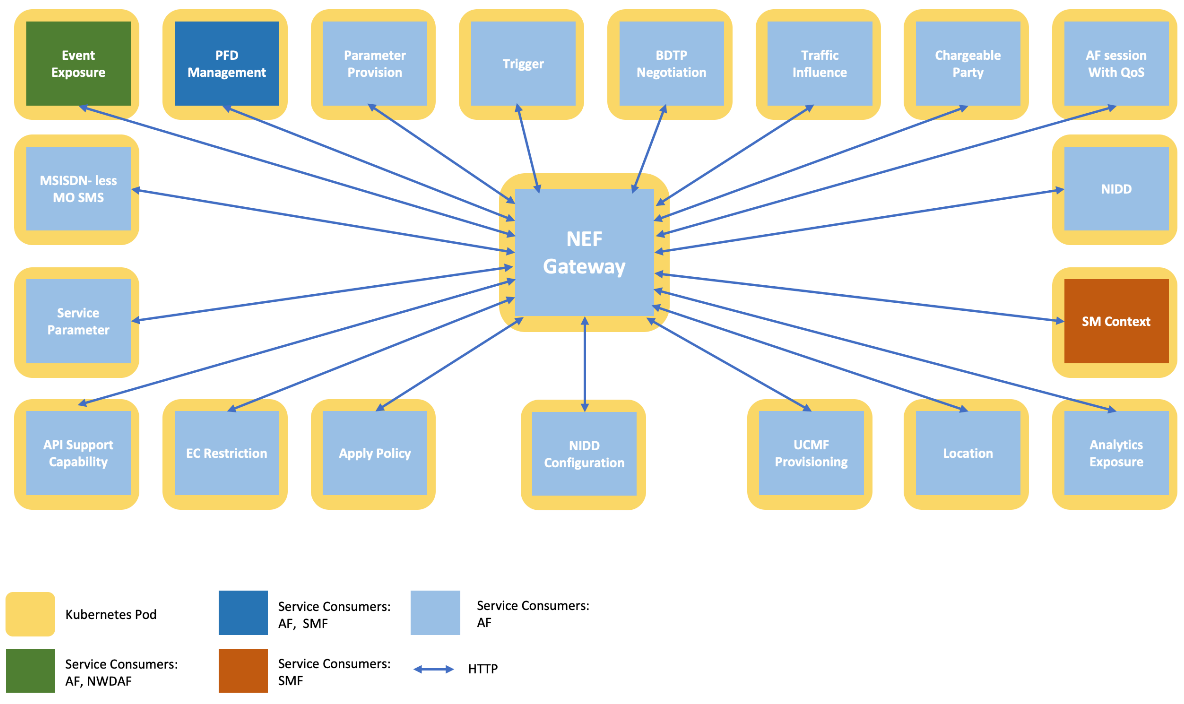
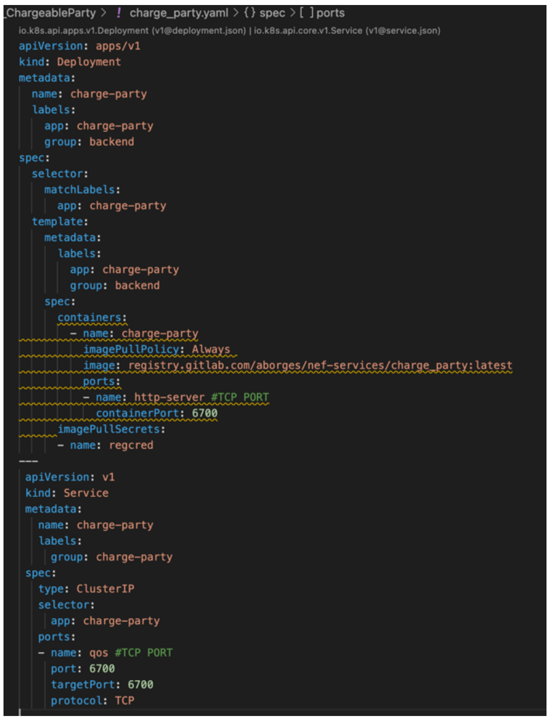


FIGURE 13 - DESINTEGRATED NEF ARCHITECTURE

Moreover, to perform the experiments, a monolithic version of NEF was also created (Scenario 2). This version was used to establish the experiment baseline that allowed for the micro-service architecture performance to be compared.

Each of the python executables, including the gateway, the micro-services, and the monolithic NEF version, were packed as a docker image. The docker images were, then, saved to a GitLab image registry. From the image registry, Kubernetes is able to pull and deploy them into a cluster.

The Kubernetes deployment instructions are written into a YAML file, Figure 14. A file for each of the microservices was written. The file has two parts. The first one defines a deployment instance - in this case it is a container. The first part has the instruction from were to pull the image and the port that is open on the container. The second part defines the service characteristics, in the example a ClusterIP exposing port 6700. The service deployment is managed by kubectl.



```
ChargeableParty > ! charge_party.yaml > {} spec > [ ] ports
io.k8s.api.apps.v1.Deployment (v1@deployment.json) | io.k8s.api.core.v1.Service (v1@service.json)
apiVersion: apps/v1
kind: Deployment
metadata:
  name: charge-party
  labels:
    app: charge-party
    group: backend
spec:
  selector:
    matchLabels:
      app: charge-party
  template:
    metadata:
      labels:
        app: charge-party
        group: backend
    spec:
      containers:
        - name: charge-party
          imagePullPolicy: Always
          image: registry.gitlab.com/aborges/nef-services/charge_party:latest
          ports:
          - name: http-server #TCP PORT
            containerPort: 6700
      imagePullSecrets:
      - name: regcred
---
 apiVersion: v1
 kind: Service
 metadata:
   name: charge-party
   labels:
     group: charge-party
 spec:
   type: ClusterIP
   selector:
     app: charge-party
   ports:
   - name: qos #TCP PORT
     port: 6700
     targetPort: 6700
     protocol: TCP
```

FIGURE 14 - NEF MICRO-SERVICE YAML FILE

In order to perform the experiments, a NEF consumer was also implemented, an AF. The AF is the entity that requests actions provided by the NEF services through the RESTful APIs. For the AF, the request process is transparent in both scenarios.

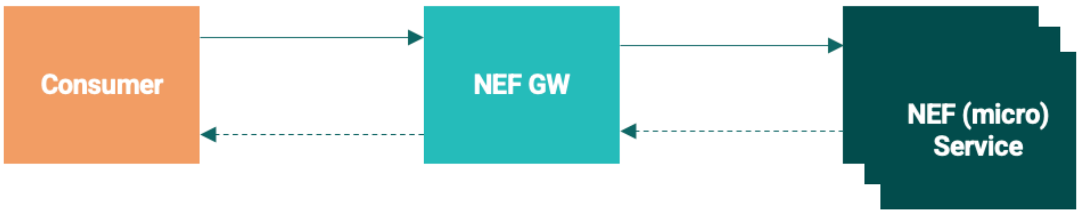


FIGURE 15 - COMMUNICATION SEQUENCE SCENARIO 1

In Scenario 1 (Figure 15), the request is made to the NEF gateway and forward to the target (micro) service. The response follows the reverse path.

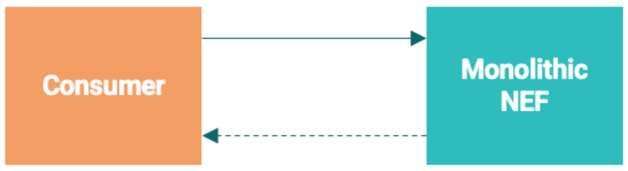


FIGURE 16 - COMMUNICATION SEQUENCE SCENARIO 2

In the second scenario, the request is made to the monolithic NEF (Figure 16), that replies as soon as the requested action is performed. It is expected that the requests made to the NEF as micro-services take a higher time to be replied, due to the additional communication step.

4.2.2 TESTBED

The testbed,(see Figure 17), is supported by a server with 8 AMD Ryzen 7 CPU cores @ 3.6GHz, 32GB DDR4 RAM and 500 GB SSD. The server deployed four Virtual Machines (VMs).

The Kubernetes cluster has three nodes, hosted within VMs (see Figure 18). A fourth VM was deployed to run the AF(s). Table 1 displays the hardware of each of the computational nodes.

TABLE 1 - TESTBED HARDWARE

Virtual Machine	Hardware
Master Node	• 8 vCPUs • 8 GB DDR4 • 40 GB SSD
Worker Node 1	
Worker Node 2	
AF(s) VM	• 2 vCPUs • 4 GB DDR4 • 20 GB SSD

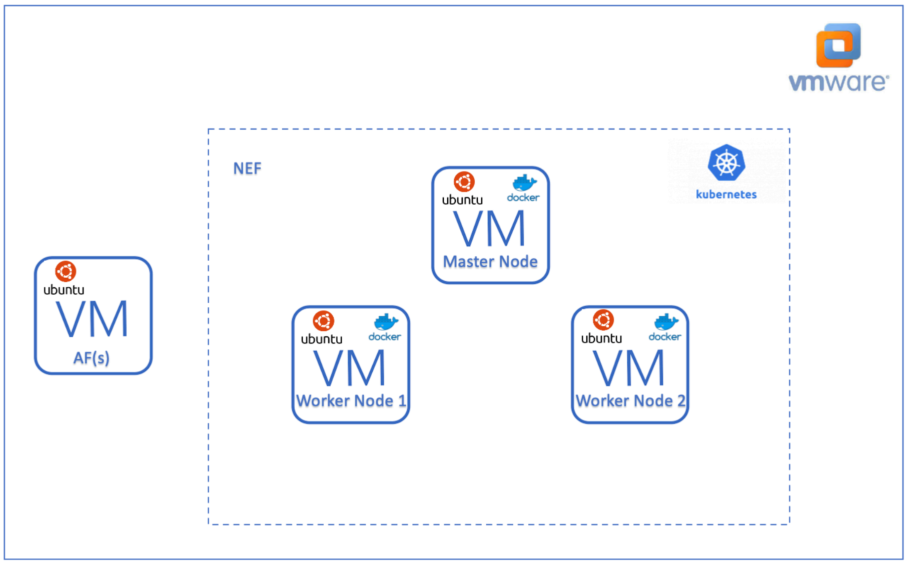
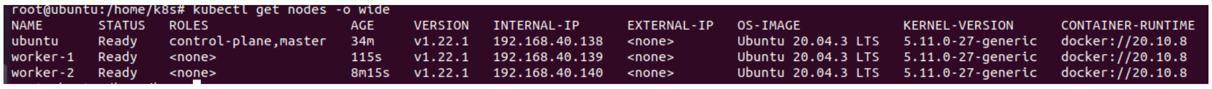


FIGURE 17 - TESTBED ARCHITECTURE

The VMs run Ubuntu 20.04.3 LTS, Kubernetes 1.22.1 and Docker 20.10.8.



```
root@ubuntu:/home/k8s# kubectl get nodes -o wide
NAME       STATUS   ROLES                  AGE     VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
ubuntu     Ready    control-plane,master   34m     v1.22.1   192.168.40.138   <none>        Ubuntu 20.04.3 LTS   5.11.0-27-generic   docker://20.10.8
worker-1   Ready    <none>                 115s    v1.22.1   192.168.40.139   <none>        Ubuntu 20.04.3 LTS   5.11.0-27-generic   docker://20.10.8
worker-2   Ready    <none>                 8m15s   v1.22.1   192.168.40.140   <none>        Ubuntu 20.04.3 LTS   5.11.0-27-generic   docker://20.10.8
```

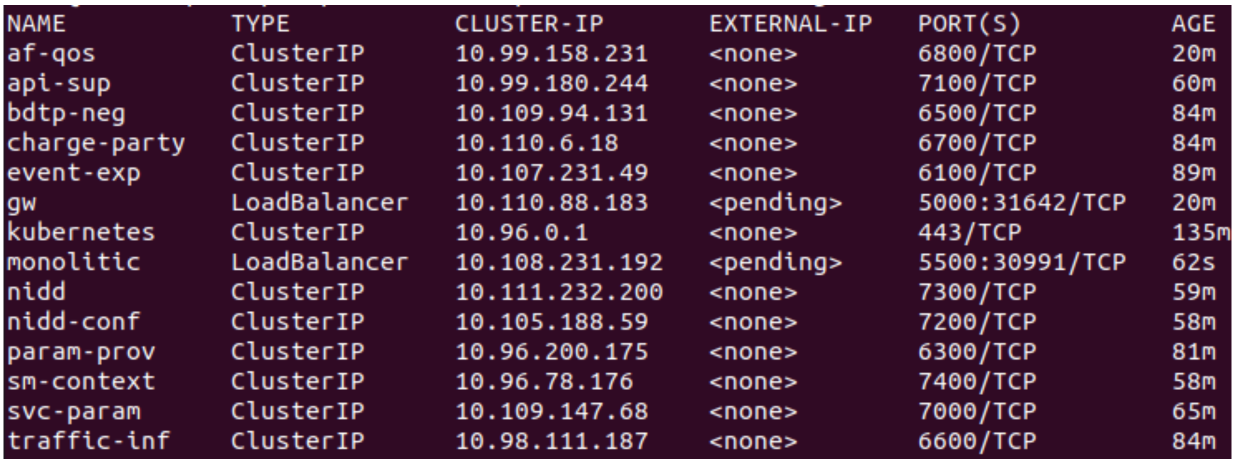
FIGURE 18 - KUBERNETES NODES

4.3 EXPERIMENTS AND RESULTS

In this section the experiments are described in detail. Furthermore, the experiment results are presented, and some conclusions are written.

4.3.1 EXPERIMENTS

For the experiment, eleven of the NEF services were deployed into the Kubernetes cluster (see Figure 19). Moreover, the NEF gateway and the Monolithic NEF were also deployed into the cluster.



```
NAME           TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
af-qos         ClusterIP      10.99.158.231    <none>        6800/TCP         20m
api-sup        ClusterIP      10.99.180.244    <none>        7100/TCP         60m
bdtp-neg       ClusterIP      10.109.94.131    <none>        6500/TCP         84m
charge-party   ClusterIP      10.110.6.18      <none>        6700/TCP         84m
event-exp      ClusterIP      10.107.231.49    <none>        6100/TCP         89m
gw             LoadBalancer   10.110.88.183    <pending>     5000:31642/TCP   20m
kubernetes     ClusterIP      10.96.0.1        <none>        443/TCP          135m
monolitic      LoadBalancer   10.108.231.192   <pending>     5500:30991/TCP   62s
nidd           ClusterIP      10.111.232.200   <none>        7300/TCP         59m
nidd-conf      ClusterIP      10.105.188.59    <none>        7200/TCP         58m
param-prov     ClusterIP      10.96.200.175    <none>        6300/TCP         81m
sm-context     ClusterIP      10.96.78.176     <none>        7400/TCP         58m
svc-param      ClusterIP      10.109.147.68    <none>        7000/TCP         65m
traffic-inf    ClusterIP      10.98.111.187    <none>        6600/TCP         84m
```

FIGURE 19 - NEF SERVICES DEPLOYED INTO THE KUBERNETES CLUSTER

The first group of experiments consisted in the execution of 20, 40 and 60 AF(s) in parallel as processes in the same VM for an interval of 120 seconds. The different number of consumers (AFs) allowed for the measurement of the system performance with different loads. On the experiments, was measured the time interval between the beginning of the request and the arrival of the response. The measurements were made at the AF(s) VM. Furthermore, the HTTP status code was also written to the log file. The experiments were made both for the NEF as Micro-Services as well as for the monolithic NEF, resulting into six datasets (M 20, MS 20, M 40, MS 40, M 60 and MS 60.

A second group of experiments was conducted. It addressed the capability of the micro service NEF to still provide a service when some of the micro-services are down, its availability [67]. The experiment was conducted by shutting down two micro services at a time. For each of the shutdown cycles, the 20 parallel AFs performed requests for 120s, resulting into 5 datasets (9, 7, 5, 3 and 1).

4.3.2 Results

With the experiments explained in detail, this section describes the results. The request-response cycle is measured, the progressive micro-service NEF degradation is studied and, finally, the implementation complexity of both architectural approaches is correlated with the code lines number of the implementation. Table 2 identifies and describes the datasets collected and studied.

TABLE 2 - DATASET DESCRIPTION

Dataset ID	Description

M 20	Monolithic Service NEF with requests from 20 parallel AFs
MS 20	Micro Service NEF with requests from 20 parallel AFs
M 40	Monolithic NEF with requests from 40 parallel AFs
MS 40	Micro Service NEF with requests from 40 parallel AFs
M 60	Monolithic NEF with requests from 60 parallel AFs
MS 60	Micro Service NEF with requests from 60 parallel AFs
9	Micro Service NEF, running 9 out of the 11 micro services, with requests from 20 parallel AFs
7	Micro Service NEF, running 7 out of the 11 micro services, with requests from 20 parallel AFs
5	Micro Service NEF, running 5 out of the 11 micro services, with requests from 20 parallel AFs
3	Micro Service NEF, running 3 out of the 11 micro services, with requests from 20 parallel AFs
1	Micro Service NEF, running 1 out of the 11 micro services, with requests from 20 parallel AFs

Request – Response Cycle Duration

The time elapsed between the request and the response is, in average, doubled in the micro service (MS) deployment when compared to the monolithic (M) deployment, as shown in Table 3. This observation is valid for the three load levels (20, 40 and 60 AFs). This result is explained with the additional communication step, between the gateway and the micro service.

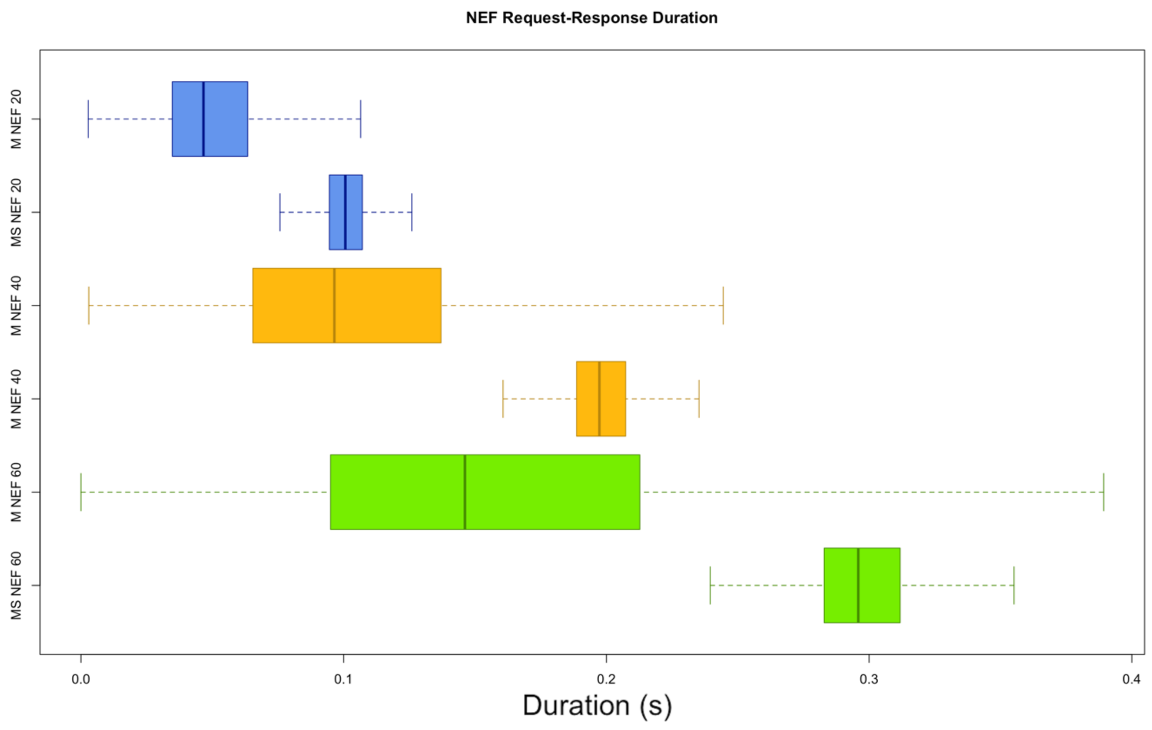


FIGURE 20 - GRAPHICAL REQUEST - RESPONSE DURATION VARIATION

Despite the previous observation, an important aspect visible in Figure 20 is the variation of the measured cycle duration. It is visible that the monolithic datasets present a bigger dispersion on the gathered values.

TABLE 3 - MEAN, STANDARD DEVIATION AND VARIANCE OF THE DATASETS

	MS 20	M 20	MS 40	M 40	MS 60	M 60
Mean (s)	0,1022	0,0524	0,2004	0,1056	0,2990	0,1578
Std. Dev. (s)	0,0174	0,0270	0,0267	0,0582	0,0415	0,0903
Variance (s)	0,0003	0,0007	0,0007	0,0034	0,0017	0,0081

By computing the standard deviation and the variance of the datasets in Table 3 and Figure 21, the numbers show that the monolithic NEF values are dispersed into a range, approximately, two times bigger than the micro-service one. This is explained by the capacity that the micro service deployment has to distribute the load among the independent services, unlike the monolithic implementation that needs to deal with all the load. It causes processing latency when overloaded with requests, resulting into bigger latency values.

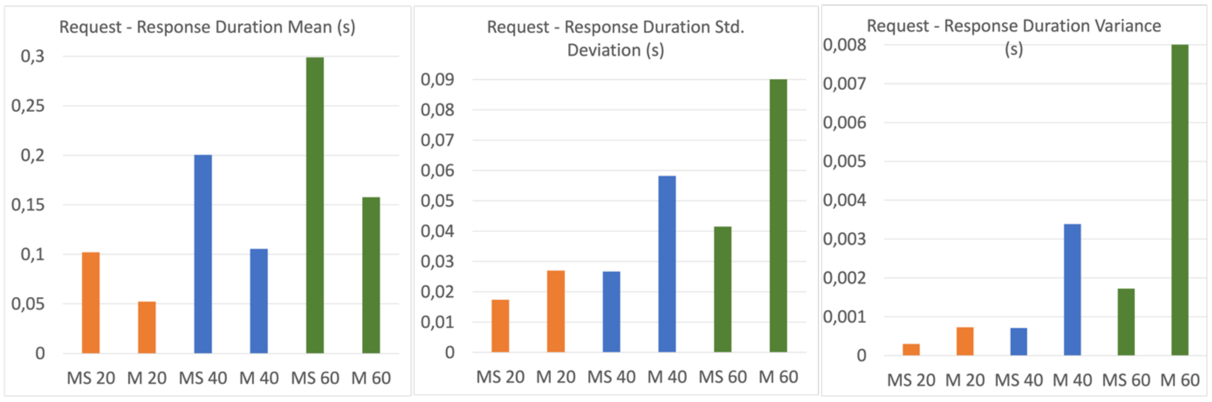


FIGURE 21 - MEAN, STANDARD DEVIATION AND VARIANCE

Regarding the capability of both approaches to deal with different load levels, the data shows that for the tested load levels, the duration of the request-response cycle increases proportionally. When the load is doubled (20 vs 40 AFs), the duration doubles in average. When there is an increment of 50% on the load (40 vs 60 AFs), the duration also increases 50% on average. On the other end, the dispersion of the data is much bigger for the monolithic NEF as the load increases. This shows that the micro-service NEF suits better services that require a smaller variation of the request-response cycle latency.

Service Availability

The capability to provide a degraded service when experience problems in some of the micro-services, is one of the advantages of that architectural approach. Compared to the monolithic implementation, there is not much to say, either it provides all services or none.

In order to measure the implemented micro-service NEF availability, the HTTP status code of each response was registered and divided into two categories, success codes and error codes. The success codes include the 2xx group (200, 201 and 204) and the error code includes the 500-error code.

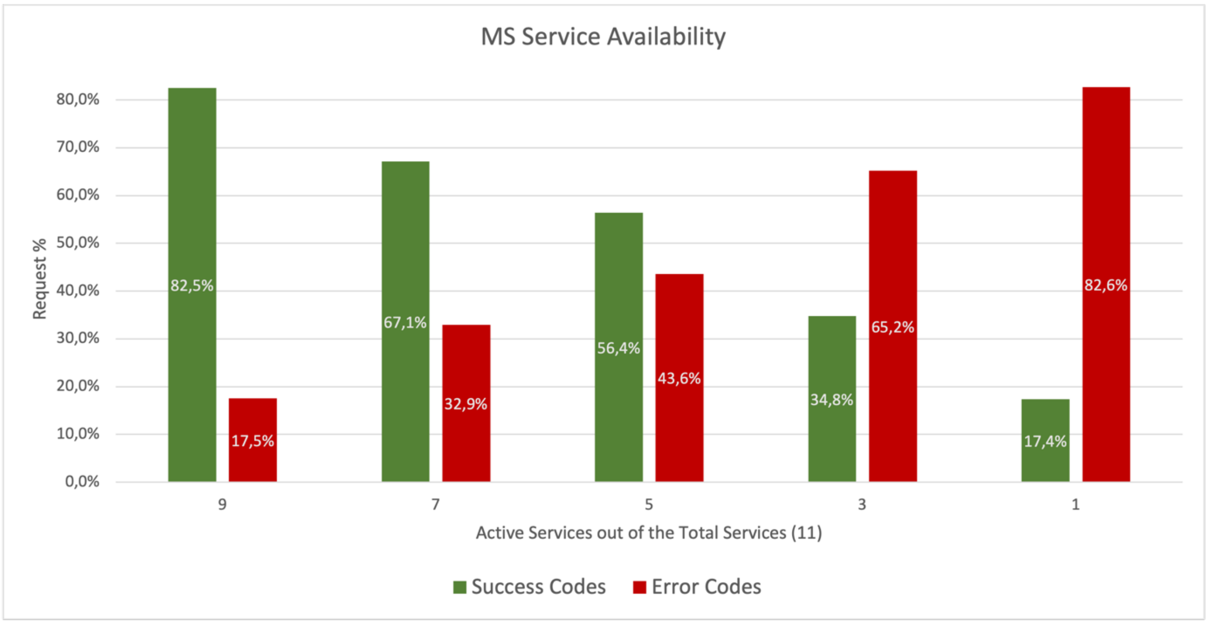
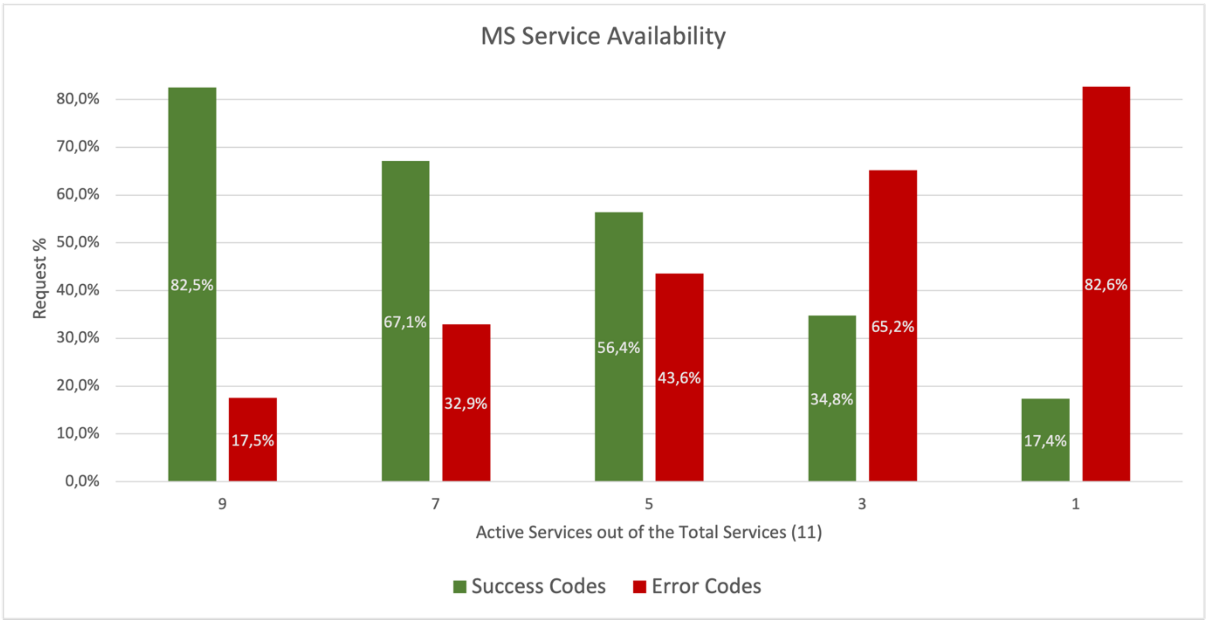


FIGURE 22 - MICRO SERVICE ARCHITECTURE SERVICE DEGRADATION

As shown in



, as the micro services are going down, the number of error codes increase, as expected. On the other end, it visible that, despite a degraded service, the architectural approach is still able to provide success responses. In order to maximize this micro service architectural characteristic, multiple replicas of the micro service can be run in parallel, impacting the response times, but avoiding the error codes.

Code Size

The complexity of an implementation can be measured by the number of lines needed to implement a particular functionality. One of the benefits of the micro-service architecture is that the micro services tend to be less complex that the monolithic implementation. By comparing the NEF implementations used to perform the experiments on this document, the monolithic NEF presents approximately 6 times more code lines that each of the micro services.

4.4 Discussion

From the experiments four main conclusions arise. When it comes to the duration of the request-response cycle, the monolithic NEF has on average half of the duration when comparing with the micro-service NEF. It was an expected result due to the network communication between the gateway and the micro service, whereas the value dispersion is much bigger for the monolithic NEF implementation. This behaviour is a result of the bottleneck that the monolithic NEF presents, when compared to the load distribution of the micro-services based approach. The experiment results, also, showed that the micro-service NEF has the capability to provide a degraded service when one or a group of micro services are unavailable. The monolithic NEF does not have this capability due to its nature. Finally, by comparing the number of code lines needed to implement the monolithic NEF vs the micro-service implementations, it becomes apparent to which extent the latter has an advantage. While this constitutes a rather crude code complexity metric, it provides some interesting insights regarding the existing differences between both implementations.

It is expected that by scaling the two NEF architectural approaches the latency results would improve. The gateway of the micro service NEF, despite not being a processing bottleneck, it would become a communication bottleneck. It would be interesting for future work to study the system performance with the presence of multiple instances of the micro-services and gateway as well as comparing the results with replicated monolithic NEFs. However, in order to do that it is important to understand the more suitable load balancing policy between the different instances.

5 PROJECT MANAGEMENT

This chapter describes the first and second semester activities, regarding the work plan of the Master Thesis presented in this document.

The first section presents the work plan for the first semester and the effective work conducted. The second section describes the proposed work plan concerning the second semester and the real work plan performed.

5.1 FIRST SEMESTER

Figure 23 and Table 4 illustrate the initial proposed plan for the first semester. Since the predicted tasks experienced delays and changes, the progress of the project took a different path from the initial plan. Therefore, the real schedule of the work plan is presented at Table 5 and Figure 24.

Description	**Start Date**	**End Date**	**Duration (Days)**
T1.1- State of the art and requirements analyses	21/09/2020	22/11/2020	62
T1.1.1 - Studying 5G Topology/Components	21/09/2020	15/10/2020	24
T1.1.2 - Infrastructure Providers	15/10/2020	02/11/2020	18
T1.1.2.1 - AWS CloudFormation	15/10/2020	24/10/2020	9
T1.1.2.2 - Docker	25/10/2020	02/11/2020	8
T1.1.3 - Orchestration Tools	03/11/2020	18/11/2020	15
T1.1.3.1 - Matilda Vertical Applications Orchestrator	03/11/2020	08/11/2020	5
T1.1.3.2 - ONAP	09/11/2020	18/11/2020	9
T1.1.4 - 5GLAN - Studying Releases/Papers	19/11/2020	22/11/2020	3
T1.2- Initial design of the proposed solution	01/01/2021	09/01/2021	8
T1.2.1 - 5GLAN - Proposed Solution Analyses and Design	01/01/2021	05/01/2021	4
T1.2.2 - Orchestration - Proposed Solution Analyses and Design	06/01/2021	09/01/2021	3
T1.3 - FUDGE-5G Use Cases	01/10/2020	20/12/2020	80
T1.4 - Writing the report	04/12/2020	19/01/2021	46

TABLE 4 – FIRST PLANNED SEMESTER SCHEDULE

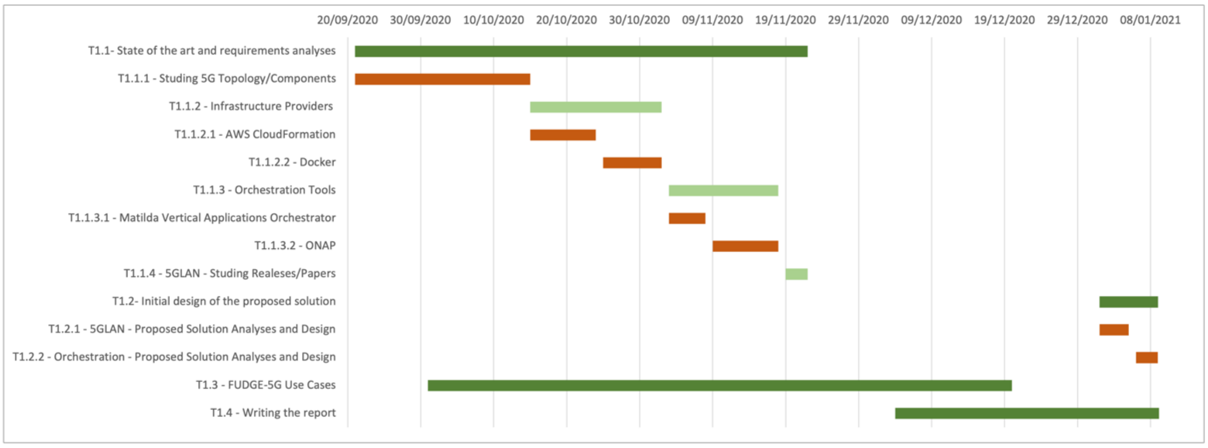


FIGURE 23 – FIRST SEMESTER PLANNING CHART

Description	Start Date	End Date	Duration (Days)
T1.1- State of the art and requirements analyses	21/09/2020	04/12/2020	74
T1.1.1 - Studying 5G Topology/Components	21/09/2020	15/10/2020	24
T1.1.2 - Infrastructure Providers	15/10/2020	02/11/2020	18
T1.1.2.1 - AWS CloudFormation	15/10/2020	18/10/2020	3
T1.1.2.2 - Docker	19/10/2020	22/10/2020	3
T1.1.2.3 - OpenStack	23/10/2020	02/11/2020	10
T1.1.3 - Orchestration Tools	03/11/2020	28/11/2020	25
T1.1.3.1 - Matilda Vertical Applications Orchestrator	03/11/2020	06/11/2020	3
T1.1.3.2 - ONAP	07/11/2020	17/11/2020	10
T1.1.2.5 - Open-Source Mano	18/11/2020	28/11/2020	10
T1.1.4 - 5GLAN - studying Releases/Papers	01/12/2020	04/12/2020	3
T1.2 - FUDGE-5G Use Cases	01/10/2020	20/12/2020	80
T1.3 - Writing the report	04/12/2020	19/01/2021	46

TABLE 5 – FIRST SEMESTER EFECTIVE SCHEDULE

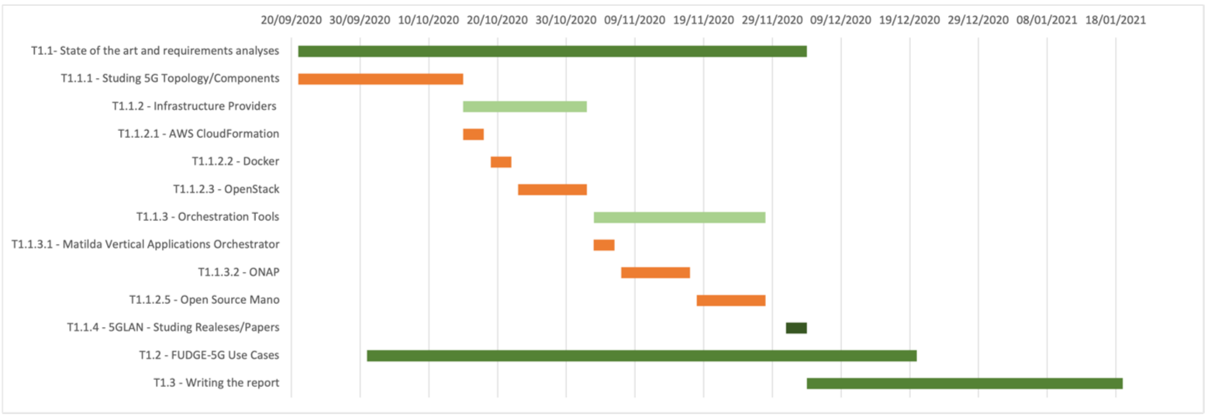


FIGURE 24 – FIRST SEMESTER EFECTIVE SCHEDULE CHART

The first performed task was the studying of the 5GC topology and components (NFs), with focus on the NEF. It was a time-consuming task due to the novelty of the content to the author. The following task the study of the cloud infrastructure providers AWS, OpenStack and Docker, which was followed by studying the VNF orchestration tools ONAP and OSM, a vertical application orchestrator Matilda and finally Kubernetes, a containerised application orchestrator.

5.2 Second Semester

The plan proposed for the second semester is presented in Table 6 and Figure 25. Like in the first semester, adjustments were made to the schedule. Some tasks suffered major delays impacting the following ones. Figure 26 and Table 7 showcase the effective schedule.

TABLE 6 – SECOND SEMESTER PLANNED SCHEDULE

Description	Start date	End date	Duration (days)
T2.1- Identification and Desintegration of the NEF services	20/01/2021	18/05/2021	118
T2.1.1 - Identification of the NEF Services and its requirements	20/01/2021	25/01/2021	5
T2.1.2 - NEF Services implementation	26/01/2021	01/04/2021	65
T2.1.3 - Service Containerization	02/04/2021	20/04/2021	18
T2.1.4 - Kubernetes PodSpecs creation	21/04/2021	18/05/2021	27
T2.2 - Test Bed instalation and Integration tests	19/05/2021	05/06/2021	17
T2.2.1 - Test Bed instalation	19/05/2021	21/05/2021	2
T2.2.2 - Kubernetes micro-services deployment	22/05/2021	25/05/2021	3
T2.2.4 - Test the integration of the micro services with the NEF gateway	25/05/2021	26/05/2021	1
T2.2.5 - Test the integration of the micro services with the NEF consumers	27/05/2021	01/06/2021	5
T2.3 Experiment and Results	02/06/2021	16/06/2021	14
T2.4- Writing internship report.	01/05/2021	30/06/2021	60

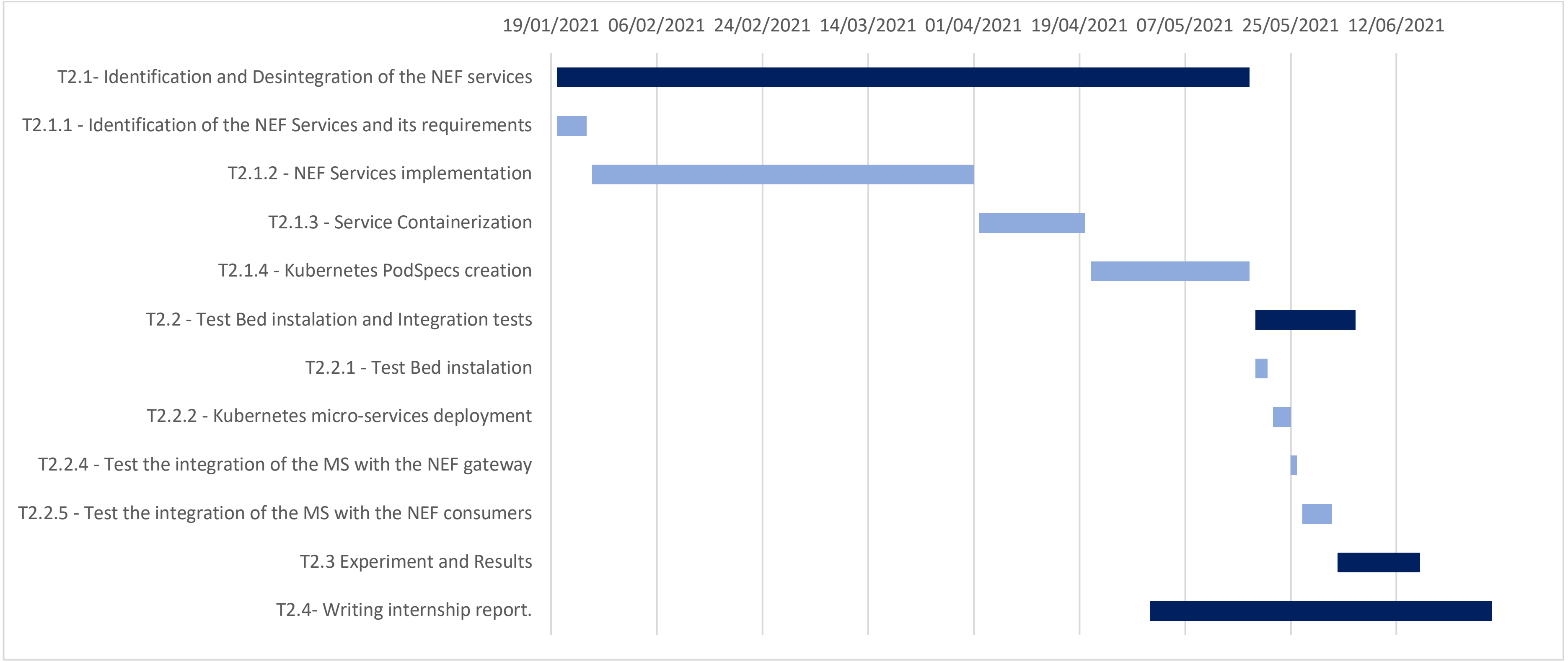


FIGURE 25 – SECOND SEMESTER PLANNED CHART

Description	Start date	End date	Duration (days)
T2.1- Identification and Desintegration of the NEF services	20/01/2021	28/05/2021	128
T2.1.1 - Identification of the NEF Services and its requirements	20/01/2021	25/01/2021	5
T2.1.2 - NEF Services implementation	26/01/2021	15/04/2021	79
T2.1.3 - Service Containerization	16/04/2021	30/04/2021	14
T2.1.4 - Kubernetes PodSpecs creation	01/05/2021	28/05/2021	27
T2.2 - Test Bed instalation and Integration tests	28/05/2021	15/07/2021	48
T2.2.1 - Test Bed instalation	28/05/2021	06/06/2021	9
T2.2.2 - Kubernetes micro-services deployment	07/06/2021	12/06/2021	5
T2.2.4 - Test the integration of the MS with the NEF gateway	13/06/2021	30/06/2021	17
T2.2.5 - Test the integration of the MS with the NEF consumers	01/07/2021	15/07/2021	14
T2.3 Experiment and Results	16/07/2021	31/07/2021	15
T2.4- Writing internship report	01/07/2021	30/08/2021	60

TABLE 7 - SECOND SEMESTER EFFECTIVE SCHEDULE

The second semester had the majority of the time dedicated to the NEF and its disintegration. The disintegration of the NEF services took 128 days to perform, 10 more days that the initial plan.

A major delay on task 2.2 was the reason for the rescheduling of the report deliver from June 30th to September 7th. The delay was caused by complexity of the Kubernetes cluster inter-service communication. This affected the integration tests and delayed the experiment.

After the integration of the NEF services with the respective consumers, and gateway, the experiment took place. The next step was to process the collected datasets to obtain the experiment results.

Finally, the remaining task was the report writing. It included the modifications from the midterm review meeting and the new content regarding the practical work.

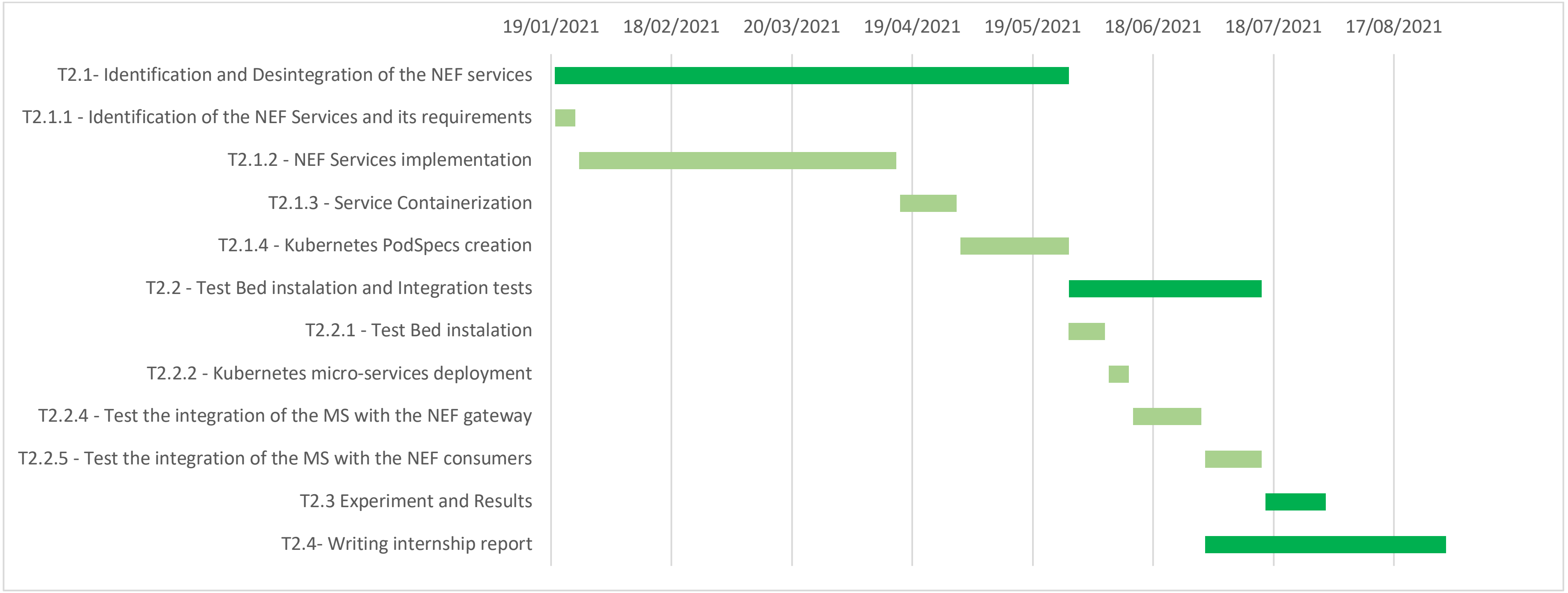


FIGURE 26 - SECOND SEMESTER EFFECTIVE SCHEDULE

6 CONCLUSIONS

With the 5th Generation of mobile networks major improvements in bandwidth, latency and equipment density per km^2 will be broth to the companies and end-users of this new technology. However, it is fundamental to the telco providers, to virtualize the network and thus provide 5G in a costly-efficient and scalable way.

3GPP standardized the 5G Core network functions and its APIS. The standardization allows for interoperability and to network operators and application developers to develop the 5G network following the standards. As the standardization evolves, the number of services provided for each of the 5G Core network functions is growing. Despite the standardization, the 5G Core network functions are becoming an heterogenous mesh of services with different requirements. Thus, the implementation of the 5G Core NFs is resulting into monolithic entities with a huge number of code lines. The consequence is that the 5GC NFs lifecycle management is becoming complex, when dealing with maintenance, updates and scaling. To solve this problem, FUDGE-5G proposes the disintegration of the 5G Core network functions into micro services.

Concerning the state of the art for this work, it was possible to identify, that the tools for the orchestration and management of those micro services already exist. Tools like Docker and Kubernetes. The former, provides a framework to packaging all dependencies and code of an application into a container that is agnostic to the operative system. The ladder provides a system for deployment, scaling and management of containerized applications. By using both tools, it is possible to decompose the 5GC NFs into a set of smaller services and orchestrate them in a way that the NF provides the expected service in a transparent way for the consumer.

The way how the services should be separated can follow different criteria. Services that have traffic peaks can be isolated and scaled, services that consume a lot of processor and memory too. The NEF decomposition into micro services, presented in this document, was made by splitting the 3GPP standardized NEF services into independent units. The criterion was chosen in order to maximize the NEF service availability.

By disintegrating the 5GC NFs, following the micro service approach, the 5G Core becomes a loosely couple and scalable system, making the 5G mobile telecommunications network more resilient and easier to manage.

7 BIBLIOGRAPHY

[1] “FUDGE-5G,” [Online]. Available: https://fudge-5g.eu/.

[2] “OneSource,” [Online]. Available: https://onesource.pt/#customizedSystems. [Accessed 2021].

[3] “mobitrust,” [Online]. Available: https://mobitrust.onesource.pt.

[4] “SCRUM,” [Online]. Available: https://www.scrum.org. [Accessed 2021].

[5] “5G New Radio,” [Online]. Available: https://www.electronics-notes.com/articles/connectivity/5g-mobile-wireless-cellular/5g-nr-new-radio.php. [Accessed May 2021].

[6] “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 16),” 3GPP, 2020.

[7] 3GPP, “3GPP TS 23.502 V16.7.1,” 3GPP, 2021.

[8] ETSI, “ETSI TS 129 554”.

[9] NGMN Alliance, “5G White paper, V.1.0,” February 2015. [Online]. Available: https://www.ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf. [Accessed 6 January 2020].

[10] 3GPP, “3GPP TS 22.261 – Section 6.25”.

[11] “NPNs – State of the art and way forward,” 5G PPP Technology Board, To be published.

[12] Trossen, Robitzsch, Reed, Al-Naday and Riihijarvi, “Internet Services over ICN in 5G LAN Environments draft-trossen-icnrg-internet-icn-5glan-03,” 1 October 2020. [Online]. Available: https://tools.ietf.org/html/draft-trossen-icnrg-internet-icn-5glan-03#page-21. [Accessed 6 January 2021].

[13] 3. G. P. Project, “Technical Specification Group Services and System Aspects; Service requirements for the 5G system; TS 22.261 V18.1.0; Stage 1 (Release 18),” 2020.

[14] M. J. R. J. R. M. G. N. F. G. X. 1. E. 2. o. E. 3. A. U. 4. 5. U. o. E. &. B. Dirk Trossen1, “IP Over ICN - The Better IP?,” in *2015 European Conference on Networks and Communications (EuCNC)*, Paris, France, 2015.

[15] "Amazon EC2 Instance Types," 2020. [Online]. Available: https://aws.amazon.com/ec2/instance-types/. [Accessed 24 12 2020].

[16] "What is Amazon S3?," 2020. [Online]. Available: https://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html. [Accessed 24 12 2020].

[17] "Amazon Route 53," 2020. [Online]. Available: https://aws.amazon.com/route53/. [Accessed 24 12 2020].

[18] "Choosing a routing policy," 2020. [Online]. Available: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html. [Accessed 24 12 2020].

[19] "Elastic Load Balancing," 2020. [Online]. Available: https://aws.amazon.com/elasticloadbalancing/?elb-whats-new.sort-by=item.additionalFields.postDateTime&elb-whats-new.sort-order=desc. [Accessed 24 12 2020].

[20] "AWS CloudFormation concepts," 2020. [Online]. Available: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-whatis-concepts.html. [Accessed 25 12 2020].

[21] "AWS CloudFormation template formats," [Online]. Available: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-formats.html. [Accessed 07 01 2021].

[22] "OpenStack Sotware," [Online]. Available: https://www.openstack.org/software/. [Accessed 10 01 2021].

[23] "OpenStack Services," [Online]. Available: https://www.openstack.org/software/project-navigator/openstack-components/#openstack-services. [Accessed 10 01 2021].

[24] "Launch an instance," [Online]. Available: https://docs.openstack.org/heat/latest/install/launch-instance.html#create-a-stack. [Accessed 10 01 2020].

[25] "VMWare," [Online]. Available: https://www.vmware.com. [Accessed 2021].

[26] "What is Kubernetes?," 22 10 2020. [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/. [Accessed 23 12 2020].

[27] RED HAT, "Understanding virtualization," 2020. [Online]. Available: https://www.redhat.com/en/topics/virtualization. [Accessed 23 12 2020].

[28] J. Shamir, “5 Benefits of Virtualization,” IBM, 2020. [Online]. Available: https://www.ibm.com/cloud/blog/5-benefits-of-virtualization. [Accessed 23 12 2020].

[29] “CONTAINERS AT GOOGLE,” 2020. [Online]. Available: https://cloud.google.com/containers. [Accessed 23 12 2020].

[30] “Docker overview,” 2020. [Online]. Available: https://docs.docker.com/get-started/overview/. [Accessed 23 12 2020].

[31] “Docker overview,” 2020. [Online]. Available: https://docs.docker.com/get-started/overview/#docker-architecture. [Accessed 24 12 2020].

[32] M. Project, “5G-Ready Vertical Applications Orchestration - Whitepaper,” 2020. [Online]. Available: https://t.co/NgfztmC2Ya?amp=1. [Accessed 24 12 2020].

[33] “ONAP,” [Online]. Available: https://www.onap.org. [Accessed 2021].

[34] “AAF - Application Authorization Framework,” [Online]. Available: https://docs.onap.org/projects/onap-aaf-authz/en/latest/index.html#master-index. [Accessed 2021].

[35] “AAI Documentation Repository,” [Online]. Available: https://docs.onap.org/projects/onap-aai-aai-common/en/honolulu/index.html#master-index. [Accessed 2021].

[36] “APPC LCM API Guide,” [Online]. Available: https://docs.onap.org/projects/onap-appc/en/frankfurt/APPC-LCM-API-Guide/APPC-LCM-API-Guide.html#introduction. [Accessed 2021].

[37] “DCAE Architecture,” [Online]. Available: https://docs.onap.org/projects/onap-dcaegen2/en/honolulu/sections/architecture.html. [Accessed 2021].

[38] “ONAP MultiCloud Architecture,” [Online]. Available: https://docs.onap.org/projects/onap-multicloud-framework/en/honolulu/MultiCloud-Architecture.html#value-proposition. [Accessed 2021].

[39] “OSM Quickstart,” [Online]. Available: https://osm.etsi.org/docs/user-guide/01-quickstart.html. [Accessed 11 01 2020].

[40] [Online]. Available: https://osm.etsi.org/docs/vnf-onboarding-guidelines/00-introduction.html. [Accessed 01 2020].

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

[42] “kubelet,” [Online]. Available: https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/. [Accessed 2021].

[43] “kube-proxy,” [Online]. Available: https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/. [Accessed 2021].

[44] “kube-apiserver,” [Online]. Available: https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/. [Accessed 2021].

[45] “kube-scheduler,” [Online]. Available: https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/. [Accessed 2021].

[46] “Kubeadm,” [Online]. Available: https://kubernetes.io/docs/reference/setup-tools/kubeadm/. [Accessed 2021].

[47] kubelet. [Online]. Available: https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/.

[48] “Overview of kubectl,” [Online]. Available: https://kubernetes.io/docs/reference/kubectl/overview/.

[49] R. Hat, “What are microservices?,” [Online]. Available: https://www.redhat.com/en/topics/microservices/what-are-microservices. [Accessed May 2021].

[50] A. Vieira, “What is a Microservice?,” [Online]. Available: https://www.outsystems.com/blog/posts/what-is-a-microservice/. [Accessed June 2021].

[51] 3GPP, “TS 23.502 V16.8.0 section 5,” 2021, pp. 458-565.

[52] “FUDGE-5G consortium,” [Online]. Available: https://fudge-5g.eu/en/consortium. [Accessed June 2021].

[53] “5G-VINNI,” [Online]. Available: https://www.5g-vinni.eu. [Accessed 2021].

[54] FUDGE-5G, “ Technical Blueprint for Vertical Use Cases and Validation Framework,” [Online]. Available: https://fudge-5g.eu/download-file/365/sq6G3zIXkRBOFWRM3bqO.

[55] [Online]. Available: https://www.eduroam.org.

[56] “FUDGE-5G: Fully Disintegrated Private Networks for 5G Verticals,” [Online]. Available: https://zenodo.org/record/5137741#.YP696C1Q1QI. [Accessed 2021].

[57] “FUDGE-5G: Fully Disintegrated Private Networks for 5G Verticals,” [Online]. Available: https://zenodo.org/record/5139613#.YP_k8y1Q1QI. [Accessed 2021].

[58] “2021 Joint EuCNC & 6G Summit,” [Online]. Available: https://www.eucnc.eu.

[59] “EuCNC 2021 FUDGE-5G poster presentation,” [Online]. Available: https://www.youtube.com/watch?v=9FERHWE_NSs&t=15s. [Accessed 2021].

[60] “Mobitrust,” [Online]. Available: https://mobitrust.onesource.pt. [Accessed 2021].

[61] FUDGE-5G, “Technical Blueprint for Vertical Use Cases and Validation Framework,” 2021.

[62] M. Richards, “The Challenges of Service-Based Architecture,” 2015. [Online].

[63] [Online]. Available: https://5go.pt/projeto/. [Accessed June 2021].

[64] L. Samsung Electronics Co., “Cloud Native 5G Core,” 2020. [Online]. Available: https://images.samsung.com/is/content/samsung/p5/global/business/networks/insights/white-paper/cloud-native-5g-core/Samsung-5G-Core-Vol-2-Cloud-Native-5G-Core.pdf.

[65] PyPI, “Quart,” [Online]. Available: https://pypi.org/project/Quart/. [Accessed 2021].

[66] PyPI, “Requests,” [Online]. Available: https://pypi.org/project/requests/. [Accessed 2021].

[67] J.-C. L. B. R. C. L. Algirdas Avizienis, “Basic Concepts and Taxonomy of Dependable and Secure Computing,” in *IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING*, 2004, pp. 11-33.

[68] [Online]. Available: https://prometheus.io. [Accessed 25 12 2020].

[69] [Online]. Available: https://qmon.io. [Accessed 25 12 2020].

[70] [Online]. Available: https://www.netdata.cloud. [Accessed 25 12 2020].

[71] P. Guntermann, “What is Water-Scrum-Fall?,” 19 10 2017. [Online]. Available: https://www.letsmakebettersoftware.com/2017/10/what-is-water-scrum-fall.html. [Accessed 16 01 2021].

[72] “How to Make Agile and Waterfall Methodologies Work Together,” [Online]. Available: https://reqtest.com/agile-blog/agile-waterfall-hybrid-methodology-2/. [Accessed 16 01 2021].