ACIT5930

MASTER'S THESIS phase III

in

Applied Computer and Information Technology (ACIT)

May 2023

Cloud Services and Operations

Data Collection for Machine Learning in 5G Mobile Networks

Furqan Ahmad

Department of Computer Science

Faculty of Technology, Art, and Design

OSLOMET

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In the modern world, humans are surrounded with machines, computers, sophisticated communication systems, digital and smart devices, etc. In the past, machines and devices were completely relying on human input to perform certain task as it was solely human responsibility to achieve certain program from machines and devices, but this usage and interaction has increased with time and it has resulted into exponential increase in communication requirements from machine to machine, as well as human to machine and machine to human over a network (See, 2021). One of the pinnacles of human nature is to offload as many tasks as possible to other means that can provide automated processes. For example, autonomous car driving or smart industries to operate automatically, smart houses to adapt to needs and requirements of users, and many more. This growth of human dependency on machines and devices demands reliable, robust, resilient communication networks to perform the task with high accuracy. The demand for high-performance communications requires the existing networks to develop and adapt accordingly, so the said requirements can be fulfilled.

The fifth generation (5G) is the next generation network of the Third Generation Partnership Project (3GPP), which has evolved to address the increasing demand of connectivity across the wireless mobile and fixed lines, emerging smart infrastructure, production automation, next-generation transportation, exploration and mining industry, education, utilities, broadcasting, and media industry, etc. (ETSI, 2015). Consequently, the 5G networks should take into consideration the following objectives:

- Faster upload and download speeds,
- Reliable and secure communication across different devices,
- High-throughput and low-latency communication for mission critical use cases,
- Support of massive number of devices which are located inside small zones, like the Internet of Things (IoT),
- Steady and seamless streaming of high-quality online content,
- High-definition quality of video and voice calls,
- Support for virtual and augmented reality (VR/AR),
- Integration with smart healthcare systems and telesurgery.

The 5G networks, compared to the previous generation mobile networks such as fourth generation (4G), third generation (3G) and second generation (2G), are revolutionary in the communication industry in terms of opening additional horizons to various verticals to be part of the interconnected ecosystem, hence the vision of a single network to facilitate communication of multiple verticals without interruption via introduction of the network slicing concept, has become a reality. The International Telecommunication Union Radiocommunication (ITU-R) has defined three main use cases for 5G, based on the type of communication requirements for ease of market adaptation. Additionally, the flexibility to use the dedicated slice for each use case within a single network is provided. The three main use cases of 5G are as follows:

- **Enhanced Mobile Broadband (eMBB),** the use case responsible for deployments with high user density and increased demand for high data rates, seamless coverage, as well as advanced mobility setups along with remarkably high traffic capacity for hotspot scenarios.
- **Massive Machine-type Communications (mMTC),** as the name suggests, this use case applies to the Internet of Things (IoT), which involves massive number of connected devices with requirement for low power consumption and low data rates.
- **Ultra-reliable and Low Latency Communications (URLLC)** to accommodate use case for safety and mission critical applications that need the shortest possible latency and high reliability in terms of connection (i.e., autonomous transportation or flight).

1.1 Motivation

5G supports heterogeneous traffic to ensure the communication of devices, services, and various other consumer-related use-cases. The heterogeneous traffic includes consumer traffic from smart devices (eMBB use cases), traffic from laptops, machine to machine traffic (IoT use cases), traffic from mission critical application (URLCC use cases). Therefore, the demand in terms of use cases and traffic is growing rapidly (ERICSSON, 2023). The increased traffic across devices requires 5G to be built on principles in an integrated service-centric frameworks, which is critical for highly granular scalability, elasticity, dynamic control, and orchestration to support high bandwidth, massive IoT connectivity, and ultra-low latency applications. One of the crucial enhancements in 5G is the segregation of the Control and User Plane. This separation enables flexible network deployment, operation and independent scaling of the control and user plane depending on actual traffic demands, ensuring that the network is highly flexible and highly programmable. This shall enable the most effective use of key technologies such as Network Functions Virtualization (NFV) and Software-Defined Network (SDN), which are key features for facilitating the distribution of services to the edge for low latency applications. Additionally, the web-scaling architecture adopted in 5G network is designed to deliver scale, performance, and service agility.

Some of the key objectives of 5G network are:

- **Heterogeneous support** for devices, services, and consumers with a varied set of functional and performance requirements,

- **Multi-Access Edge Computing (MEC)**: Most essential to achieve the low-latency related requirements of 5G. Low-latency and high throughput applications require placing network functions closer to the edge. This also offloads the traffic to the edge of the network,

- **Access Agnostic Core**: A common core architecture that supports all types of access (3GPP, non-3GPP), allowing for seamless interworking between all types of networks and enabling operational efficiencies,

- **Cloud-Native Infrastructure**: Uses containerized cloud-native infrastructure practices with open-source components from the Cloud-Native Computing Foundation (CNCF): Docker (DOCKER.INC, 2022), Kubernetes (KUBERNETES, 2023) , Istio, Envoy, Prometheus deployments enabling, shorter startup time due to lower overhead and statelessness, etc.,

- **Continuous Development & Integration (CI/CD)**: Adopts native DevOps (Development and Operations) practices for software release and upgrade cycles, this enhances the automated deployment, integration and testing process compared to previous approaches,
- **Reducing Risk**: Supports Kubernetes canary deployment model to continuously evaluate new releases in production, while reducing risk of outage and minimizing downtime,
- **Service Agility**: Introduces new services efficiently and expeditiously on-demand,
- **Network Slicing**: Ability to customize the network to specific needs of consumers and industry verticals. Provides traffic isolation and security as well as specifically tailored Quality of Service for end users or verticals,
- **Orchestration and Automation**: Operational efficiency and network scalability to serve large numbers of devices with reduced operational expenditure (OPEX) by extensive use of automation, driven by Artificial Intelligence and Machine Learning (AI/ML).

Orchestration and automation are crucial to achieve the operational efficiency for 5G scalability to serve substantial number of devices at different network slices. Consequently, the main objective of this thesis is to design a framework that supports the identification of monitoring points with associated network functions in 5G, for the purpose of capturing the desired information in form of raw data at defined regular intervals. An AI/ML system can further analyze the data to deliver analytics functionality in terms of identifying anomalies for the purpose of securing the core networks or for self-optimizing network purposes in cases when the utilization conditions are dynamically changing. In such a way, the raw data can be utilized to provide automation for the sake of autonomous scalability, increasing the operational efficiency, and minimizing the need for human intervention in case optimizations and re-configuration is required.

1.2 Problem statement

In the world of service delivery, an efficient and scalable monitoring system is mandatory for the continuity of services and identification and troubleshooting of various incidents. 5G core networks deployed in complex models such as cloud-native or hybrid infrastructures, supported by orchestration and automation capabilities, pose a challenge for the legacy monitoring systems to operate and function as expected due to increased performance scaling and architectural demands. This requires a novel approach that can provide isolation of the collected data and multi-tenancy support for each network slice separately, by collecting relevant data from the core network functions based on the principle of scalability so that the monitoring system can adapt to any number of existing or newly provisioned network slices and their associated core functions.

In this thesis, one of the main objectives is to identify the limitations of legacy monitoring systems and design a new monitoring framework for 5G networks based on cloud-native principles. 5G predicts two functions for this case, referred to as NWDAF (Network Data Analytics Function) and UDSF (Unstructured Data Storage Function). The UDSF should be designed in such a way that it will be able to collect and store unstructured data from the network control plane for the sake of using it from an optimization perspective, based on the AI/ML functionality provided in the NWDAF function. Some of the features should be as follows:

a. Dynamic architecture support, which forms a flexible structure to enable the monitoring via variable monitoring solution that can re-configure and scale accordingly.
b. Provisioning the data collection functions as a service. Telco operators should be provided with the possibility to choose what kind of data should be collected, as well as how to incorporate this data for different use cases (cybersecurity, system optimization, dynamic network control, etc.)
c. Capturing data at network functions from relevant protocols in a technology-agnostic manner.
d. Data collection from control and user plane for enabling machine learning and artificial intelligence analytics. Data can be collected at the control plane or user plane. Data originating from the user plane is the information that flows from specific users through the 5G network like application data. The control plane data is what the system transmits for the sake of enabling connectivity using various protocols at different network layers.
e. Data aggregation and presentation with the support of multi-tenancy in a cloud-native environment.
f. Data integrity and privacy to ensure confidentiality via data isolation and anonymization in case of monitoring multiple network slices served by the same 5G network.
g. Data should be anonymized to protect the user privacy while enabling network analytics based on the collected information.

The final goal of the thesis is to implement the data collection framework in a cloudified 5G facility of the Oslo Metropolitan University, identify the monitoring points for the 5G core network functions and propose a solution for metrics associated to detect and evaluate anomalies in the traffic within the core and between network slices. This shall consequently provide better insight into the security of the 5G core and pave the way for designing mitigation methods for various cyberattacks on 5G core networks as well as conditions for self-organizing network capabilities.

1.3 Thesis Organization

The thesis is organized as follows:

- Chapter 1: Introduction that provides an overview of the 5G requirements and motivation for this thesis work.

- Chapter 2: Background, which covers the communication framework, model, and architecture on which the thesis research and related work has been built upon. The chapter elucidates the necessary information for the used technologies to better understand the research in question.
- Chapter 3: Methodology that explains the details on the practices followed in this thesis for designing the data collection solution.
- Chapter 4: Architecture of the 5G networks that explains the 5G system, as well as the relevant core network functions that are mandatory to understand the research work in the thesis.
- Chapter 5: Establishing a virtualized 5G Network that encompasses the details of the 5G Core (5GC) network deployment in controlled environment in an OpenStack cloud at the Metropolitan University with available open-source tools and technologies.
- Chapter 6: Data generation in 5GC network that covers the metrics identification and evaluation at interfaces and protocols level for each 5GC network function to probe and gather insight on the core network. Data capturing in a 5GC network that covers the capturing of the data from a network function and methods to capture the data.
- Chapter 7: Data collection Solution for 5G Networks: this chapter covers the end-to-end solution to process and clean the captured data. The chapter also shows the result of storing the data and making it available for further machine learning purposes.
- Chapter 8: Discussion on the 5G architecture, challenges encountered during the implementation and experimentation.
- Chapter 9: Conclusion includes the final discussion on the work and future research opportunities to enhance this work and provide a roadmap for building upon the solution to incorporate AI/ML based analytics function.

2 BACKGROUND

This chapter covers the essential traits of telecommunication technologies which strive to establish interaction across various network technologies. The Open System Interconnection (OSI) model has been discussed as a guiding principle of communications due to its pertinence of standard architecture across a broad spectrum of industry-grade applications. Thus, all communication protocols can be analogized and compared via the OSI model as a common reference point. The OSI model introduces the essential concept of layered architecture to dissect any protocol. The layered architecture helps to understand, compare, and analyze any industry protocol belonging to any commercial, public, or private network. As represented in the further chapters, the OSI layered architecture has been used as framework architecture to analyze the 5G-related network functions and corresponding protocols to pave the path for describing the monitoring and data capturing solution for 5G networks.

Moreover, the chapter encompasses the details of available open source-based solutions which are being used for designing and architecting the monitoring and capturing framework. Since we discuss the deployment of a 5G core network based on cloud-centric approach, the main concept shall revolve around service-oriented architectures. Consequently, the management and orchestration will be based on the best practices of cloud native principles that involve virtualization and container technologies for providing a pathway towards dynamic management, granular monitoring, and data collection as well as reconfigurable networks and analytics. Conclusively, this chapter will cover the following areas:

- The OSI model and its architecture.
- Brief description of the TCP/IP model.
- Existing data capturing in Virtualized and containerized environment.
- Apache Kafka architecture.

2.1 OSI Model and Functionality

Networks emerge from software, hardware, and integration of various hardware devices such as storage devices, input/output devices, operating systems, communication devices, and networking devices. Network operations require communication standards to promote interoperability, scalability, and generality across the network elements. Depending on the role of devices within the network, such communication task based on role, becomes complex and upsurges the need for multiple communication standards; hence common communication architecture, along with a framework, should overcome such complication. In 1977, this led the International Organization for Standardization (ISO) to establish a subcommittee to develop such an architecture. The result was the Open Systems Interconnection (OSI) reference model. (Alani, 2014)

Open System Interconnection (OSI) came into existence to address the communication challenges in the late seventies between different network elements when interoperability among vendor equipment was not standardized, and the industry was exploring a solution to solve this problem. The OSI protocol and model provides a theoretical template which serves as an abstract framework for various protocols to function on ideal hardware. Initially, the telecommunication industry's work on signaling protocols has not progressed within a standard framework, and this resulted in closed box individual protocols which were not well aligned to industry standards. At the same time, this approach encouraged the environment-based protocols, which led to interworking difficulties. Later, Subsystem No 7 evolved based on Open Systems Interconnection (OSI) Reference Model, but now all the modern-day network protocols in telecommunication networks use OSI as the reference model. (ITU-T, 2014)

The OSI model provides network product vendors with an open network architecture guide to develop new protocols, networking services, and hardware devices based on a common foundation. The adaptability to the OSI model ensures that newly created products and developed protocols will integrate with products or protocols from any vendor and this lays the foundation for software defined networks, virtualization, and cloud-native applications to be able to interwork with existing technologies and hardware solutions.

2.1.1 OSI Layers

The OSI Model is based on a layered concept, which has advantages such as an easier understanding of the network and its operation. The distinction of each layer helps in the introduction of new protocols, while the logical separation of each layer according to its respective function helps to troubleshoot the network in a much faster way. The OSI model divides the networking communication into seven distinct layers and these layers are always numbered from bottom to top as shown in Figure 2-1. Normally, each layer is known by its number or name, such as Layer-7, also known as the application layer. The network devices and physical hardware such as servers, switches, or routers, perform in the bottom three layers, while the reset of network elements such as hosts, software-based applications, storage, and others, work under all the seven layers.

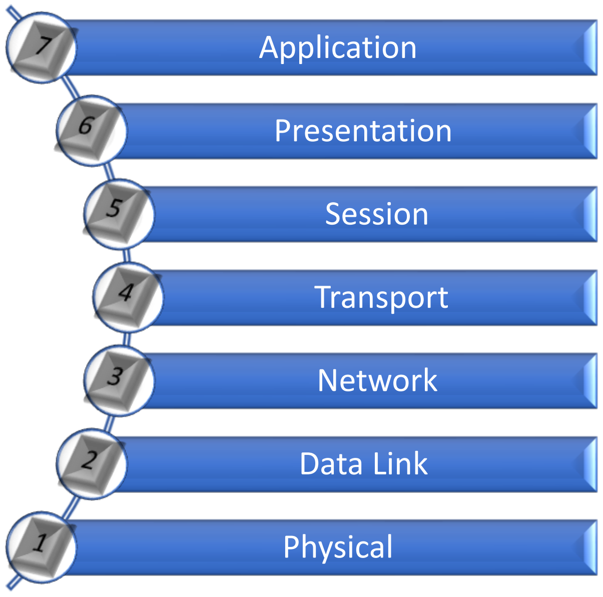


Figure 2-1 : OSI Model seven Layers

The end goal of the model is to exchange the supporting data between the two network entities using certain protocols, therefore, each layer performs its own operations and specific tasks independently. Each layer manages the data unit as a Protocol Data Unit (PDU), and it is overseen in its unique way compared to other layers. Protocols that operate at certain layers add supplementary data in the form of a header or a footer, or both, before sending the information to the next layer. Header information is added at the start of the PDU, while the footer information is added at the end of the PDU. The supplementary information is important to ensure the communication between two remote entities and to understand the exchange information, because this is referred to as “peer-layer strategy” (Alani, 2014). In a peer-layer strategy, the control information added by the sender to the PDU at one layer, is meant to reach the peer layer of the remote receiving devices or hosts thus making the added information insignificant for other layers in the communication stack. This requires compatible protocols at both ends for establishing a successful communication and therefore delivery of user data (Alani, 2014).

Understanding the function of each layer of the OSI model will ensure comprehension of how the network communication is achieved at each layer, starting from the bottom. The consequent sub-chapters discuss the layers in a bottom-top approach.

2.1.1.1 Physical Layer

The physical layer-1 contains physical devices and manages the data as raw bits. The physical layer accepts the frames from the data link layer and converts them into bits for transmission over physical connection media, and at the same time, the physical layer is responsible for receiving the bits on the physical connection and converts them into information to be used by the data link Layer-2. Interface standards such as X.21, V.24, V.35, EIA/TIA-232, EIA/TIA-449, High-Speed Serial Interface, etc. are in the physical layer (Denenberg, 1985).

The physical medium carries a signal, such as an optical signal via a fiber channel, the electrical voltage over a cable, or an electromagnetic signal in the air, and the device drivers help protocols to engage the relevant hardware for transmission and reception of bits. The physical layer also controls throughput rates, multiplexing and demultiplexing, synchronization handling, and managing line noise. Network Interface Cards (NICs), amplifiers, hubs, and repeaters are examples of network hardware devices that function at the physical layer.

2.1.1.2 Data Link Layer

The data link layer manages the data as a frame, and the packets received from the network layer-3 are formatted at the data link layer to frames based on the technology-specific protocol before transmitting to the Physical layer. The Data Link layer includes protocols such as (Technologies Inc, 2007):

- Address Resolution Protocol (ARP) – Resolve IP address into MAC (Media Access Control) physical addresses,
- Reverse Address Resolution Protocol (RARP) – Resolve MAC into IP addresses,
- Layer-2 Forwarding (L2F),
- Point-to-Point Tunneling Protocol (PPTP),

- Link-Layer Discovery Protocol (LLDP),
- Virtual Local Area Network (VLAN),
- Integrated Services Digital Network (ISDN),
- Password Authentication Protocol (PAP),
- Layer 2 Tunneling Protocol (L2TP).

The data link layer provides divergent functions for connection-oriented and connectionless communications, such as error detection, error correction, sequence control, flow control, connection mode data transmission, and others. Additionally, the data link layer includes hardware source and destination addresses in the form of Media Access Control (MAC) addresses to the frame. MAC addresses are comprised of 6-byte (48-bit) binary address written in hexadecimal notation (for example, 78:ac:44:41:88:dc, where the first three bytes denote the manufacturer, and the last 3 bytes represent a unique number assigned interface. It is not possible to have the same MAC address assigned to two devices.) (IEEE, 2022). Network devices that run at the data link layer are switches and bridges. Such devices support MAC-based routing; hence the frame received on one port is routed out to another port based on the destination MAC address. Another way is to use a bridge to transfer the frame from one network to another.

2.1.1.3 Network Layer
The network layer-3 manages the data in the form of packets and is responsible for adding critical routing and addressing information to the data. The network layer receives the segment from the transport layer-4 and transforms it into a packet by including the source and destination IP address. The network layer includes routing protocols such as Signaling Connection Control Part (SCCP), Network Address Translation (NAT), Virtual Router Redundancy Protocol (VRRP), Hot Standby Router protocol (HSRP), Internet Control Message Protocol (ICMP), Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), Internet Protocol (IP), Internet Protocol Security (IPSec) and many more (Technologies Inc, 2007).

The network layer supplies functions such as routing, relaying, sequencing, flow control, fragmentation, error detection, error recovery, and quality of service (QoS). Network devices that run at the network link layer are routers and bridge routers. Routers select the best logical path based on the destination IP address for packet transmission.

2.1.1.4 Transport Layer
The transport layer-4 manages the data as a segment and oversees connection integrity and session control. The transport layer receives the payload PDU from session layer-5 and transforms it into a segment. The transport layer includes protocols such as Stream Control Transmission Protocol (SCTP), User Datagram Protocol (UDP), Encapsulating Security Payload over IP or IPSec (ESP), Transport Layer Security (TLS), Secure Sockets Layer (SSL), Transmission Control Protocol (TCP) and many more (Technologies Inc, 2007).

The transport link layer delivers end-to-end functions such as end-to-end connection lifecycle, segmentation, end-to-end flow control, Quality of Service (QoS) monitoring, and end-to-end error detection. The Transport layer instructs on addressing the device in the network and defines the rules of a session to set up communication connection between nodes. The handshaking process is also implemented at the transport layer as part of session rules (RFC5246, 2023).

2.1.1.5 Session Layer

The session layer-5 manages the data as it arrives from the upper layers without partitioning and concatenation, and governs token management, establishing, maintaining, and terminating communication sessions. The session layer manages dialogue discipline or dialogue control as below (Alani, 2014).

- Simplex: Information will flow in a single direction, and it corresponds to a one-way communication.
- Half-Duplex: Information will flow in both directions, but it is not possible to send and receive the data simultaneously.
- Full-Duplex: Information will flow in both directions, and it is possible to send and receive data simultaneously.

The session layer includes protocols such as Network File System (NFS), RTP Control Protocol (RTCP), Short Message Peer-to-Peer (SMPP), Remote Procedure Call (RPC) and many more (Technologies Inc, 2007).

2.1.1.6 Presentation Layer

The presentation layer-6 manages the data as it comes from the application layer-7 and is responsible for imposing structure and formatting rules to support standards, encryption, compression, and translation. The presentation layer negotiates the date format with the endpoint and then leaves it to the application to use a particular service.

The presentation layer includes protocols such as Transport Layer Security (TLS), Lightweight Presentation Protocol (LLP), Tagged Image File Format (TIFF), Network Data Representation (NDR) and many more (Technologies Inc, 2007).

2.1.1.7 Application Layer

The application layer-7 manages interfacing user applications and defines services that allow the application to communicate with the rest of the protocol stack. The application layer does not stand for the location of the application, but rather denotes the protocol stack and services needed for this layer. For example, the choice of security aspects for authentication and access control is up to the application layer to decide upon.

Most of the protocols are included in the application layer such as the Hypertext Transfer Protocol (HTTP), Transaction Capabilities Application Part (TCAP), File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), Telnet, Simple Network Management Protocol (SNMP), Internet Message Access Protocol (IMAP), Simple Mail Transfer Protocol (SMTP), Network Time Protocol (NTP), Hypertext Transfer Protocol Secure (HTTPS) and many more (Technologies Inc, 2005).

2.1.2 Encapsulation / Decapsulation

Encapsulation is the process of enriching data at each layer by the addition of a header or a footer to the original data before the same data is transferred to the lower layers in the OSI model. The encapsulated data becomes the payload to the receiving layer, and this process continues as the data moves downwards from the application to the physical layer. Similarly, the inverse process is triggered when the data moves from the lower to the upper layers, in which case the process is known as decapsulation. The processes of encapsulation and decapsulation are detailed in Figure 2-2, and it works as follows (M. Stewart, Chapple and Gibson, 2012):

a. The message is formed at the application layer and moved to the presentation layer.
b. The presentation layer encapsulates the message by adding a header at the beginning of the message.
c. The same process continues; respective information is added to the message and is moved to the next layer, until it reaches the physical layer.
d. At the physical layer, a message is covered to bits and transmitted over a physical connection.
e. The receiving end will capture the bits from the physical connection, and messages are recreated at the physical layer.
f. The physical layer will convert the message from bits to frames and send it to the next layer up.
g. The data link layer will receive the message from the physical layer, and it will strip the information and move the message to the network layer.
h. The same process continues, and decapsulation is applied at every layer till messages reach back to the application layer on the receiving side.

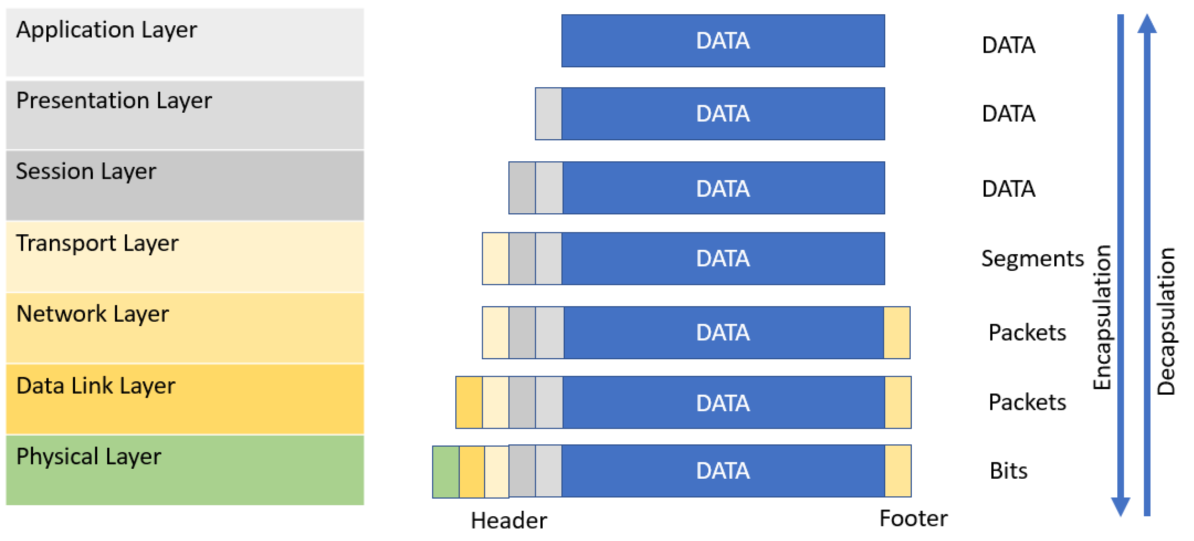


Figure 2-2: Encapsulation and Decapsulation via OSI Model

In the process of decapsulation, the information stripped by each layer is understood by the peer layer that originally added or created the information, and this enables the creation of a logical channel at different peer layers for communication as shown in Figure 2-3 below.

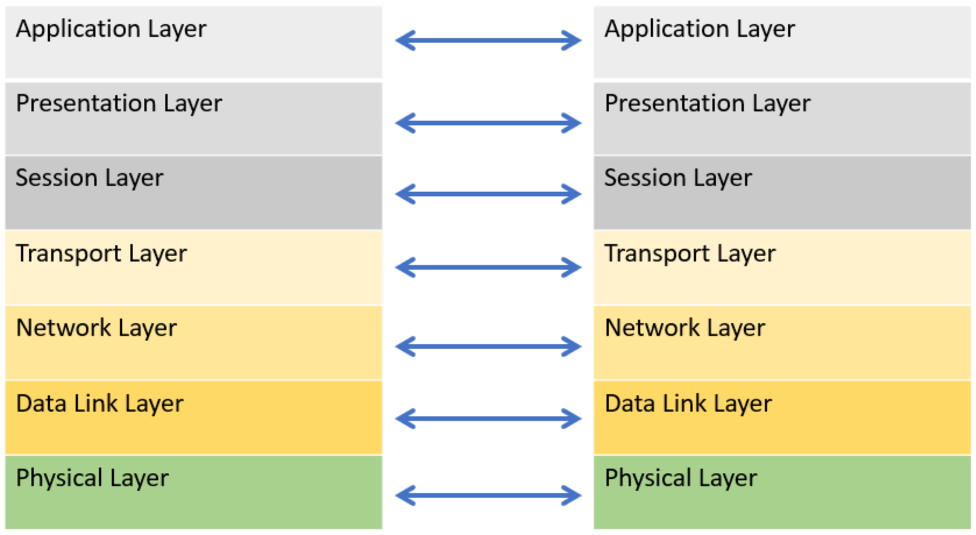


Figure 2-3: Logical Channel via peer layer

2.1.3 Complete Data Flow

A simple setup has been assumed to elucidate the communication via the OSI model and the message flow across each layer. The setup includes a router with two laptops connected to it, which is connected to the internet via a firewall as shown in Figure 2-4

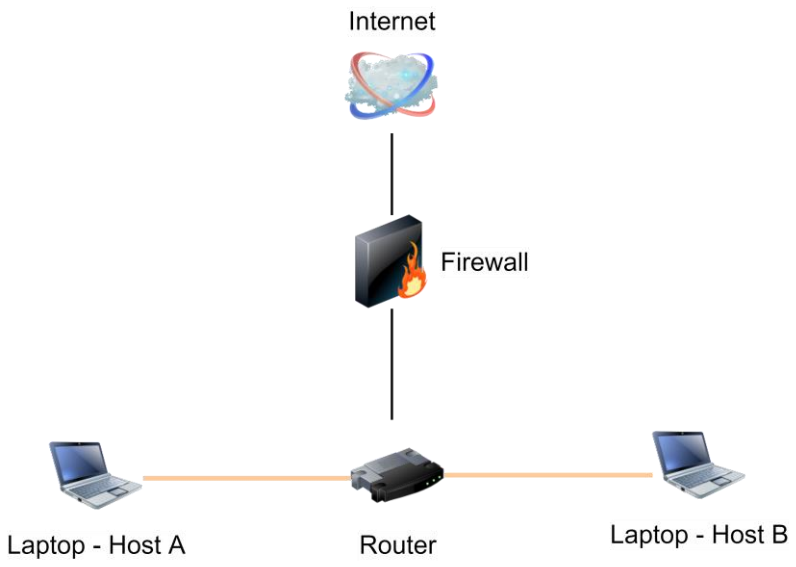


Figure 2-4 : Simple Setup for communication

Normally, network communications occur over a physical connection (whether that physical connection is a copper wire, fiber link, or wireless radio) (M. Stewart, Chapple and Gibson, 2012). The communication of Host A with Host B has been clarified in Figure 2-5

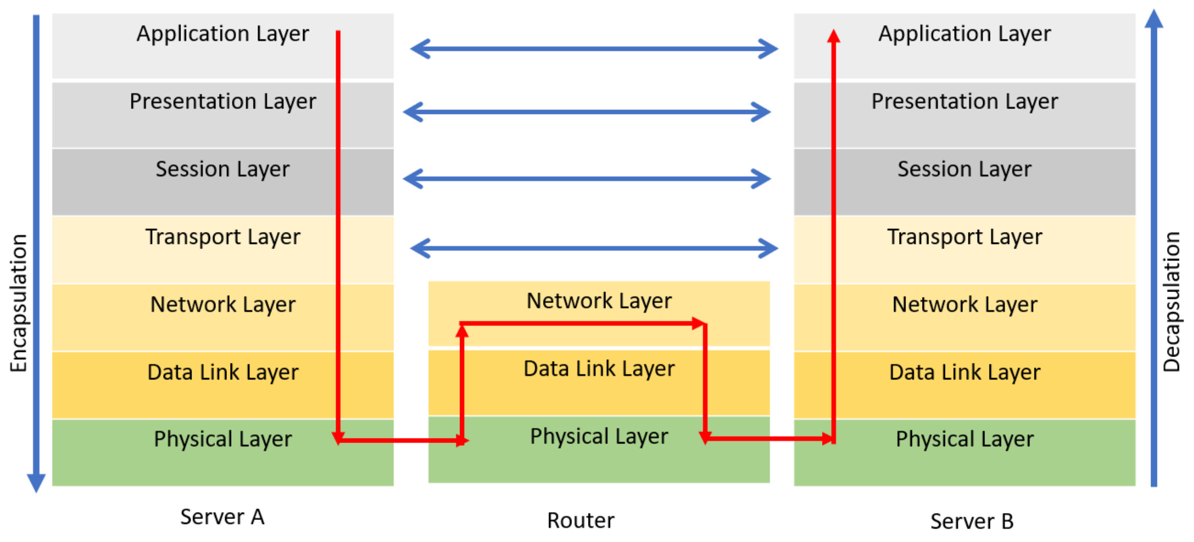


Figure 2-5: End to End Communication Flow

Host-A (Laptop): The transmission is started from the application layer of Host-A. The application needs to communicate with Host-B and so, the data is sent down to the presentation layer, session layer, transport layer, network layer, data link layer and physical layer. It is the responsibility of each layer to encapsulate the data by adding layer-related information to the header and footer of the data before passing it to down to the next layer. At the physical layer, raw bits are transferred through the physical channel to the router (Alani, 2014).

Router: The router runs at the network layer, and it needs to partially decapsulate the data to extract the header information for routing to the desired network or destination. Therefore, the router does not need to read the data up to its application layer and it is not needed to do so. If there is another router in place then the same process that took place on first router is repeated until reaching the destination host (Alani, 2014).

Host-B (Laptop): The destination host receives the raw bits from the router and elevates it in form of frames before sending it to the data link layer. Decapsulation is applied at each layer and the data is traversed to the upper layer until it reaches the application layer of Host-B.

2.2 TCP/IP Model

The Transmission Control Protocol/Internet Protocol (TCP/IP) model was introduced in 1974 (Cerf & Kahn, 2005). The TCP/IP model is based on the protocols that already existed, while the OSI model was defining the framework for the future protocols. TCP/IP is being widely used due to hardware and software independency, flexible addressing scheme, and public documentation of the standards (Russell, 2013).

The TCP/IP model includes four layers instead of 7: Network Access layer, Internetwork layer (known as Internet layer), Transport layer, and Application layer. Figure 2-6 shows the comparison of the OSI model with the TCP/IP model, physical and data link layer of the OSI model has been combined in the TCP/IP model to a network access layer. Session and presentation layers do not exist in the TCP/IP model.

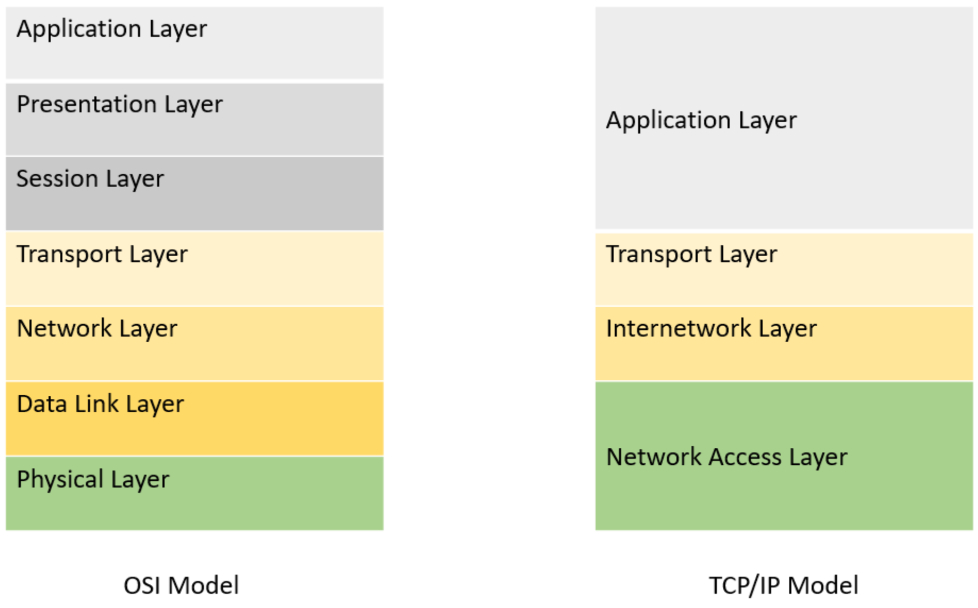


Figure 2-6: OSI vs TCP/IP Reference Model

2.2.1 Network Access Layer

The network access layer processes the IP packets coming from the internetwork layer and their delivery into the physical link and back from the physical link to the network access layer. The TCP/IP model at this layer does not consider the type of local or wide area networking technology or media if an IP packet can be delivered by the network, therefore this means that any Local Area Network (LAN) and Wide Area Network (WAN) technologies can be used below the Internetwork layer. The network access layer maps the IP used in the internetwork layer to a hardware address such as the Media Access Control (MAC), making this layer protocol-agnostic. This layer also does not define its own standard, but instead relies on existing LAN and WAN standards (Alani, 2014).

2.2.2 Internetwork Layer

The internetwork layer also known as “internet layer” handles routing of packet from the source to the destination by selecting best path upon logically designated criteria and algorithms. Internet Protocol (IP) is the main protocol that resides at this layer. Additionally, there are groups of protocols supporting IP such as the Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), and the Reverse Address Resolution Protocol (RARP).

As per RFC791, the IP protocol defines a packet and an addressing schema, transporting of data between the network access layer and the transport layer, selecting best routes, fragmentation, and reassembly of packets. The IP protocol is connectionless, and it does not guarantee the delivery of packets to the destination as these functionalities are managed by protocols at higher layers (RFC791, 1981).

2.2.3 Transport Layer

The transport layer enables the end-to-end conversion between the source and destination. There are two protocols which exist at the transport layer of the TCP/IP model:

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), facilitating connection-oriented and connectionless communication, respectively. The TCP protocol has the sequencing feature to enable the reassembly of packets in case packets reach the destination in different sequences from source, therefore ensuring that all data packets arrive safely. In contrast, the UDP protocol does not provide reliability mechanism for packet delivery to the destination, hence TCP is preferred over UDP by mission critical applications.

TCP and UDP are widely used protocols at the transport layer by different applications, which also depends on the scope and requirements of the said applications (Alani, 2014).

2.2.4 Application Layer

The application layer's responsibility is to collect the data from the transport layer and ensure the delivery to the running application and vice versa. In TCP/IP model, the application layer manages the data representation, dialogue control and encoding/decoding, hence the session and presentation layer functionality of OSI model are implemented at the application layer. The TCP/IP model presents the multiple high-level protocol to different applications such Hyper Text Transfer Protocol (HTTP), File Transfer Protocol (FTP), Simple Mail Transfer Protocol (SMTP), Post Office Protocol 3 (POP3), Telnet, and Domain Name Service (DNS).

The application layer protocols mostly perform plain text conversion using American Standard Code for Information Interchange (ASCII) code. The plain text conversion introduces the security issue as it allows the networking elements to read and alter the session information transferred during the communication between the source and destination, which is why secure versions of the protocols were created such as the Secure HTTP (HTTPS). HTTPS and Secure Socket Layer (SSL) are implemented to verify the host identify of a HTTP conversion and encrypt the data in transit between the source and destination nodes (Alani, 2014).

2.3 Capturing in bare metal

In telecommunication, a bare metal server is a physical server with processor, motherboard, Network Interface Card (NIC), Central Processing Unit (CPU) and memory. The operating system and application are installed on the bare-metal server without any virtualization technology support. The bare metal is deployed without multi tenancy concept, and each bare metal server is used by the single telecom application completely, providing the functionality to the network for the specific service hosted on the server. Communication between two bare metal nodes through a switch or router is the simplest form of communication in any telecommunication datacenter setup.

Figure 2-7 shows the communication between the two servers via a router. Server-1 will be sending traffic to Server-2 the router and vice versa. The communication is achieved by configuring each interface on both servers to send the traffic to the router and then the router updates the MAC table to be able to route the traffic to the destinated server. The communication between the two servers can be captured as below:

- Mirroring at the router: Packets from Server-1 and Server-2 can be mirrored on the router to another port of the router and then forwarded to the server for storage and monitored through a terminal. This way, all the ingress and egress (incoming and outgoing) traffic to the router can be subject to active monitoring.
- Port Capturing: Another way is to perform active monitoring on the port at the server itself. This can be achieved by capturing the packets going to and from the server port. An example is Wireshark, which is a tool to enable capturing of packets on a level of an interface. The capturing of data can be performed in real time and then processed for further analysis.

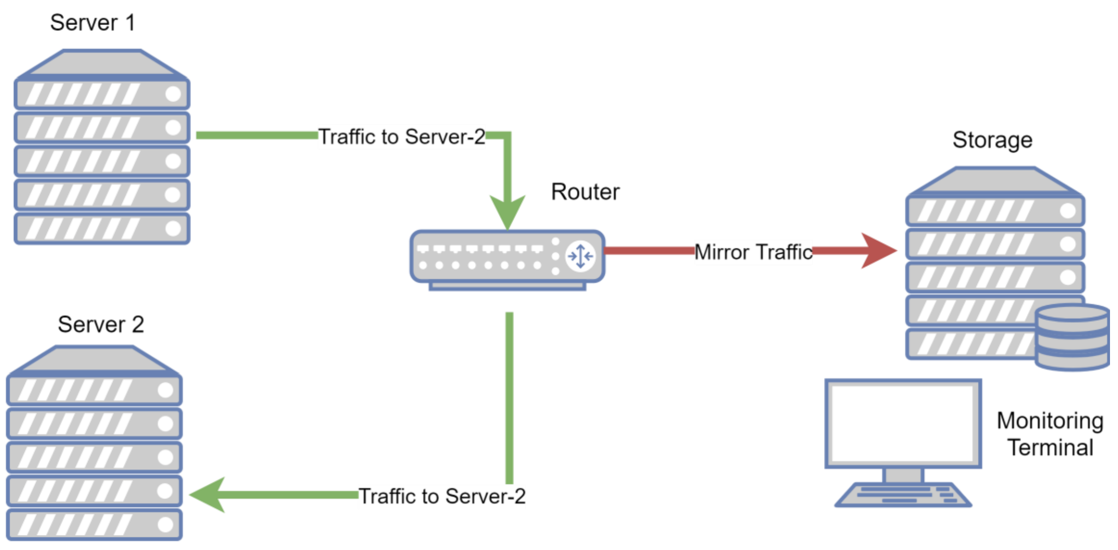


Figure 2-7: Traffic Mirroring

Typically, the application deployed on the servers requires multiple network interfaces to communicate with the rest of the network elements, the interface separation avoids the blending of different traffic for security reasons in the network. To achieve such setup, servers are configurated with multiple network interfaces and each interface is assigned to the respective traffic such as Operation and Maintenance (OAM), Diameter signaling, Session Initiation Protocol (SIP) signaling, Media, charging, roaming and many more, depending on the network requirements. A Telecom-grade application hosted on multi-interface servers is utilizing the respective interfaces to carry different types of traffic towards the serving network via a switch, as shown in Figure 2-8, the traffic from the OAM interface will be sent to the OAM Virtual Routing and Forwarding (VRF) of the network via L3 switch. Similar configuration can be applied on all the other interfaces, ensuring the traffic separation at the interface level, and routing the traffic to the network elements via respective VRF. The application can also communicate with other applications hosted on different servers and connected to the same switch via an internal interface. All those internal and external communications can be captured.

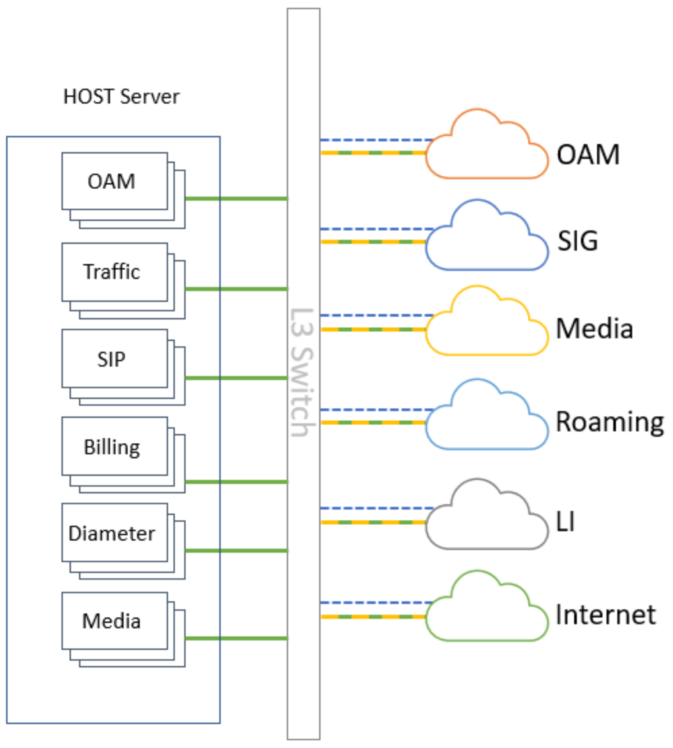


Figure 2-8: Bare Metal Host Interfaces

2.4 Capturing in Virtualization

Virtualization enables sharing of physical hardware resources such as servers, storage, networks, and other physical resources with multiple Virtual Machines (VM). Each virtual machine includes a CPU, memory, storage, and network interfaces. The Operating System (OS) runs inside each virtual machine which can host diverse applications, allowing the same physical hardware to run multiple applications inside each VM with different OS requirements. Virtualization is achieved using the hypervisor[1] by splitting the resources across the virtual machines (TRIANZ, 2022).

Figure 2-9 shows the multiple applications running on different virtual machines on the same physical hardware, achieved by the virtualization concept. The VM utilizes the assigned CPU, memory, storage, and network resources from the physical hardware by the hypervisor, to execute the workload deployed inside the VM, enabling increased efficiency of usage of the underlying hardware.

[1] Hypervisor is the software that creates and run virtual machines.

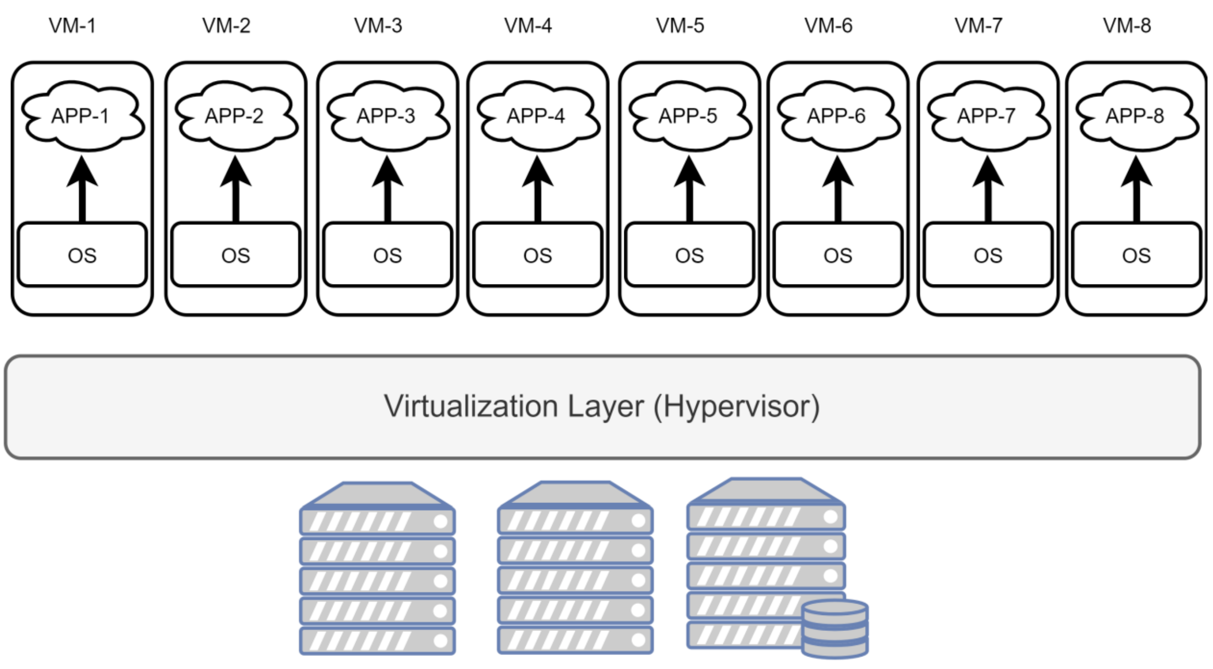


Figure 2-9: Virtualization Concept

The communication of the virtual machine with the external environment is conducted by the host's physical interface. The Linux host for example has a virtual network switch software to manage the communication between the physical interfaces and the virtual interfaces on the VM. Based on the applied configuration, a virtual machine can use an existing virtual network that is managed by the hypervisor or a different network connection method. When the hypervisor is deployed on the host, then by default all the VMs on the host are connected to the same virtual network (named default). VMs can be connected to other VMs on the host and on the network outside the host depending on the OS security system, and the configuration of the Network Address Translation (NAT) rules. Figure 2-10 illustrates the default network configuration for a Red Hat hypervisor "libvirtd"; *virbr0* virtual bridge interface is initialized with the hypervisor and all the VMs on the host use the virtual interface for communication (REDHAT, 2022b).

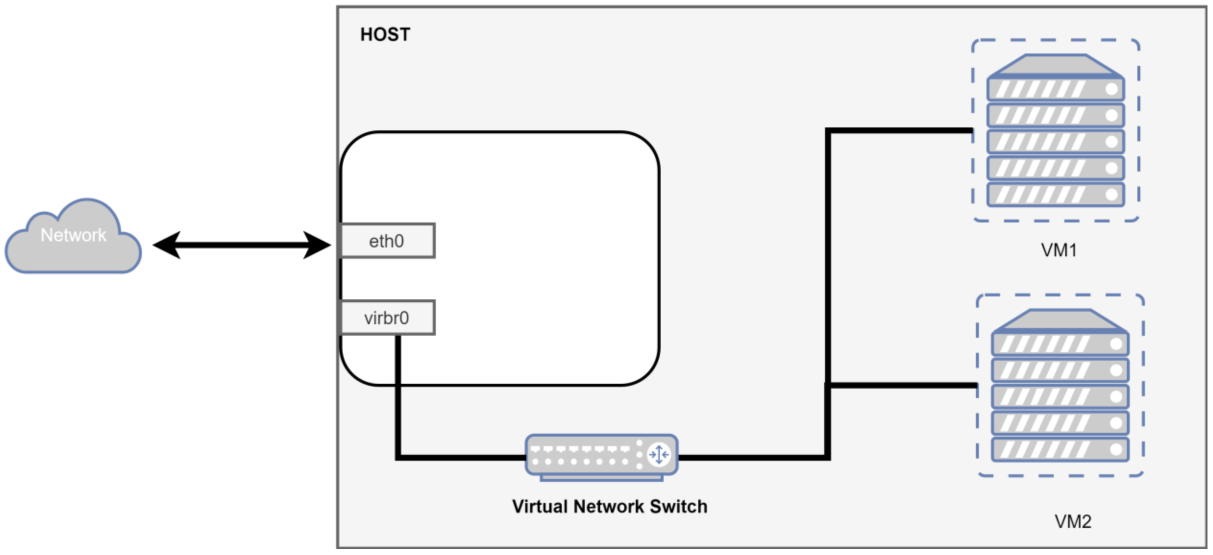


Figure 2-10: VM communication via Virtual Network Switch

An alternate procedure to the default NAT connection is a macvtap driver to manage the guest's[2] Network Interface Card (NIC) and mapping to the physical interface of host. Macvtap supports several types of connection modes to enable different use cases. The bridge mode configuration allows the traffic from VM to VM to be forwarded within the physical interface without going to the switch, while external communication is routed via a switch as illustrated in Figure 2-11 (RED HAT, 2022).

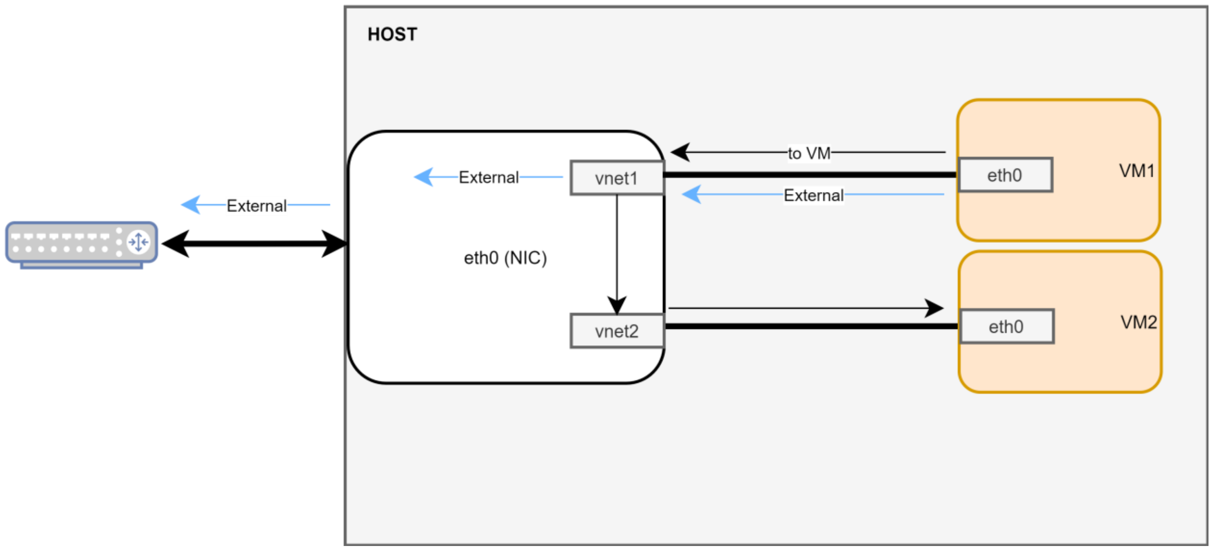


Figure 2-11: VM to physical Interface mapping (macvtap)

The communication between VMs can be captured with the port mirroring technique in Linux by configuring the network connection profile to mirror the network ingress and egress packets from one interface of a host to another interface, for the purpose of monitoring and data collection. The process applies to default NAT and macvtap, as shown in Figure 2-12 (REDHAT, 2022a).

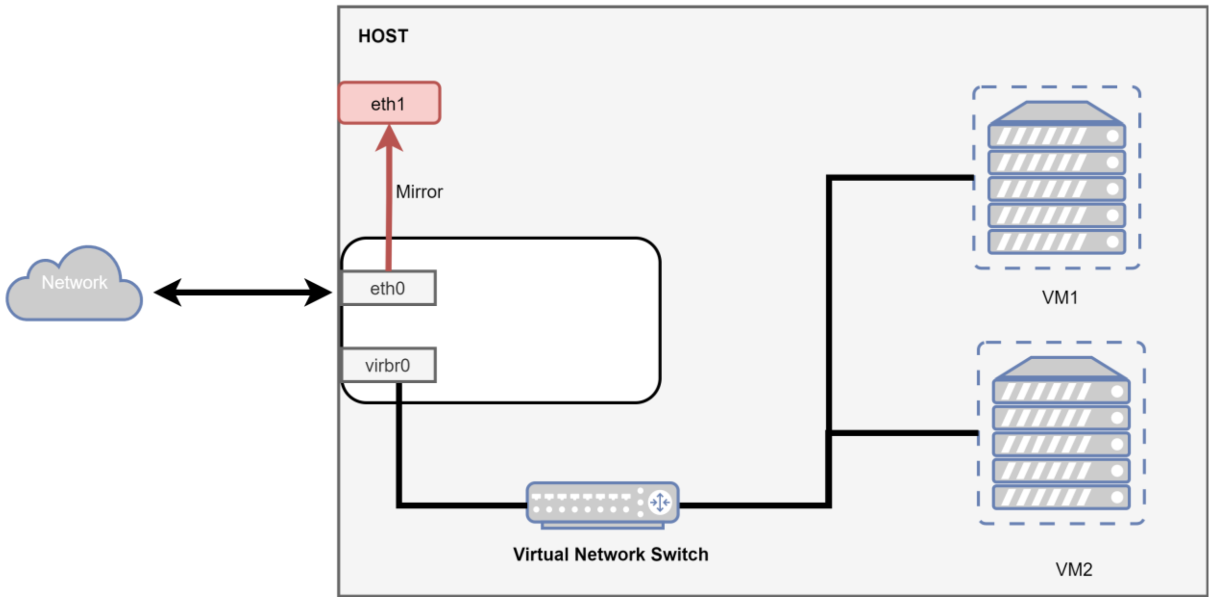


Figure 2-12: VM Traffic Mirroring

[2] Guest operating system (OS) is the OS running inside the virtual machine.

2.5 Capturing in Containerization

Contrary to virtualization, containerization does not require any hypervisor and uses the functionality of the operating system's kernel by isolating the multiple independent instances/containers as namespaces. Containerization is a software virtualization technique applied at the OS level, compared to the virtualization of hardware, thus avoiding the extra overhead the hypervisor introduces. Containerization can be used on the virtual machine itself, allowing a different application to run on the same virtual machine by sharing the same OS kernel. The most common container virtualization solutions are Docker and Linux Container Daemon (LXD), which enable packaging of software into a single container. The application running inside the container doesn't have any influence from the host operating system with same flavor and can execute across different platforms of same flavor without any issues, as all the necessary libraries required for the application to run inside the container are already included at the time of creation of container (Ubuntu, 2023).

Figure 2-13 illustrates the containerization concept and the hardware virtualization with an example of creating three virtual machines via a hypervisor and an operating system being deployed on each virtual machine separately. To enable the further granular virtualization, a container engine is installed on each virtual machine with already deployed operating system to enable the virtualization of an operating system. This enables the creation of containers on the virtual machines but at the same time, containerization is not aware of the infrastructure, such as if it is running on a virtual environment or physical hardware.

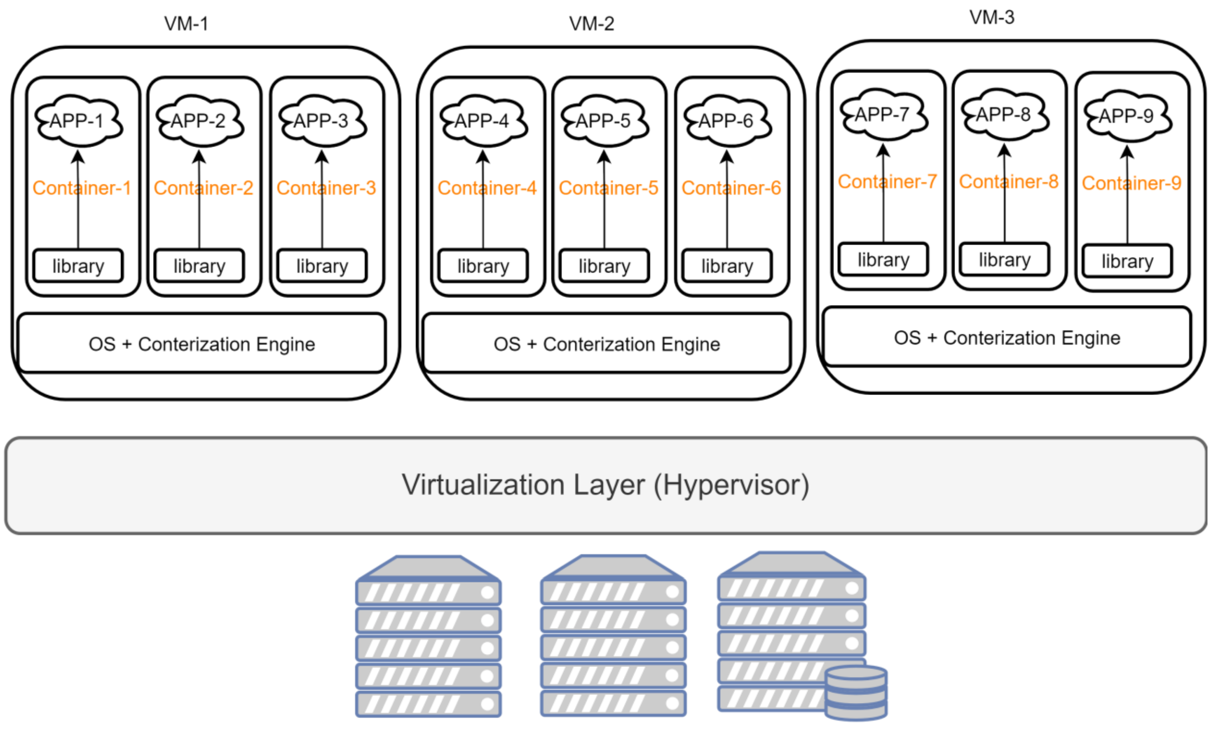


Figure 2-13: Containerization Concept

In Figure 2-13, each virtual machine accommodates three different applications within the individual container along with the necessary libraries, resulting in the same virtual machine hosting multiple applications with minimal to no compromise on the performance. Containers are lightweight solutions as they require fewer resources than virtual machines, and containerization allows application deployment quickly on any environment (Ubuntu, 2023). Containers can also run on bare metal, without the need for virtual machines or hypervisor-based environments. However, such deployments would require bare metal management systems.

Containerization can enable management of immense number of containers in the environment, as well as facilitate lifecycle management of such containers becomes a challenge for the system administrator; hence Kubernetes platform was introduced to deploy and manage the containers on distributed systems, and it also simplifies the networking by combining the multiples containers into a group. Kubernetes includes load balancing, service discovery, automatic rollout and rollback, automatic scale in/scale out, self-healing of failed containers, and configuration management. (KUBERNETES, 2023)

Container network allows the containers to communicate to workload running in any environment and docker achieve this on Linux host by manipulating the iptables. The core container networking functionality is provided by several driver such as: (Docker, 2023)

- **bridge:** a bridge network uses a software bridge in Docker. The containers connected to the same bridge network can communicate with each other, enabling isolation from containers which are not connected to that bridge network (Docker, 2023).
- ***host:*** the "host" network mode attaches the container directly with the docker host and the container is not allocated its own IP as it will be accessible on the docker host IP. This allows only for one container as there cannot be two existing containers using the same IP address. (Docker, 2023)
- **overlay:** the overlay network driver creates a distributed network among multiple Docker hosts, allowing for communication between the containers running on different docker hosts. (Docker, 2023)
- **IPVLAN:** IPVLAN driver allows the control over both IPv4 and IPv6 addressing by building the VLAN driver on top of that provides the complete control of layer 2 VLAN tagging and even IPVLAN L3 routing. (Docker, 2023)
- **MACVLAN:** MACVLAN network driver attaches the container's virtual network interface to the physical network on a Docker host by assigning a MAC address to each container's virtual network interface hence requiring the subnet and gateway for network attachment, allowing for creation of virtual sub-interfaces, and using physical network switching to include containers in an Ethernet network to appear as physical devices Containers can communicate in the same subnet or across VLANs, etc. (Docker, 2023)

The native pod in Kubernetes cluster has one interface apart from a loopback and it is not possible to attach additional Container Network Interface (CNI) to one pod. The attachment of multiple interfaces to the pods in Kubernetes cluster requires additional CNI plugin called Multus CNI (MULTUS, 2023). Multus enables the pods to have multiple interfaces also known as multi-home pod.

Figure 2-14 shows the network interface attached to a pod provisioned by Multus CNI. The Pod has three interfaces such as eth0 used to communicate with Kubernetes cluster network, net0 connect to external network2 and net1 connected to external network1.

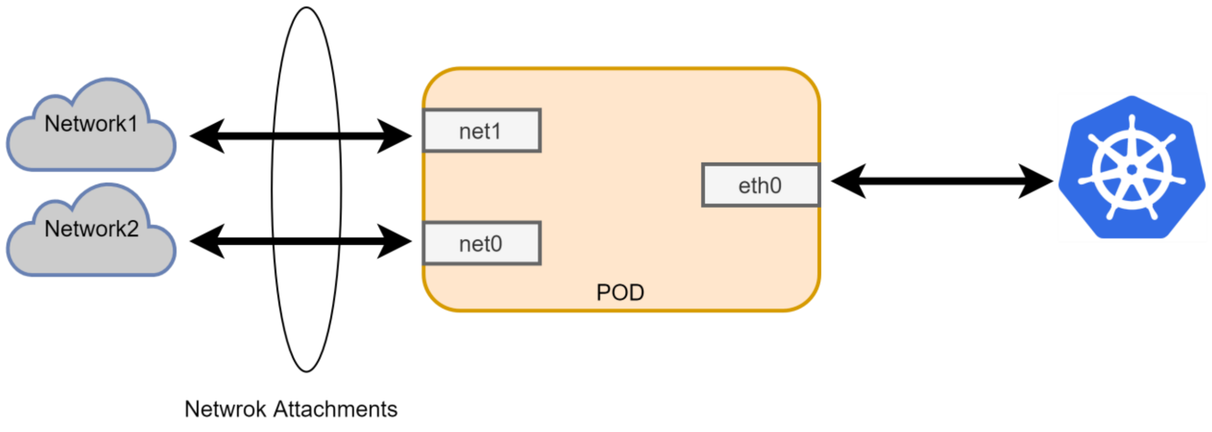


Figure 2-14: Pod with Multus plugin (MULTUS, 2023)

2.6 Apache Kafka - Messaging System

Apache Kafka is the most popular open-source messaging system for distributed applications, and it is backbone for the popular companies such as Netflix, LinkedIn, Spotify, Cisco, Goldman Sachs, and others. The usage of Kafka various from each company to other, for example, it is used by Netflix for real-time monitoring and event-processing, LinkedIn uses Kafka in Newsfeed and Spotify uses Kafka for its log delivery system. There are thousands of companies using Kafka for processing large volumes of streaming data from the real-time applications (PROJECT-PRO, 2023).

Kafka is more popular choice for companies across the globe for the following reason:

- **Scalability:** Kafka is cloud native and highly scalable without any downtime impact (PROJECT-PRO, 2023).
- **Low Latency:** Kafka offer latency in ten milliseconds hence making is product with extremely low latency (PROJECT-PRO, 2023).
- **High Throughput:** Kafka can manage thousands of incoming messages. The messages coming in at high velocity and high volume does not impact Kafka performance (PROJECT-PRO, 2023).
- **Fault Tolerance:** Kafka by default offer fault tolerance for up to n-1 servers in the Kafka cluster (PROJECT-PRO, 2023).
- **Reliability:** Kafka having very high fault tolerance and being distributed platform offers very high reliability (PROJECT-PRO, 2023).
- **Durability:** the data stored in Kafka cluster remain persistent than on the disk (PROJECT-PRO, 2023)

2.6.1 KAFKA Architecture

Apache Kafka is designed for distributed streaming, and it comprises of several components. Figure 2-15 shows the high-level architecture of Kafka along with its necessary components and based on this it can divided in to eight different components and Kafka manager.

- Brokers: The servers associated to the Kafka cluster are known as broker and each cluster can have multiple brokers as shown in Figure 2-15, there are three server and three broker in the cluster. Connection with any one of the broker within the implies a connection to the whole cluster (PROJECT-PRO, 2023).

- **Topics:** A stream of messages is referred to as Kafka topics. Producers are storing the data into topics and consumers are reading it from the topic.
- **Producers:** Producer in Kafka is responsible for publishing the messages into the topics. The message from produced is received by broker and then it appends it to the partition. Producer also has flexibility to publish the message into its own choice of partition (PROJECT-PRO, 2023).
- **Consumers:** Consumers read the data from the specific topic of the Kafka cluster.
- **Partitions:** Topics in Kafka are divided into partitions to enable multiple consumers to read from the topic in parallel. The partitions are distributed across servers in the Kafka cluster as shows in Figure 2-15 of Topic A.

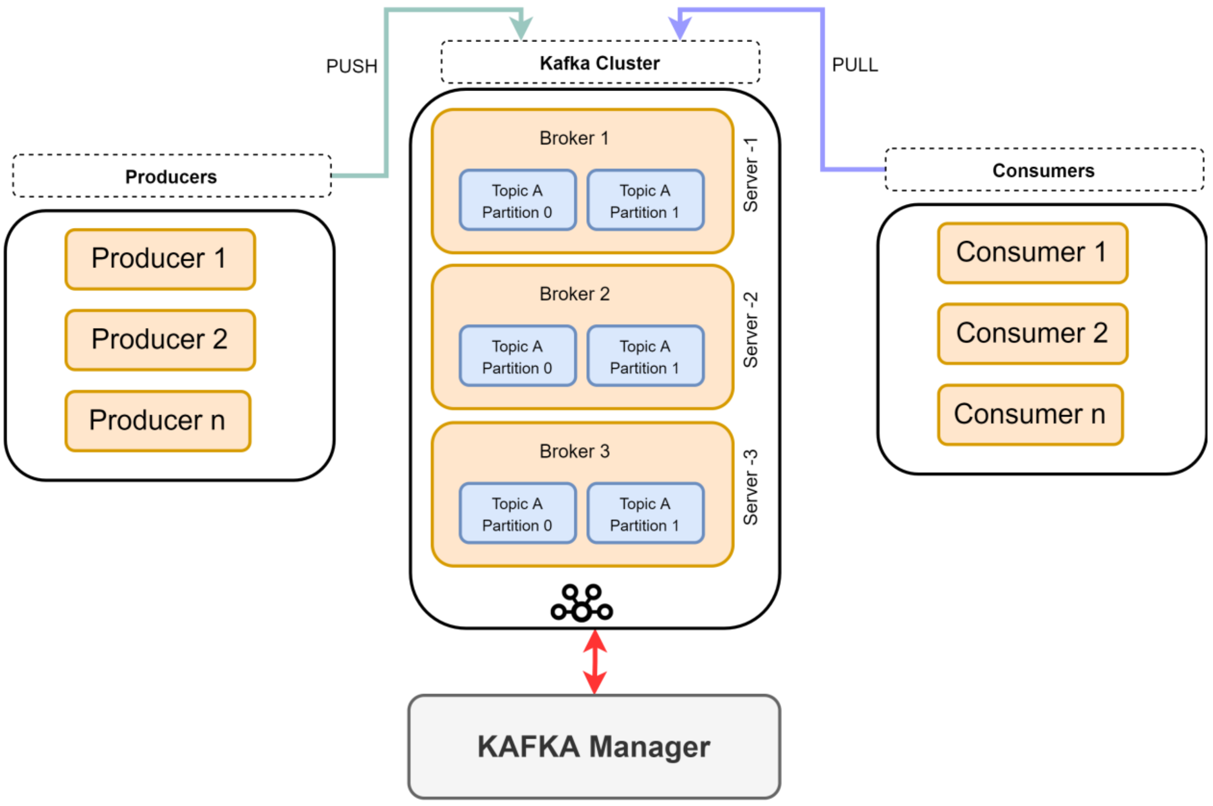


Figure 2-15: Kafka Architecture

- Partition Offset: Messages records within the partition are provided with the offset. The offset value associated to the message can uniquely identify the message (PROJECT-PRO, 2023).
- Replica: Partition of the topics are kept on the different brokers, and these are called replica.
- Leader and Follower: One server will always act as leader for the particular partition and leader is responsible for performing all the read and write tasks (PROJECT-PRO, 2023).
- Kafka Manager (CMAK): Kafka manager provides the UI to manage the Kafka cluster and there are Kafka managers available freely on the internet. CMAK is a comprehensive tool to manage Apache Kafka cluster.

3 METHODOLOGY

This chapter discusses the research design and describes the method chosen to study the 5G core network functions, supported interfaces, communication protocols, and proposed framework to enable data collection from the 5G control plane. The strategy for the data collection method is also discussed, to support the design of the novel framework adapted to the principles of 5G networks in terms of scalability and agility. The chapter begins by describing the research goal and what we strive to achieve, while continuing with a research principle based on the design science method to supply an overview of existing solutions and combine the existing knowledge to help building the stated solution. Finally, the chapter concludes by describing the approach adopted for experimentation and analysis of the described method.

3.1 Research Objective

5G enables the network providers to support heterogeneous traffic for communication of diverse devices, services, and various other consumer-related use cases. The 5G network can be considered as a single network serving different verticals enabling high bandwidth, possibility for massive IoT connectivity, and ultra-low latency. Therefore, it requires scalable and elastic framework to collect data from respective 5G network function and thus enable a machine learning system to use that data for various purposes like network management or. The data consumed by machine learning system can be used to watch the networks for security threats, enabling initiative-taking intelligence in the network to self-heal and scale dynamically based on the traffic requirements and predict the network expansion in real time.

Data traffic on the Internet will grow exponentially with introduction of 5G networks (ERICSSON, 2022) and this requires introducing efficient data collection and monitoring solution to be developed. With the evolution of mobile networks up to 5G, certain questions need to be answered regarding the data that needs to be collected for the purpose of enabling active monitoring at scale:

- How flexible are existing monitoring and capturing solutions to be considered for the future generation networks?
- How 5G networks are designed to provide unified connectivity for multiple consumer verticals?
- How relevant data can be collected from the core network functions in 5G and why are interfaces important for precise collection of the data from the network functions?
- How should 5G network functions be probed to collect the data? What could be the possible procedures and methods to apply such probes on the said functions?
- Which protocols are supported by the 5G core network and how 5G supported protocols become relevant for data collection?
- What are impacts on the underlying infrastructure for data collection, if we consider that5G is designed on cloud native principles? Consequently, how containerization

and virtualization of the infrastructure impacts the data collection procedures and methods?
- What kind of methods and procedures can be followed to process and aggregate the collected data to enable the machine learning models to craft autonomous decision mechanisms related to the 5G core network?
- How data isolation can be enabled across the collection framework to ensure confidentiality across multiple network slices served by the same 5G core network?
- How can the monitoring and capturing solution be designed to support scalability and agility?

3.2 Research Method

5G is the next-generation network and researchers are performing experimentation on various aspects related to 5G security, performance, throughput, network slicing, roaming, data collection, and other. This thesis focuses on the procedures and process for data collection in a 5G core and proposes an innovative framework to collect data from a network function and monitor the overall connectivity based on the data by interfacing to an active monitoring solution. Further, the data can be used to supply a machine learning platform for delivering intelligence in the network for various purposes.

3.2.1 Design Methodology

5G core is the next generation packet core implemented by telecommunication operators and other verticals in various domains. 5G being a future technology, is more reliant on invention work by using the existing and new theoretical knowledge along with the published standards by 3rd Generation Partnership Project (3GPP, 2022). This thesis uses design science method due to the nature of the technology involved in this research work.

The study carried by Venable et al (Venable, Pries-Heje and Baskerville, 2017) proposed the innovative design science research method framework including philosophy (paradigm, objectives, domain), model, techniques & tools, scope, outputs, practices, and product as the main pillars. The researcher applied the framework on six design science research methodologies such as Systems Development Research Methodology (SDRM), Design Science Research Process Model (DSRPM), Design Science Research Methodology (DSRM), Action Design Research (ADR), Soft Design Science Methodology (SDSM), and Participatory Action Design Research (PADR). The rules proposed by Venable et al (Venable, Pries-Heje and Baskerville, 2017) can be applied to choose the right design science methodology and the rules “one design artefact (construct, model, method, or instantiation) or design theory can be found to be best” & “scientific results have to be objective” are relevant to the research work of this thesis so it will be beneficial to follow the design science research methodology to get systematic result from this thesis work. The design science research method encourages innovation by new means to change and improve reality; reality is influenced by creating and evaluating artifacts that solve the problem.

Vaishnavi and Kuechler (Kuechler and Vaishnavi, 2008) proposed five steps process for the design science research. The process starts with the awareness of the problem followed by suggestions to solve the defined issue, so the output of the suggestion includes the various design proposals to solve the problem. The suggestion phase produces the engineering or system design. The third step in the process is the development phase to create artifact based on the design proposed in the suggestion phase, through construction and iterative refinement. The fourth phase is the evaluation phase to frequently measure the performance and process the iterations between the development and evaluation phases until the desired performance metrics are achieved (Kuechler and Vaishnavi, 2008). The last step in the process is a conclusion to analyze the result and propose future work.

Following the design science research method, the thesis addresses the following research components:

- Identification and verification of core network functions in 5G, the relevant data shall be collected,
- Identification and substantiation of interfaces on each network function, which should be probed to apply the monitoring of the relevant functions,
- Identification and verification of relevant data from protocols to be collected from the selected network functions,
- Defining the scalability and agility of the monitoring and capturing solutions,
- Enabling data stream isolation and anonymization to ensure user confidentiality across multiple network slices served by the same 5G network.

5G core networks software and end-to-end solutions are going through continuous development and evolution to adapt to the commercial requirements in various areas, as well as the data collection and autonomous monitoring being one of the major concerning topics. By using the design science research methodology, we build a framework for data collection, which shall pave the way for future research and development of flexible solutions for data collection and monitoring in 5G and 6G.

In the previous section, we described the Research Objective, which defined the questions for the research work covered in this thesis. Consequently, Figure 3-1 represents the high-level research process that is adapted upon the principles of the design science method to address the raised questions. The following section of this chapter explains each process in detail.

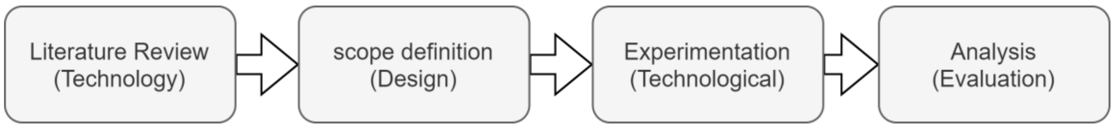


Figure 3-1: Research Method

3.2.2 Literature Review

5G is designed to support various applications and services and proposes a flexible architecture that takes advantage of user and control plane separation to allow greater elasticity in handling heterogeneous devices on that shall be accessing diverse kinds of applications. This thesis studies the 5G core (5GC) architecture in accordance with the 3rd Generation Partnership Project (3GPP) 3GPP standards as a reference (3GPP, 2022).

Figure 3-2 shows the focused blocks during the literature review to enable seamless design creation of the framework for the data collection framework. The focus areas of the literature review are listed below:

- The study of the Open System Interconnection (OSI) model to better grasp the functionality provided by each OSI layer to enable the communication between different entities in any networks.
- The review of the Transmission Control Protocol/Internet Protocol (TCP/IP) model to understand the involved layer within the packet transmission and the feature & functions of each layer to enable the communication between the source and destination.
- Literature explaining the cloud native infrastructure to enable the cloudification (virtualization and containerization) and the network communication of the different entities depending on the nature of used technology.
- Overview of the tools to capture the data and move the data in form of streams for further processing with more scalability and agility.
- The architecture review of the 5G core network. The 5G System architecture is represented in two ways in the 3GPP standards: one is a service-based representation in which the control plane network functions access each other's services, and the other is a reference point representation in which the interaction between the network functions is shown with point-to-point reference arguments (Lei et al., 2021). As part of a detailed review in later chapters, the 5G core (5GC) architecture is studied and explained using 3GPP standards as a reference.
- The study of literature relevant to the Network Functions (NFs) constituting a 5G Core Network (CN),
    - Design of each network function from reference point representation and service-based representation.
    - The role of the network functions within the 5G core network.
    - Feature supported by the network functions.
    - Functionalities provided by the network functions.
- Literature review to understand the interfaces supported by each Network Functions in 5G core network and the information transiting from the network functions through the relevant interfaces.
- Review of relevant protocols supported within a 5G Core Network using OSI and TCP/IP model layers to understand the information processed by each layer and which enable the communication between different network functions of a 5G core network.
- Literature reviewed for various tools and assets such as Wireshark (TSHARK, 2023), Kafka (KAFKA, 2023), Elasticsearch, Logstash, Kibana (ELASTIC, 2023) and JavaScript Object Notation (JSON) (JSON, 2023). Wireshark literature has been reviewed to understand the network analyzer and JSON literature to understand the objects and arrays of the data format. Kafka, Elasticsearch, Logstash and Kibana literature are reviewed to understand the data processing and data presentation in graphical format for deep analysis.
- Finally, reviewing literature to interpret the end-to-end communication occurring within the 5G core networks.

The detailed literature review enables the definition of the scope and further allows for proposing assorted designs to capture the data from 5G core functions along with the identified interfaces and protocols, on which such data capture should be applied.

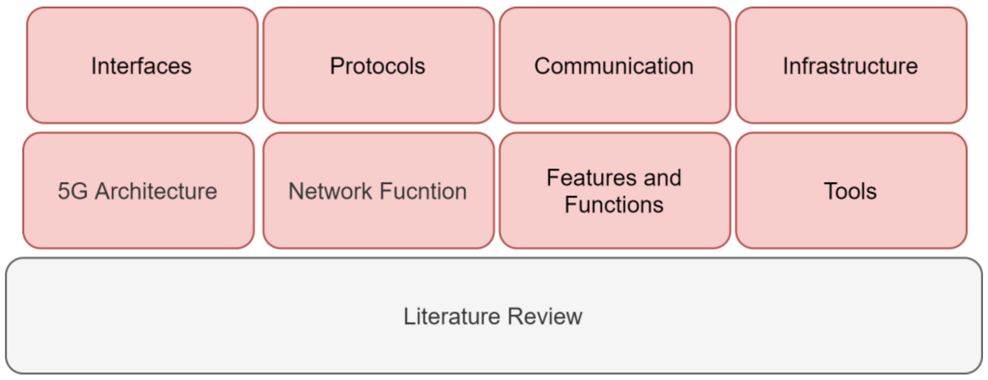


Figure 3-2: Literature Review structure

3.2.3 Scope Definition (Design)

Figure 3-3 shows the focused blocks during the scope definition for experimentation with the data collection framework. The focus area of the scope definition is listed below:

- **Infrastructure:** The 5G core networks are agnostic to the underlaying infrastructure, thus enabling the deployment of the 5G core network over virtual machines or containers. Oslo Metropolitan University has the cloudified environment available based on OpenStack (OPENSTACK, 2023) thus creating the environment based on multiple virtual machines and then using the virtual machines environment to enable containerization by deploying Docker and then Kubernetes for the life cycle management of the containers. Additionally, Oslo Metropolitan University has BareMetal servers that ease the deployment of Network Function Virtualization and so, 5G core networks.
- **5G Simulator:** 5GC network simulator is primarily prerequisite to build, implement and evaluate the framework to show the monitoring points for the 5G core network functions and data collection solution with corresponding metrics. There is multiple open source 5GC simulators and projects available, and few are listed below:
    - **Simu5G:** The Simu5G is an open-source network simulator, and it is developed in collaboration of Intel and the Department of Information Engineering of the University of Pisa. (SIMU5G, 2022) (HEXA-X, 2022)
    - **Open5GS:** The Open5GS is also an open-source implementation of 5G core. It is written in C language and provides WebUI for testing purpose. (OPEN5GS, 2023)
    - **Free5GC:** The free5GC is an open-source project to implement the 5GC network in Communication Service/Software Laboratory (CS Lab) managed by National Chiao Tung University (NCTU) (FREE5GC, 2022).
    - **OpenAirInterface5G:** The OpenAirInterface5G is a project developed by EUROCOM, a French graduate school and a research center in

communication systems, based in the international science park of Sophia Antipolis within the new Campus Sophia Tech (EURECOM, 2022a).

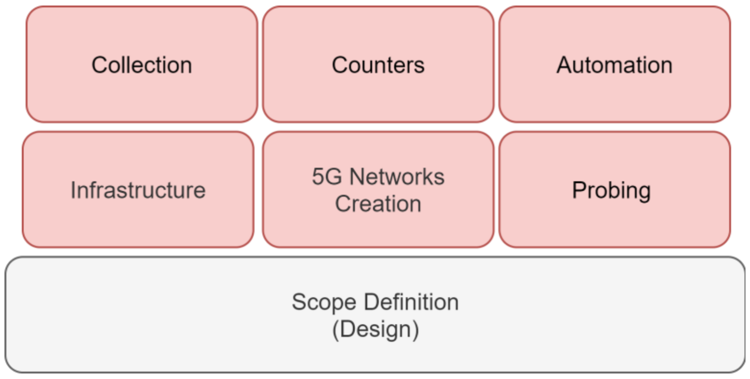


Figure 3-3: Scope Definition

- **Automation:** Infrastructure and 5G core deployment via automated process to enable the network restoration; design of such process is needed for fast redeployment of the 5G core networks to solve the failures scenarios. Manage Large Network (MLN) tool (Begnum, 2009) has been used to automate the provisioning of the OpenStack cloud with necessary virtual instances. Puppet (PUPPET, 2022) and Foreman (FOREMAN, 2022) have been used in combination to automate the deployment of the Docker and Kubernetes environments on the multiple virtual machines. The Kubernetes cluster will be created with master and worker nodes by assigning the roles to the virtual machines during the deployment process via puppet classes. Finally, 5G core network function will be deployed via helm (HELM, 2022).
- **Probing:** Data capture is possible by probing the interfaces of network function of 5G core network, which shall be collected and stored in what we will call a UDSF (Unstructured Data Storage Function), a supplementary 5G core function that has the task of collecting and storing the relevant data. The following core network function will be probed to collect the data:
    - *Access and Mobility Function (AMF)* is the operator’s network gatekeeper. It ensures the mobile subscriber is authenticated and authorized by its home operator before it is registered and offered access to network service. During subscriber registration, AMF sets up a security association with the User Equipment (UE) to secure all future Non-Access Stratum (NAS) Communications between the UE and AMF. After a UE is successfully registered, AMF starts to manage the ongoing state and mobility events for the UE to keep reachability for always-on access to services and facilitates UE’s communication with other 5GC NFs, e.g., SMF (Session Management Function), to establish one or more connections with different 5G Quality of Service Identifiers (5QIs) based on user request and operator authorization.
    - *Session Management Function (SMF);* After User Equipment (UE) is successfully registered with the 5G core network, i.e., the AMF, the UE starts using the non-access stratum (NAS) connection to request the setting up of one or more sessions. As part of the Protocol Data Unit (PDU) session establishment, the AMF uses Single Network Slice Selection Assistance

Information (S-NSSAI), Public Land Mobile Network (PLMN) ID, and Data Network Name (DNN) to select a SMF via the NRF. For SMF to manage the same PDU session for the length of its lifetime, AMF ensures 1:1 mapping between the UE PDU session and the SMF. AMF forwards NAS SM messages over the N11 interface for SMF to set up a PDU session. SMF utilizes either a locally configured address pool or uses Dynamic Host Configuration Protocol (DHCP) to dynamically allocate an IP address/prefix for the PDU session. SMF may also give a static IP address via the Unified data Management (UDM) based on the subscriber profile. It also selects the Session and Service Continuity (SSC) mode, by either using local configuration or via UDM based on subscriber profile. (3GPP-TS.23.501, 2022)

    - *The User Plane Function (UPF)* is a fundamental part of the 3GPP 5GC architecture. As part of the evolution of user plane and control plane separation architecture, the UPF is the decoupled user plane part while the SMF is the control plane. This evolution enables the data forwarding component (UPF) to be decentralized while the control plane (SMF) remains in the core network. The UPF can be deployed to process packets and aggregate traffic closer to the network edge or the subscriber location. This increases bandwidth efficiencies while reducing network complexity and at the same time reduces traffic latency to meet the 5G use cases requirements, e.g., the URLLC use case requirements for low-latency communication.

- **Data Collection:** There are various methods to capture data and information from the interfaces. The following methods will be implemented in this thesis:
    - *Wireshark*: Wireshark is a network protocol analyzer that captures packets from a network interface. Wireshark capture will be used on the interfaces of the worker nodes to capture the packets transiting the interfaces and it will be capturing all the packets and protocols between the different networks function and the external entities.
    - *A SideCar Container*: When 5G, core network is deployed in containers, then a Sidecar container can be attached to main container of the network function. The SideCar container will be used to capture relevant metrics from the Network Function, without interrupting the actual functionality of the NFs. The information collected by the SideCar container can be directly embedded into the code of the network function, hence metrics are shared by the network function after processing the packets received from other network functions.
- **Counters:** The control and user plane split enable flexible network deployment and operation, by distributed or centralized deployment and independent scaling between control plane and user plane functions. This makes the network highly flexible and highly programmable, enabling the most effective use of key technologies such as Cloud and Software-Defined Networking (SDN). This is also the key feature which enables distribution of services to the edge for low latency applications. The information transiting the interfaces is massive so this can be divided into two stream counter groups:

- *Control Plane Counters*: In a 5G core network, all NFs, i.e., AMF, SMF, and UDM, are in the control plane except UPF which is counter-related to initial registration requests received, mobility update requests received, emergency registration requests received, registration accepts sent, retransmissions, registration complete received, registration reject sent, registration rejected due to different 5GMM cause, deregistration, authentication, NAS security mode, Downlink (DL) and Uplink (UL) NAS transport, NAS PDU session, NAS configuration, NAS notification and many more counters from other network functions. The counters for each network function are discussed in detail in chapter 6.
- *User Plane counters*: The UPF is in the user plane. Separating the user and control planes allows each plane resource to be scaled independently. This separation also allows the UPFs to be deployed separately in a distributed fashion.

3.2.4 Experimentation via Implementation

In the experimentation phase, the design will be implemented to collect the data from the network interfaces by deploying a 5G core network in the Oslo Metropolitan University lab facility with the desired network functions. Figure 3-4 shows the high-level implementation of the flow to capture the data from various interfaces.

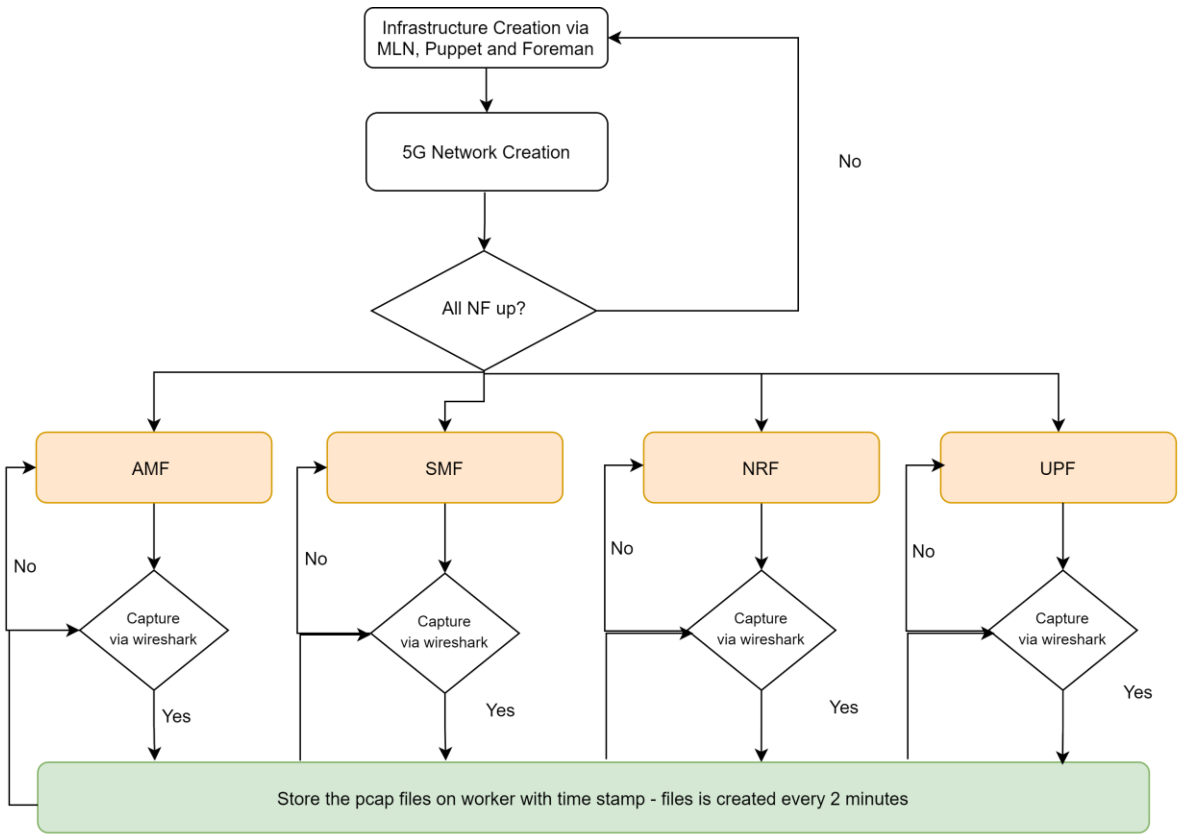

Infrastructure Creation via MLN, Puppet and Foreman
5G Network Creation
No
All NF up?
AMF
SMF
NRF
UPF
No
Capture via wireshark
Yes
Store the pcap files on worker with time stamp - files is created every 2 minutes

Figure 3-4: Capturing the Data

- MLN is the tool used to create the infrastructure in the lab facility at the Oslo Metropolitan University private OpenStack cloud. Multiple virtual machines are created to distribute the workload.

- Puppet and Foreman are used to install Docker to enable the containerization layer and Kubernetes is deployed to perform the lifecycle management of the containers. The 5G core network will be deployed on top of the created infra and container layer.
- The 5G core network functions will be deployed on the created infrastructure, while the creation of the network functions will also be verified. In case of failure, the infrastructure will be automatically redeployed, and the flow shall commence from the first step.
- The Access and Mobility Function (AMF) deployed will be probed via Wireshark. Wireshark will start capturing the interface traffic on the work node and begin storing the output files on the worker nodes. The capture will stop after two minutes, and a new capture will start. This will help to control the size of the stored capture file and consequently the number of captured packets inside the file. The capture will be implemented on the N2 reference point between NG-RAN and AMF.
- The Session Management Function (SMF) will be probed via Wireshark. Wireshark will start capturing traffic on the interface on the work node and store the output files on the worker nodes. The capture will stop after two minutes, and new capture will be started. This will help control the size of the file and number of captured packets inside the file. The capture will be implemented on the N4 interface.
- The same process will then be repeated for User Plane Function (UPF), with the only exception that the user plane signaling traffic on N4 will be captured and processed.

The packets transiting through the interfaces are captured by Wireshark and made available for further processing at the worker nodes. Figure 3-5 shows the flow to process the captured packets by Wireshark.

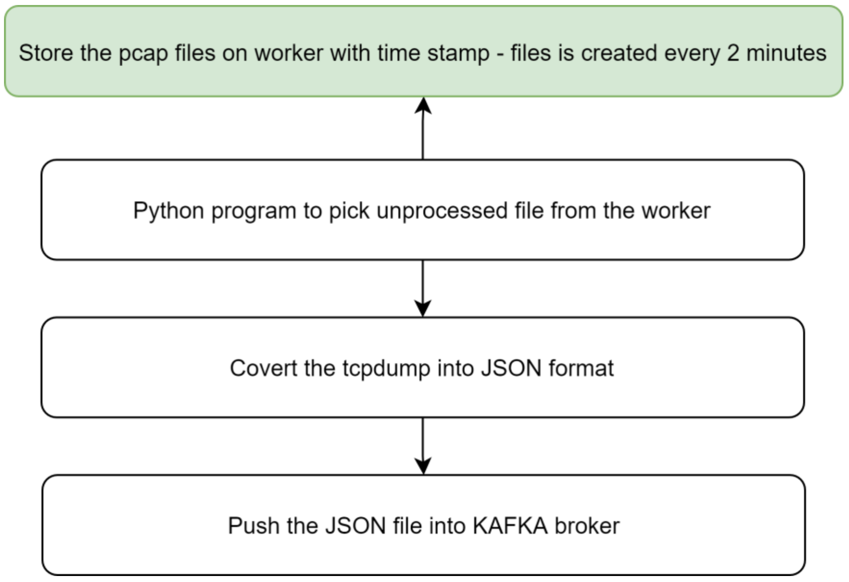

Store the pcap files on worker with time stamp - files is created every 2 minutes
Python program to pick unprocessed file from the worker
Covert the tcpdump into JSON format
Push the JSON file into KAFKA broker

Figure 3-5: Processing of Packets

- Python is used to build the processing engine for pulling the Wireshark output file from the stored location on the worker nodes. The file is then removed from the workers after it is successfully consumed by the processing engine.

- The processing engine will also convert the raw file into a JavaScript Object Notation (JSON) format, thus creating the object and array of the packets processed by network function. Table 3-1 shows the object and array of the sample NAS messages extracted from the Wireshark captured trace.

Table 3-1: NAS message sample in JSON

```
        "nas-5gs": {
          "Security protected NAS 5GS message": {
            "nas_5gs.epd": "126",
            "nas_5gs.spare_half_octet": "0",
            "nas_5gs.security_header_type": "2",
            "nas_5gs.msg_auth_code": "0x418bf535",
            "nas_5gs.seq_no": "6"
          },
          "Plain NAS 5GS Message": {
            "nas_5gs.epd": "126",
            "nas_5gs.spare_half_octet": "0",
            "nas_5gs.security_header_type": "0",
            "nas_5gs.mm.message_type": "0x68",
            "nas_5gs.spare_half_octet": "0",
            "Payload container type": {
              "nas_5gs.mm.pld_cont_type": "1"
  },
```

- The Kafka broker will be set up during this process, and a JSON format output produced by the processing engine, which shall be pushed by python further into the Kafka topic for consumption.
- Kafka enables a solution for storing, reading, and analyzing streamed data in real-time. The data that is pushed into the Kafka topic can be kept inside the Kafka topic for configured period, and the expired data will be cleaned up automatically. Furthermore, Kafka provides a possibility for the data to be read by other tools for aggregation, structuring and better presentation for use in other scenarios.
- A Machine learning platform can use the raw data directly from the Kafka topic to start learning about the network and process the data to enable monitoring of the network for health and security.

3.2.5 Analysis

The data produced during the experimentation phase is stored in the Kafka topics and can be analyzed to understand the quantity and quality of the extracted data from the 5G core network functions. The analysis phase will focus on the quantity and quality of the data captured during the experimentation phase and shall identify the information transiting through the core network functions. The information can be used in machine learning to build data pools for realizing the network behavior over time, depending on the desired factors.

- **Quantity:** The data is captured from four network functions of the 5G core network, and the captured data is converted into JSON format and stored in a Kafka topic. The data is then stored in raw format without applying any filters for removing the unnecessary information, hence these results from the parsing of substantial unnecessary data. The raw data will be filtered by removing the irrelevant aspects to the 5G network, such as Ethernet and Media Access Control (MAC) details. Each filter will be assessed during the experimentation phase until the right quantity of the data has been captured.

- **Quality:** Data captured from four network functions will have information about the traffic passing through the 5G network. This information will be related to the registration, NAS messages, PDU establishment, IP assignment, NF registration into NRF and other data groups. The quality of the captured data will be accessed and analyzed and if needed, further experimentation will be performed to capture the data from the other network function to fully understand the traffic processed by the 5G core network.

The processed data passing the gate of quantity and quality needs to be aggregated so that it can be stored for a longer time and with less storage space requirements. The aggregation allows identical data to be stored in a more efficient way by enabling the data availability for the longer duration and allowing a machine learning system to bring more efficient decisions based on the old data.

There will be multiple iterations between the analysis and experimentation phase as the quality and quantity of the data is decided within the analysis phase. The experimentation may be conducted after every analysis, until the desired acceptable data is collected. Figure 3-6 outlines the revised method with updates of iterative approach between the experimentation and analysis, instead of applying the waterfall approach (Kuechler and Vaishnavi, 2008). The rest of the process is unaffected, while only the iteration is implemented between the experimentation and analysis phases.

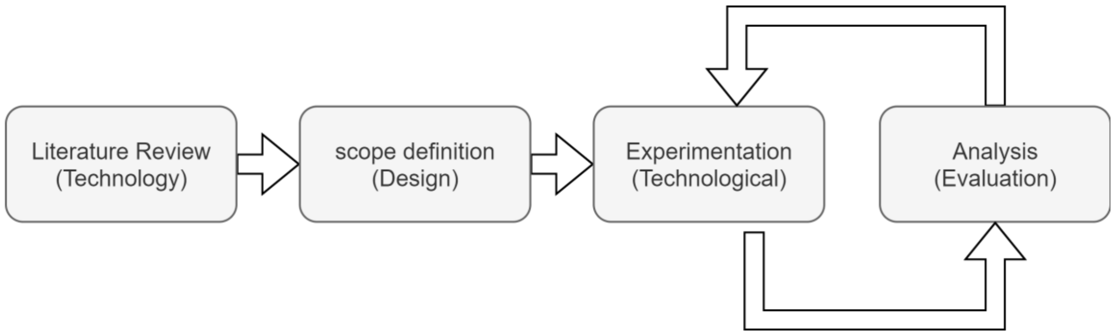


Figure 3-6 Revised Methodology

3.3 Summary

In this thesis, we use the design science research method to construct the data collection and monitoring framework by addressing the following research components:

- Identification and verification of network functions in 5G network to collect the relevant data,
- Identification and substantiation of interfaces on each network function, which should be probed to apply the monitoring of the relevant functions,
- Identification and verification of relevant data from protocols to be collected from the selected network functions,
- Defining the scalability and agility of the monitoring and capturing solutions,
- Enabling data isolation and anonymization to ensure confidentiality across multiple network slices served by the same 5G network.

Furthermore, the thesis elucidates the research components mentioned above, in the following steps:

- Scope definition to perform the experiment,
- Research problem identification,
- Experimentation via implementation,
- Analysis and evaluation,
- Discussion and conclusion.

4 ARCHITECTURE – 5G NETWORKS

Mobile data traffic is rising rapidly, and demand for increased connectivity has been growing due to the usage of multiple devices by end-users. Smart industries, intelligent houses, and the Internet of Things (IoT) require networks that can control immense number of device connections in a brief period. The growing number of devices and continuous increase in data traffic demand for flexible network architecture. Hence, this chapter clarifies all the essential architecture traits of the 5G network concerning access and core network.

Furthermore, the chapter clarifies the fundamentals of the predecessor fourth-generation Long Term Evolution (4G LTE) technology, to realize the needs for the next-generation network with evolved architecture due to the 4G LTE limitations. Such constraints can be the ability to adapt to Softwarization, virtualization, containerization, communication complexity, integration complications, scalability challenges, automation constraints, orchestration flexibility, and agility. Softwarization, containerization, complexity, and orchestration have been the main drivers of the 5G networks development. Therefore, understanding the 4G LTE architecture is vital, and as a result, it has been explained in detail in the consequent sub-chapter.

4.1 Cellular Communication History

The mobile cellular networks have been developing and progressing since the 1970s. Figure 4-1 shows the schematic view of the history of cellular communications, where the first generation (1G) mobile network provided the analog service based on frequency division multiple access (FDMA). After ten years, a second generation (2G) network was introduced with time division multiple access (TDMA), enabling digital voice and low data rate service. In the early 2000s, a third generation (3G) was developed in the form of a Universal Mobile Telecommunications System (UMTS) as its core network architecture, which introduced new protocols to deliver significantly faster data rates and many other data-intensive services.

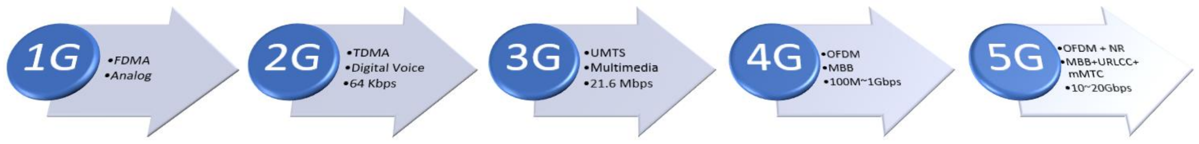


Figure 4-1: Cellular communication History

The demand for continuous connectivity and data services kept on increasing, which showed the limitations of the 3G system. At the same time, users expected to connect to any network on-the-go and demanded broadband experience from wireless mobile network services. In 2005- 2010s, 4G LTE was developed and deployed with Orthogonal Frequency Division Multiple Access (OFDMA) technology, and an Evolved Packet Core (EPC) resulted in a massive improvement in data rates and support for many more relevant services. Additionally, the OFDMA concept is well incorporated with multiple-input multiple-output (MIMO) technology to increase network efficiency. The ever-increasing demand for numerous services, spectral efficiency, higher data rates, and lower connection latency, laid the foundation of 5G. In this context, the International Mobile Technology-2020 (IMT-2020) vision for communication was published to set the target for future generation networks, as shown in Figure 4-2 (ETSI, 2015). The fifth generation (5G) network delivers significantly increased operational performance (for example, increased spectral efficiency, higher data rates, and ultra-low latency) with superior user experience and full mobility and coverage. The 5G network aims for massive deployment of the Internet of Things (IoT) while still offering acceptable energy consumption levels, equipment costs, and network deployment and operational costs (ETSI, 2015).

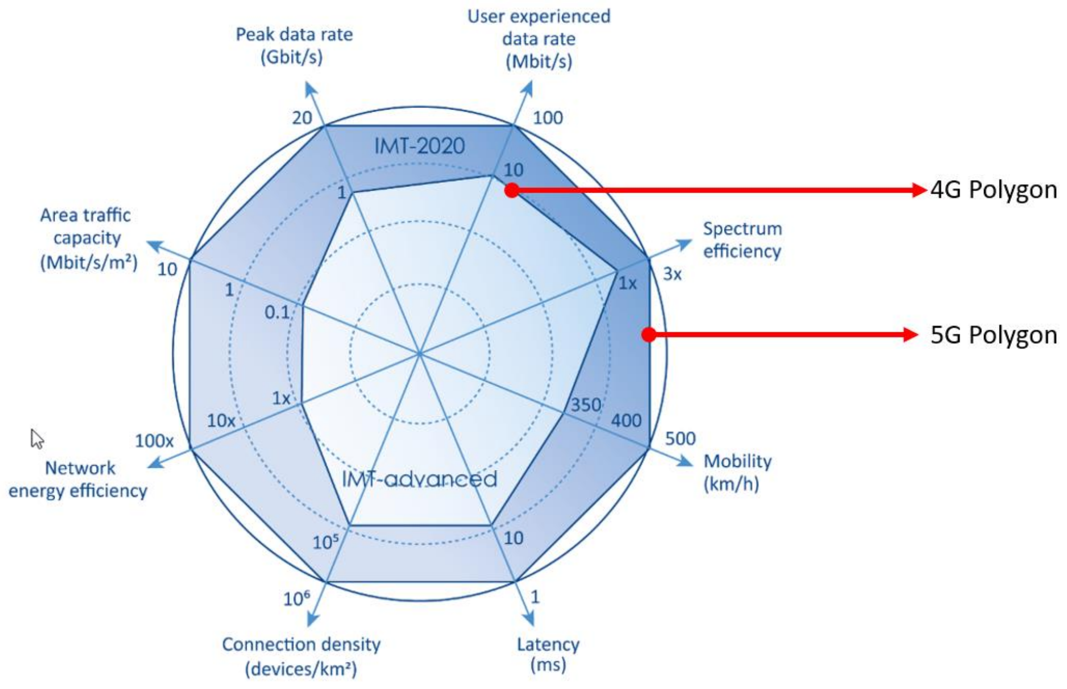


Figure 4-2: 5G vs 4G Polygon

4.2 4G LTE (Long Term Evolution)

Third Generation Partnership Project (3GPP) planned for the long-term evolution of UMTS by designing the network to deliver high data rates and lower latencies for future needs. Figure 4-3 shows the consequential evolved architecture of 3G UMTS. In the new architecture, the Evolved Packet Core (EPC) replaces the packet switch domain from UMTS/GSM and enables enhancement on data traffic by supporting more throughput, mobility, and more efficient service continuity, nevertheless it does not have a direct equivalent for the circuit switching domain. The lack of circuit switching domain requires additional setup for the voice calls in the LTE network; therefore, the IP Multimedia subsystem (IMS) has been integrated with LTE to provide voice-related services in the 4G networks. The IP Multimedia subsystem (IMS) replaces the voice calls on the traditional circuit-switched-style network with an IP packet-switched network, and it also supports the handover functionality between the circuit switch domain to an IP domain to ensure mobility and roaming between base stations. Similarly, the UMTS terrestrial radio access network (UTRAN) is replaced with an “evolved” UMTS terrestrial radio access network (E-UTRAN) that manages the radio communication access to the 4G system from the mobile phones. The mobile phone also refers to as User Equipment (UE), though its internal operation is very different from before (Cox, 2012).

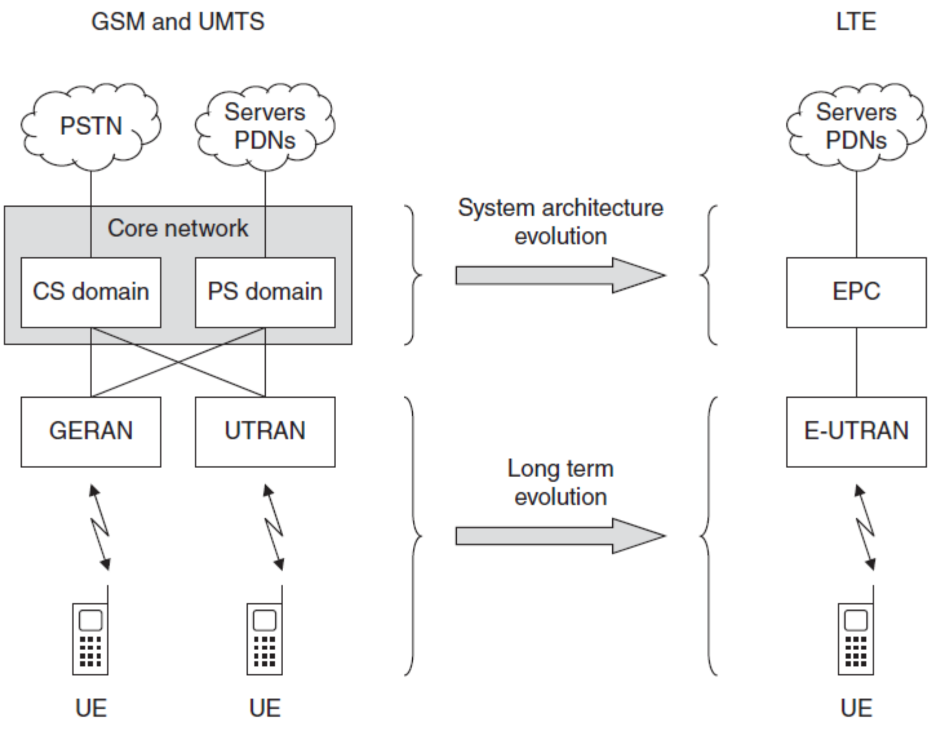


Figure 4-3: 4G LTE Architecture Evolution (Cox, 2012)

The EPC is designed on the principles of the Internet, which transports packets that originate from any device via any application software to the destination without having the inspection of the packet and behavior of the receiving application (as described previously within the OSI model). The LTE acts as a data pipe, and the responsibility of such pipe is to simply transport information to and from the user; the network is not concerned with the application, or the information being held. This concept was innovative compared to the previous traditional circuit-switched networks, in which the voice application is an integral part of the system, therefore, resulting in limitations and restrictions on communication devices and applications. In other words, the 4G LTE is the first IP-based mobile network. Figure 4-4 represents a simplified LTE network providing access to a high-speed Internet for the end user. Short Messaging Service (SMS), voice and other telephony services are provided via an IMS network connected to the Evolved Packet Core (EPC).

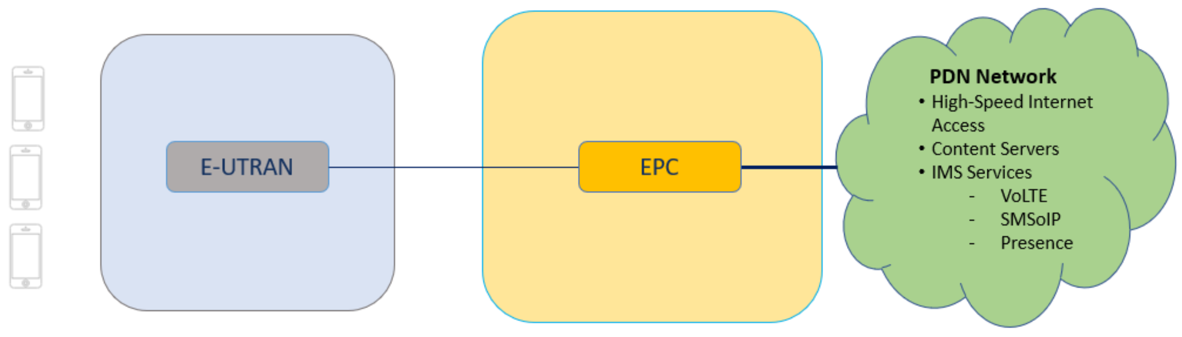


Figure 4-4: LTE Data Traffic Pipe

4.2.1 4G LTE – Access Network

Figure 4-5 shows the components of LTE networks. The E-UTRAN comprises of new base station concept called Evolved Node-B (eNodeB), which provides wireless radio access to the UEs and includes radio channel modulation/demodulation as well as channel coding/decoding and multiplexing/ de-multiplexing. The eNodeB hosts all RRC (Radio Resource Control) functions such as broadcast of system information and RRC connection control, including:

- Paging of subscribers,
- Establishment, modification, and release of RRC connections involving the allocation of temporary UE identities (Radio Network Temporary Identifier (RNTI)),
- Initial security activation, which means the first configuration of the Access Stratum (AS) integrity protection for the control plane and AS ciphering for both control plane and user plane traffic,
- RRC connection mobility that includes all types of intra-LTE handover (intra-frequency and inter-frequency handovers for roaming between base stations).,
- Establishment, modification, and release of DRBs (Dedicated Radio Bearers) carrying user data,
- Quality of Service (QoS) control to ensure that, for example, user plane packets of different connections are scheduled with the required priority for Downlink (DL) transmission and that UEs receive the scheduling grants for Uplink (UL) data transmission according to the QoS parameters of the radio bearers.

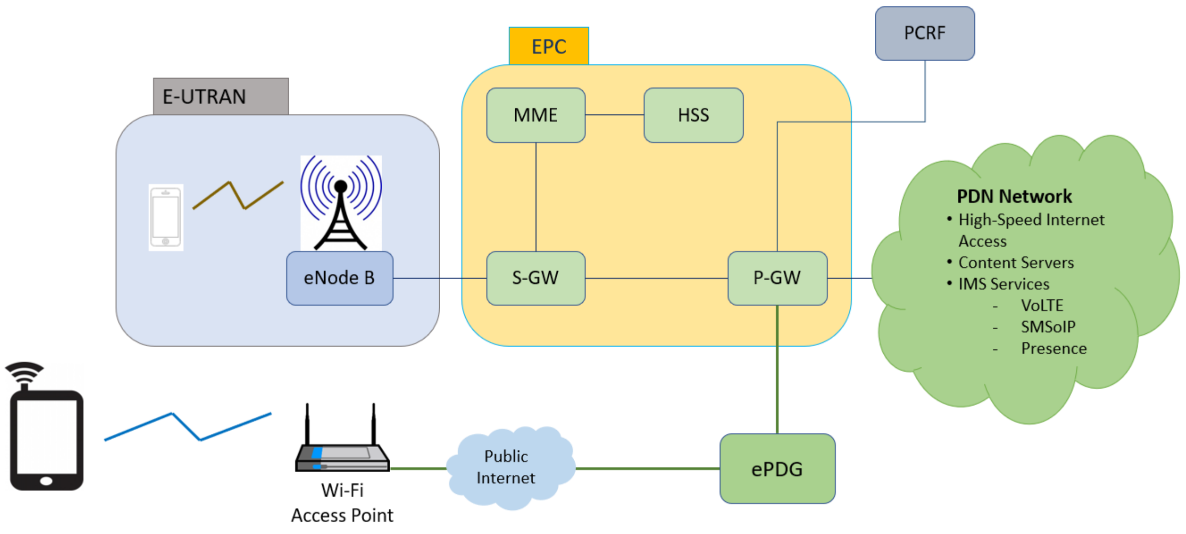


Figure 4-5: LTE Network Components

The LTE networks support access via 3GPP and non-3GPP technologies. User Equipment (UE) can reach the EPC via 3GPP defined access E-UTRAN, but also supports reachability via non-3GPP accesses. Non-3GPP refers to access types that were not specified in the 3GPP, such as Wireless Local Area Network (WLAN) or fixed networks (which are IEEE specifications). Non-3GPP is further segregated into trusted and untrusted categories. Trusted non-3GPP accesses can interact with an EPC directly, whereas untrusted non-3GPP accesses can interact with EPC via Evolved Packet Data Gateway (ePDG). The ePDG provides seamless connectivity between the two accesses and enables the security mechanisms such as tunnelling of connections with the UE over an untrusted non-3GPP access (3GPP-TS.24.302, 2022).

4.2.2 4G LTE – Packet Core Network

Evolved Packet Core (EPC) includes multiple components as illustrated in Figure 4-5 under the EPC block. The key functions of each part are listed below:

- **Mobility Management Entity (MME):** The MME provides 3GPP standards-based mobility management and control functions including intra-LTE, intra-3GPP mobility, security procedures, Adaptive Congestion control, intelligent tracking area management and session management. (Kozina, Soós and Varga, 2016)
- **Home Subscriber Server (HSS):** The HSS is a central database that holds user-related and subscription-related information. The functions of the HSS include functionalities such as mobility management, call and session establishment support, user authentication and access authorization. (Kozina, Soós and Varga, 2016)
- **Serving Gateway (S-GW):** The role of SGW is to manage user plane mobility and acts as a terminating point between RAN and the core network. It also acts as a mobility anchor or gateway when a UE moves between eNBs hence providing the key feature like mobility management/anchor point, packet routing and forwarding, inter-operator charging, transport level packet marking in the uplink and the downlink, inter-eNodeB handover, and inter Radio Access Technology (inter-RAT) handover. (Kozina, Soós and Varga, 2016)
- **Packet Gateway (P-GW)**: The PGW connects the UE to the external packet data networks and serves as a mobility anchor or gateway during mobility events. The PGW is also responsible for billing, Lawful Intercept (LI), and internet protocol (IP) address allocation hence supplying the key feature like multiple Packet Data Networks (PDNs) management, bearer context management, generation of charging records for PDN connectivity, static and dynamic policy management, and IP address allocation. (Kozina, Soós and Varga, 2016)
- **Policy Control and Resources Function (PRCF):** PCRF manages policy making and control decisions. The key functions are delivering Quality of Service (QoS) information to packet gateway, dynamically managing & controlling data sessions, enforcing minimum QoS parameters and assigning charging policy for packets. (Kozina, Soós and Varga, 2016)

4.2.3 4G LTE – Bearers

LTE transports data packets using the same protocols that are used on the Internet; however, the transport mechanisms are more complex because LTE must ensure mobility as a connected device moves from one base station to another and expects to maintain its connection without any interruption. Secondly, the UE supports running multiple applications in parallel to provide different services, and each service will require different Quality of Service (QoS) depending on the service requirements. For example, UE can be engaged in a Voice over IP (VoIP) call while at the same downloading software in the background. In such a case, VoIP will expect QoS in terms of latency and jitter, while the downloading process requires a much lower packet loss rate. Therefore, LTE can offer QoS guarantees and can assign different qualities of service to different data streams and to different users. LTE transports data from one part of the system to another using Evolved Packet System (EPS) bearers. An EPS bearer can be thought of as a bi-directional data pipe, which transfers data on the correct route through the network and with the correct quality of service (Cox, 2012).

Bearers can be segregated into two categories based on provided QoS: Minimum Guaranteed Bit Rate (GBR) bearers and non-GBR bearers. The former has an associated GBR value for which dedicated transmission resources are permanently allocated at bearer establishment. Higher bit rates than the defined GBR are allowed for the bearer if resources are available. In contrast, non-GBR does not guarantee any bit rate, and no bandwidth resources are allocated permanently to the bearer. Each bearer has an associated Class Identifier (QCI) and an Allocation and Retention Priority (ARP). The QCI is characterized by priority, packet delay budget, and acceptable packet loss rate, and the QCI label for a bearer decides the way it is managed in eNodeB. QCIs have been standardized so that vendors can all have the same understanding. The standardization ensures that LTE operators can expect uniform traffic handling behavior throughout the network, regardless of the manufacturers of the eNodeB equipment (Palat and Godin, 2011).

During the LTE Attach process, a "*Default Bearer*" is created for the purpose of UE to perform network signaling, and a default Bearer is created with non-GBR QCI; hence it provides best-effort service. Each default bearer comes with an IP address, and it is assigned a QCI of 5 – 9. On the contrary, a "*Dedicated Bearer*" provides a reserved tunnel to one or more specific traffic types (such as VoIP or video). A dedicated bearer acts as an additional bearer on top of the default bearer. The dedicated bearer can be GBR or non-GBR (whereas the default bearer can only be non-GBR). A dedicated bearer uses Traffic flow templates (TFT) to give special treatment to specific services. Figure 4-6 shows the LTE QoS with respect to the default and dedicated bearer, along with the supported QCI.

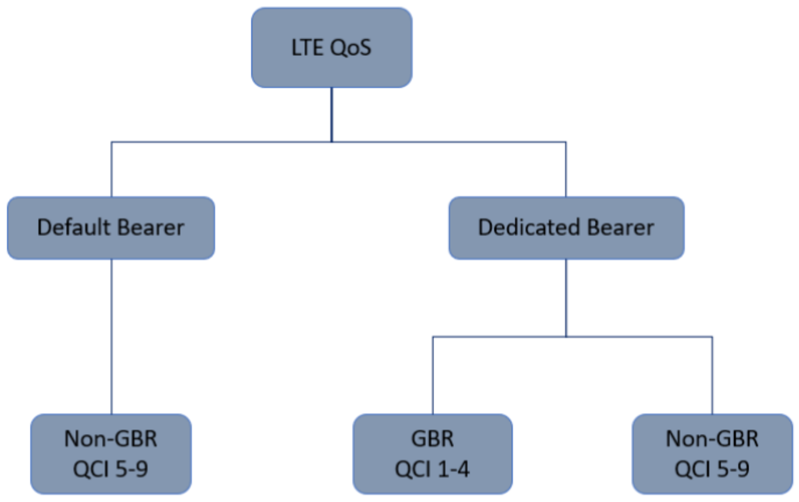


Figure 4-6: LTE Bearer and Quality of Service

An EPS bearer must cross multiple interfaces for end-to-end establishment, as shown in Figure 4-7. Across each interface, the EPS bearer is mapped onto a lower layer bearer, each with its own bearer identity. Bearers are identified by the GPRS Tunnelling Protocol (GTP) tunnel ID across interfaces and each node must keep track of the binding between the bearer IDs across both different interfaces. An S5 / S8 bearer transports the packets of an EPS bearer between a P-GW and an S-GW. The S-GW stores a one-to-one mapping between an S1 bearer and an S5 / S8 bearer. An S1 bearer transports the packets of an EPS bearer between an S-GW and an eNodeB. A radio bearer transports the packets of an EPS bearer between a UE and an eNodeB. An E-UTRAN Radio Access Bearer refers to the concatenation of an S1 bearer and the corresponding radio bearer. An eNodeB stores a one-to-one mapping between a radio bearer ID and an S1 bearer to create the mapping between the two. The overall EPS bearer service architecture is shown in Figure 4-7 (Palat and Godin, 2011).

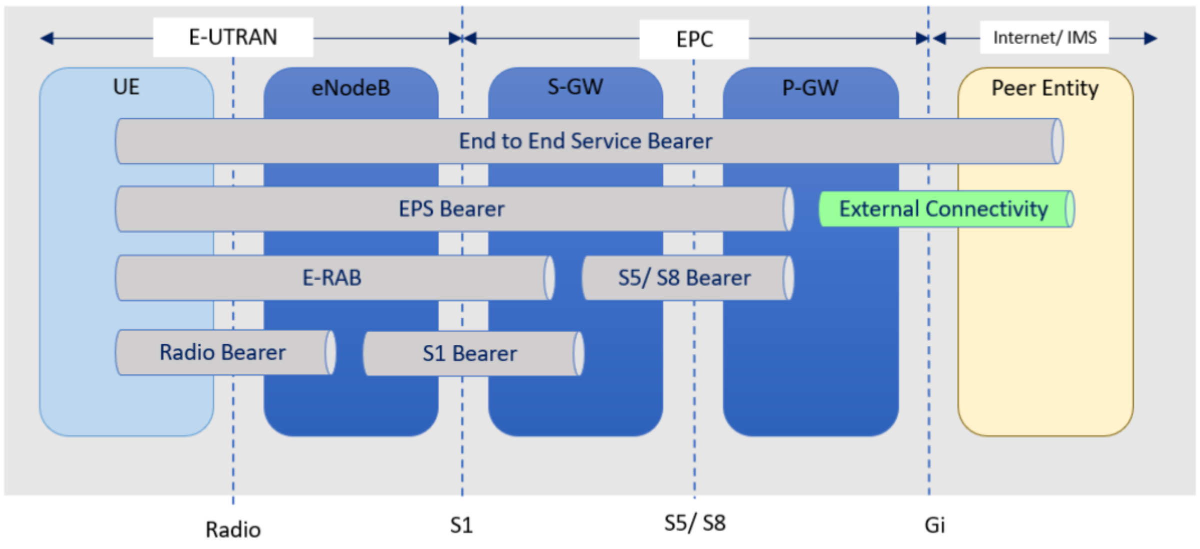


Figure 4-7: Overall EPS Bearer service Architecture

4.2.4 4G LTE – Interfaces

LTE defines various interfaces to communicate between the different network components. The interfaces are associated with the protocol stack which enables the network elements to exchange data and signaling messages between the different LTE constituents. Figure 4-8 shows the LTE constituents and the interfaces over which it can exchange information with the other network elements. The protocol stack has two planes: user and control plane. Protocols in the user plane handle data that are of interest to the user, while protocols in the control plane handle signaling messages that are only of interest to the network elements themselves (Cox, 2012).

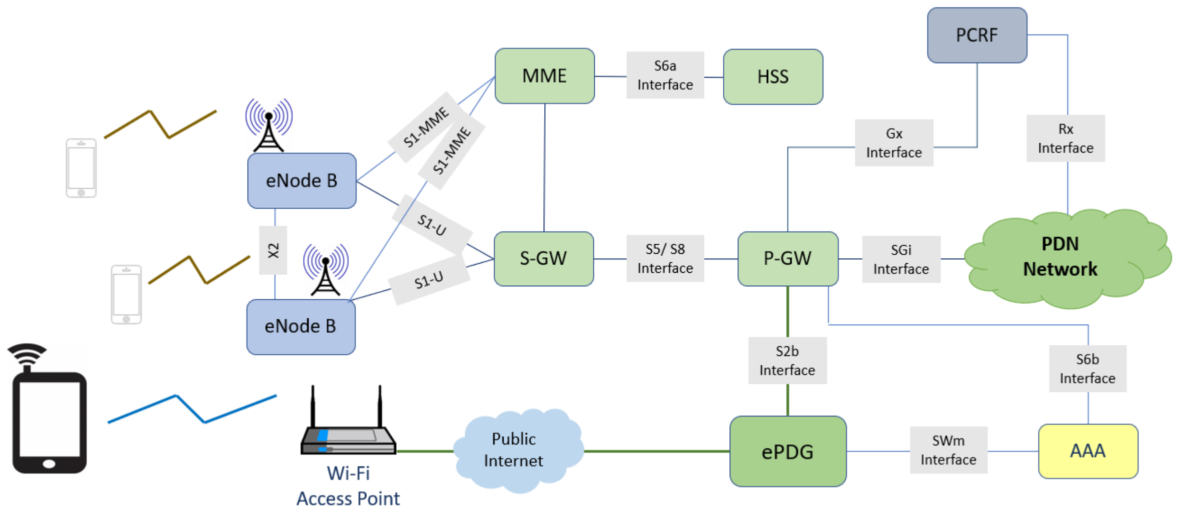


Figure 4-8: LTE Network Interface Summary

4.2.5 LTE Architecture Limitation

There are also challenges and drawbacks related to LTE architecture, which has escalated the need for enhanced architecture. Few of the known challenges are as follows:

- **Interfaces:** The number of interfaces and protocols defined for LTE to establish the communication in LTE network are manifold.
- **Interoperability:** The Application Programming Interface (API) is closed between the nodes, which has resulted in data request implementation differently from different vendors. This has raised the interoperability issue and limitation on the network to be part of vibrant and robust ecosystem.
- **Flexibility:** Gateway selection is inflexible, as only one access gateway (SGW) and one packet gateway (PGW) must be present between communication from the UE to the Packet Data Network.
- **Complexity:** The session management is complex due to the QoS, charging server selection or the identification of level of service. A Domain Name System (DNS) is typically used for gateway selection, and this complicates the flows and implementation.
- **Scalability:** Application deployed in LTE networks are not cloud native and even lack virtualization capabilities, therefore scaling of application is highly dependent on the HW. Only SW scale out become challenge in LTE network.

4.3 5G System Architecture

The 5G network architecture design supports various applications and services. It defines an architecture that takes advantage of user and control plane separation to allow greater flexibility in handling heterogeneous devices on multiple access types for applications. In the next sections of this subchapter, we explain the 5G core (5GC) architecture by using 3GPP standards as reference.

The 5G System architecture can be described in two ways according to the 3GPP standards. One is a service-based representation, in which the control plane network functions access each other's services. The other is a reference point representation, in which the interaction between the network functions is shown with a point-to-point reference argument (Lei *et al.*, 2020).

4.3.1 5G System Architecture – Core Network

The 5G Core architecture represents a mobile core network that is responsible for session management, authentication, service continuity, and security. Within the 5G Core Network architecture, since there are no pre-defined interfaces, one or more network functions (NFs) can be chained to create new services, which enables rapid creation and deployment of new services. According to the European Telecommunications Standards Institute (ETSI) and 3GPP, the 5G system architecture is represented by Service-Based Architecture (SBA) or reference point architecture (point-to-point-based architecture) (3GPP-TS.23.501, 2022).

The Figure 4-9 depicts the 5GC service-based architecture, where every network service is disaggregated into its own independent function communicating over a common communication bus. The service-based architecture enables flexible and efficient network management. This architecture minimizes dependencies between the Access Network (AN) and Core Network (CN). In a SBA, services can register themselves and subscribe to other services. This enables flexible development of services and connections to other components without introducing new interfaces.

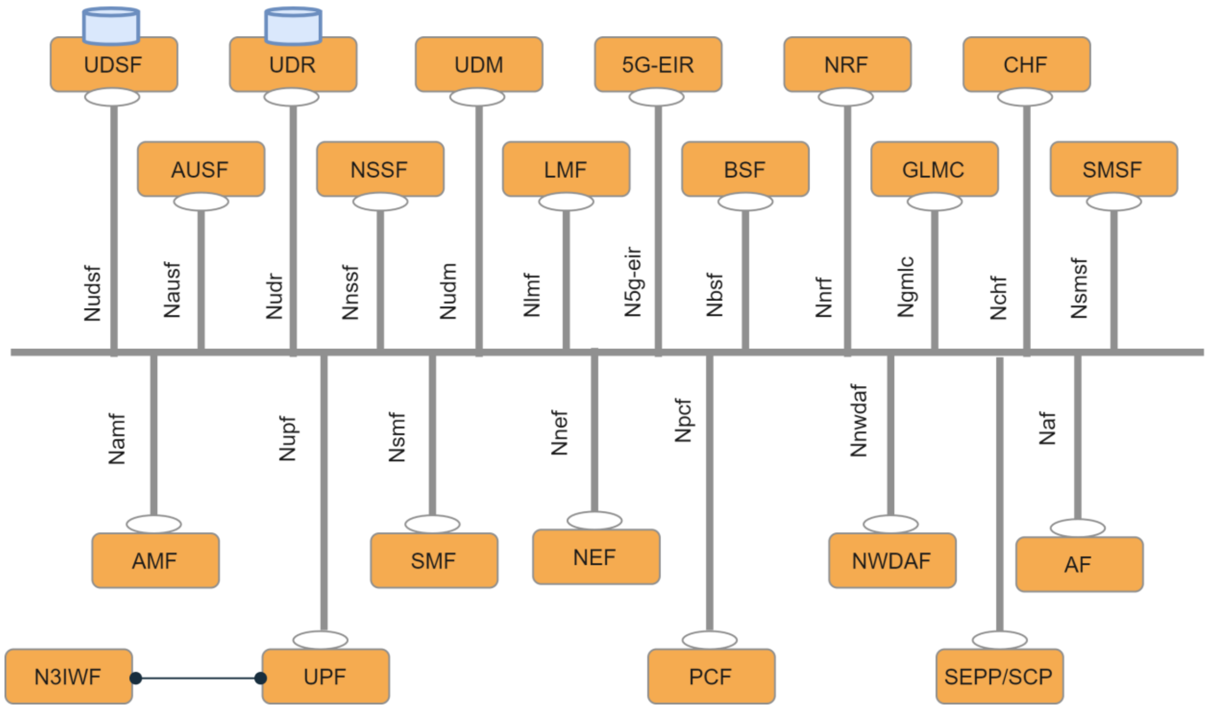


Figure 4-9: 5GC Service-Based Architecture

Communication within a 5G Core is enabled via a service framework, which involves service registration, authorization, and discovery. The Network Functions (NF) within the 5G Core will only use Services Based Interfaces (SBI). Every NF using SBI can potentially interact with any other NF directly by using SBI. All other NFs are not supporting SBI, or outside the trust domain communicate via the Network Exposure Function (NEF) or via Message Routing Forwarding Function (MRFF), which is a proxy-like function to support point-to-point (P2P) on the Northbound interface. In addition, NFs can support (if required) an extension/plugin framework to enable customization, extension, and enrichment. This enables the 5GC NFs to interwork with any custom APIs of multiple vendors or third-party applications. (ETSI 123 501- V15.5.0 -5G, 2019)

Within the 5G Core architecture, new services can be created by combining and reusing existing network functions in a dynamic manner. This enables rapid creation and deployment of new services. A Service Based Architecture can be described as a toolbox of capabilities that can be orchestrated, optimized, and sliced to support a wide variety of services as well as fulfill specific operator's needs. Service Based Architectures (SBA) have become an important enabler for network simplification, automation, and operational efficiency. (ETSI.129.500, 2018)

In Service Based Architectures, the NF service producer exposes the service to other authorized NF service consumers through a Service Based Interface (SBI). The interaction between two network functions (consumer and producer), within this NF service framework, follows the request-response and subscribe-notify mechanism. The consumer functions ask the producer functions for services that the producer functions expose. This is a one-to-one communication between consumer and producer NFs. Producer NFs can use the service of one or more NFs, in which case this NF acts as a consumer NF. (ETSI.129.500, 2018)

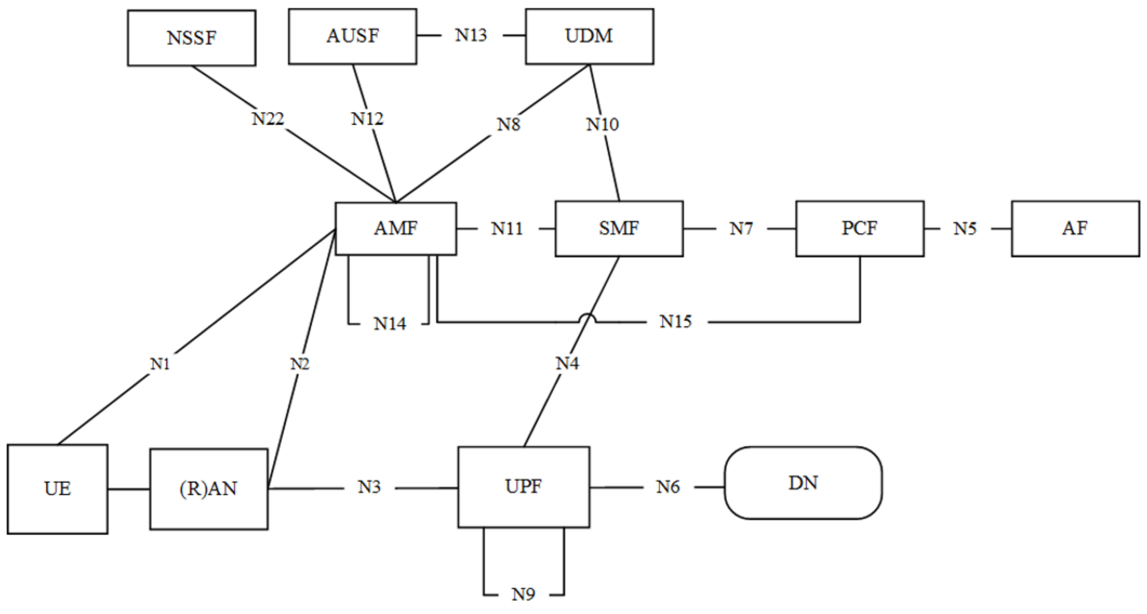


Figure 4-10: 5GC Service-Based Interfaces (3GPP-TS.23.501, 2022)

The Figure 4-10 shows the 5GC reference point architecture where different network functions are connected over standardized interfaces, in other words, the interaction between pairs of network functions is defined by a point-to-point reference point. In this architecture the user plane functions are connected over point-to-point links. For example, the N4 reference point connects the network function SMF and UPF (TS 123 501- V15.5.0 - 5G, 2019).

In the Figure 4-10, the UPF is in the user plane and all other NFs, i.e., AMF, SMF, AUSF, and UDM, are in the control plane. Separating the user and control planes allows each plane resource to be scaled independently. This separation also allows the UPFs to be deployed separately in a distributed fashion. In this architecture, the UPFs can be placed close to the UEs to shorten the Round-Trip Time (RTT) between UEs.

The 5G Core architecture takes advantage of the user and control plane separation. The Control and User Plane split at software level enables flexible network deployment and operation, by distributed or centralized deployment and independent scaling between the control plane and the user plane functions. This makes the network highly flexible and highly programmable, supporting the most effective use of key technologies such as Cloud and Software-Defined Networking (SDN). This is also a crucial feature which will enable distribution of services to the edge for low-latency applications.

4.3.1.1 5GC Network Functions

The 5G core Network contains network functions defined by 3GPP. The main network functions with their respective support and features are listed below:

- **Access and Mobility Management Function (AMF):** The AMF supports Non-Access Stratum (NAS) termination, NAS ciphering and integrity protection, connection management, registration management, mobility management, access authentication, reachability management, access authorization, security context management (SCM), and applying mobility related policies from the Policy Control Function (PCF) such as mobility restriction. (3GPP-TS.23.501, 2022)
- **Application Function (AF):** The AF supports operation such as application influence traffic routing, accessing the Network Exposure Function (NEF) for resource retrieval, exposure of services to end users, and interaction with the policy framework for policy control. (3GPP-TS.23.501, 2022)
- **Authentication Server Function (AUSF):** The AUSF is an authentication server. Enables mutual authentication between the user equipment and the network. (3GPP-TS.23.501, 2022)
- **Network Repository Function (NRF):** The NRF supports the service discovery function, maintains the NF profile, allows other NF instances to subscribe to it, gets notified about a NF status, and the registration in NRF of new NF instances. (3GPP-TS.23.501, 2022)
- **Network Exposure Function (NEF):** The NEF provides the secure mechanism to expose services and features of the 5G Core network, control plane parameter provisioning, translation of internal and external information, and secure provisioning of information from an external application to a 3GPP network. (3GPP-TS.23.501, 2022)
- **Network Slice Selection Function (NSSF):** The NSSF supports selecting the network slice instances to serve the UEs or devices by determining the configured and allowed different Network Selection Assistance Information Slice (NSSAI), as well as the AMF set to be used to serve the UE. (3GPP-TS.23.501, 2022)
- **Network Data Analytics Function (NWDAF)**: NWDAF collects data from all Network Functions (NFs), Application Functions (AFs), operations, administration, and maintenance systems. The NWDAF supports machine learning model training and provisioning as well. (3GPP-TS.23.501, 2022)

- **Policy Control Function (PCF):** The PCF provides policy rules for control plane functions such as network slicing, roaming and mobility management, enforcement of QoS and charging policies. (3GPP-TS.23.501, 2022)
- **Session Management Function (SMF):** The SMF supports session management (session establishment), device IP address allocation, Dynamic Host Configuration Protocol (DHCP) IPv4 and IPv6 functions, termination of session management parts of NAS messages, selection & control of User Plane Function (UPF) instances, traffic steering at UPF, downlink data notification, roaming functionality, and lawful interception for the control plane. (3GPP-TS.23.501, 2022)
- **Unstructured Data Storage Function (UDSF):** The UDSF supports the storage and retrieval of information as unstructured data by other 5G core network functions. (3GPP-TS.23.501, 2022)
- **Unified Data Management (UDM):** The UDM supports the generation of authentication credentials, access authorization, user identification handling, Short Messaging Service (SMS) management, and subscription management. (3GPP-TS.23.501, 2022)
- **Unified Data Repository (UDR):** The UDR supports storage and subscription data by the UDM, storage and subscription of policy data by the PCF, storage, and retrieval of structured data. (3GPP-TS.23.501, 2022)
- **User Plane Function (UPF):** The UPF supports PDU session anchoring, packet flow processing such as routing and forwarding, packet inspection, Deep Packet Inspection (DPI), QoS control, GPRS Tunnelling Protocol - User (GTP-U) path management, UE inactivity detection and reporting, policy enforcement, lawful interception, and TCP optimization. (3GPP-TS.23.501, 2022)

4.3.1.2 Service Based Architecture (SBA)

One major change in the 5G Core architecture compared to previous generations is the introduction of the service-based architecture. In the service-based architecture, a common framework has been introduced, which enables the Network Functions (NF) to expose their services for use by other network functions. In the 5GC architecture model, the interfaces between the network functions are referred to as Service Based Interfaces (SBI). The Service Framework defines the interaction between the NFs over SBI using a Producer–Consumer model. As such, a service offered by a NF (Producer) could be used by another NF (Consumer) that is authorized to use the service. The services are referred to as "NF Services" in 3GPP specifications (ETSI.129.500, 2018) (Lei *et al.*, 2020).

The NFs interact with each other via a "Request-response" or a "Subscribe-Notify" mechanism. In the "Request-response" model, NF (consumer) requests from another NF (producer) to provide a service and/or perform a certain action. In "Subscribe-Notify" model, the NF (consumer) subscribes to the services offered by another NF (producer), which notifies the subscriber of the result as shown in Figure 4-11 (Lei *et al.*, 2020).

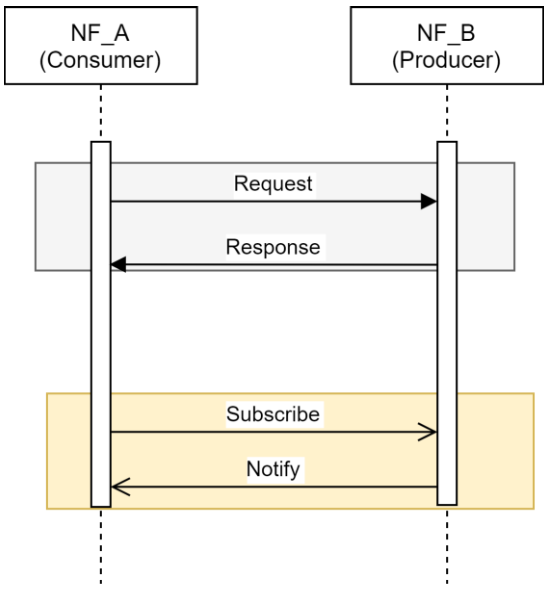


Figure 4-11: NF Service illustration (direct)

As per 3GPP (3GPP-TS.23.501, 2022; 3GPP-TS.23.502, 2022), there are three main defined procedures associated with the service framework: (Lei *et al.*, 2020)

- **NF service registration and de-registration**: Responsible for informing the Network Repository Function (NRF) about the NF instances and supported services.
- **NF service discovery**: Enables a Consumer NF to discover a Producer NF instance(s) that provided the expected service(s).
- **NF service authorization**: Ensures the authorization of the NF service consumer to access the NF service provided by the NF service producer.

A. Access and Mobility Management Function

AMF is the operator's network gatekeeper. It ensures the mobile subscriber is authenticated and authorized by its home operator before it is registered and offered access to network services. During the subscriber registration, AMF establishes a security association with the User Equipment (UE) to secure all future NAS (Non-Access Stratum) signaling communication between the UE and AMF.

After a UE is successfully registered, AMF begins to manage the ongoing state and mobility events for the UE to maintain reachability for constant access to services. AMF also facilitates UE's communication with other 5GC NFs, e.g., SMF, to establish one or more connections with different 5QIs based on user request and operator authorization.

I. AMF Service-Based Representation

Figure 4-12 represents the 5G core network service-based architecture and denotes the role of AMF thus. The AMF offers services through a Namf service-based interface. The Namf - AMF exposes its services to other NFs via the Namf service. AMF provides the following services (3GPP-TS.29.518, 2022):

- Namf_Communication service allows AMF and other NFs to communicate with UE and RAN over N1 and N2

- Namf_EventExposure service enables an NF to subscribe to UE related event notification on its behalf or on behalf of another NF.
- Namf_MT services allow an NF to request capabilities[3] information to send Mobile Terminated signaling or data to UE, e.g., paging in idle state.
- Namf_Location service allows an NF to receive UE location information, e.g., local time zone, event info related to emergency call, etc.

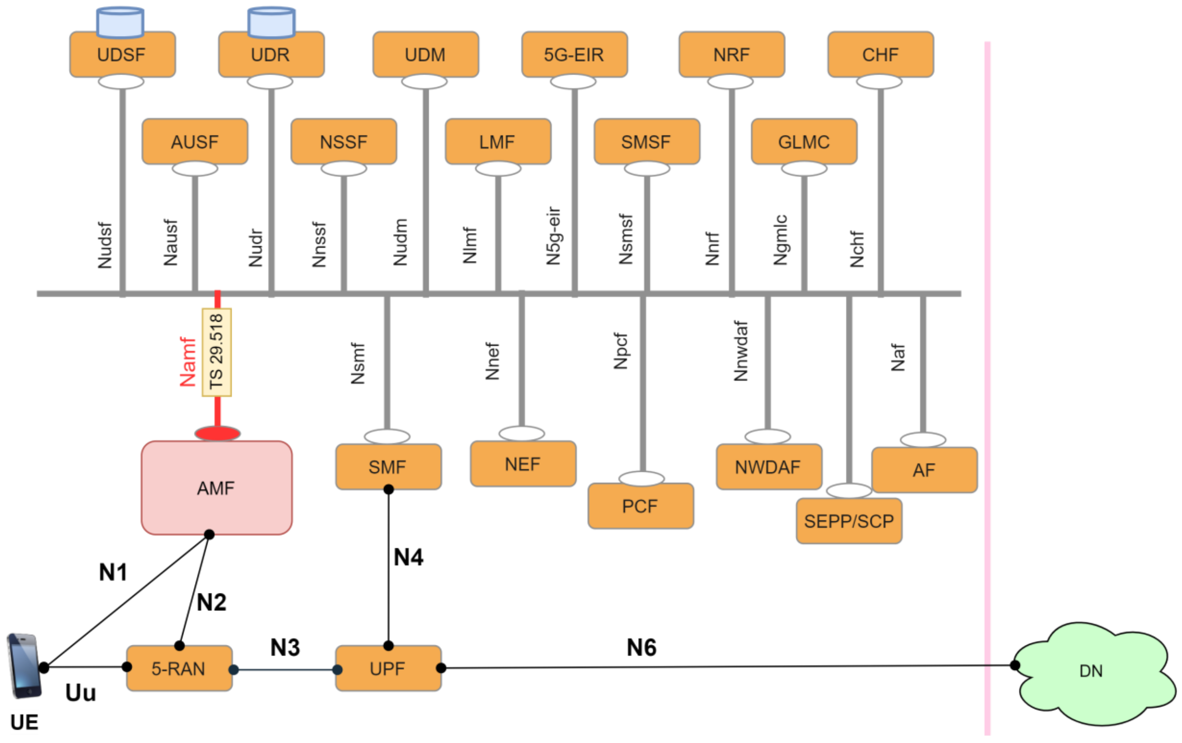


Figure 4-12: AMF Service-Based Representation

II. AMF Reference Point Representation

Figure 4-13 represents the AMF reference point architecture. The AMF supports the following reference point for communication (3GPP-TS.23.501, 2022):

- N1 - Reference point for SMS transfer between UE and AMF. The N1 reference point is used by the protocols for:
  - mobility management between the UE and AMF for both 3GPP and non-3GPP access.
  - session management between the UE and SMF for both 3GPP and non -3GPP access.
- N2 - Reference point between NG-RAN and AMF. The N2 reference point is used by the protocols for:
  - delivery of signaling messages between AMF and NG-RAN
- N8 - Reference point between UDM and AMF
- N11 - Reference point between SMF and AMF
- N12 - Reference point between AUSF and AMF

[3] Capabilities refers to the possibility of a UE to connect with certain parameters (radio frequency like Sub-6 spectrum and/or mmWave, if it supports SA or NSA modes, etc.)

- N14 - Reference point between two AMF
- N15 - Reference point between PCF and AMF
- N22 - Reference point between NSSF and AMF

We can observe the complete AMF reference point representation in detail in Figure 4-13 as per the 3GPP TS 23.501 specification (3GPP-TS.23.501, 2022):

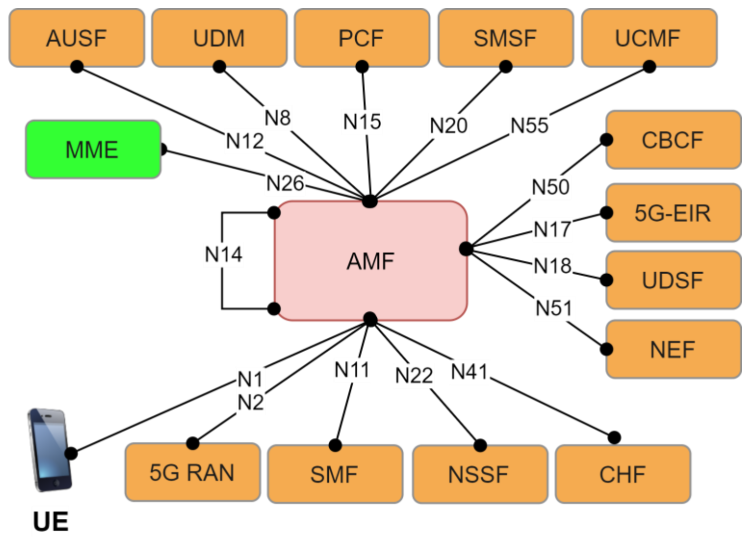


Figure 4-13: AMF Reference Point Representation

III. AMF Key Functions

AMF supports the following functions and services (3GPP-TS.23.501, 2022):

- SMF discovery and selection - AMF allocates an SMF instance to manage the UE PDU session(s),
- User Authentication - AMF allocates an AUSF instance to perform UE authentication with the home operator,
- Transports SMS between UE and Short Messaging Service Function (SMSF),
- NAS ciphering and integrity protection,
- UE reachability and connection management,
- AMF controls the UE mobility within a same 5G access or across different 3GPP and non-3GPP accesses,
- Lawful interception (for AMF events and interface to LI System[4]),
- Access Authorization & UE policy support via PCF,
- Supports Security Anchor Functionality (SEAF) for authentication process between a UE and its home network.
- Security Context Management (SCM): SCM receives a key from SEAF to use for deriving access-network specific keys,
- Location services management for regulatory services,
- Transport Location Services messages between UE and Location Management Function (LMF) and between RAN and LMF,
- EPS Bearer ID allocation during interworking with EPS via N26,

[4] LI system allows law enforcement agencies to selectively wiretap individual subscriber, this action can be performed on approval from court.

- Multiple PLMN support for Network Sharing,
- Network slicing support.

IV. AMF Key Features

Some of the notable features supported by AMF are listed below (3GPP-TS.23.501, 2022):

a. Permanent identifiers:

A globally unique Subscription Permanent Identifier (SUPI) is allocated to each subscriber for 5G-based services. The International Mobile Subscriber Identity (IMSI) and the network access identifier (NAI) are valid SUPI types. A UE supporting NG-RAN includes a Subscription Concealed Identifier (SUCI) when a valid 5G Globally Unique Temporary Identity (5G-GUTI) is not available from the PLMN or an equivalent PLMN to which the UE is attempting to register or if requested by the network, during the registration procedure. The SUCI is a privacy preserving identifier containing the concealed SUPI. Each UE supporting NG-RAN contains a permanent equipment identifier (PEI) for accessing 5GS-based services.

AMF supports SUPI, SUCI with “null-scheme” and PEI as the permanent identities.

b. Temporary identifiers:

A 5G Globally Unique Temporary UE Identity (5G-GUTI) is allocated to each subscriber for 5G-based services. The purpose of the GUTI is to provide an unambiguous identification of the UE that does not reveal the UE or the user's permanent identity.

The 5G-GUTI has two main components:

- one that identifies the AMF(s) which allocated the 5G-GUTI; and
- one that uniquely identifies the UE within the AMF(s) that allocated the 5G-GUTI.

AMF supports allocation and use of 5G Globally Unique Temporary Identity (GUTI) for the UE.

c. AUSF Selection:

AMF selects an AUSF to perform authentication between the UE and 5G CN in the HPLMN. The following factors are considered during the AUSF selection:

- SUPI.
- Home network identifier (e.g., MNC and/or MCC) of SUCI.

d. NRF Selection:

AMF selects the configured default NRF when NSSF is not configured or when NSSF does not return the IP/Fully Qualified Domain Name (FQDN) for the NRF.

e. NSSF Selection:

AMF supports NSSF selection based on the configured FQDN. AMF requests the configured DNS for NSSF FQDN. If NSSF FQDN is not configured, then AMF uses the configured default NSSF IP address.

f. UDM Selection:

AMF selects a UDM to manage the user subscriptions in the HPLMN.

g. PCF Interaction:

AMF interacts with the PCF to obtain Access Management (AM) policy information and with PCF for UE policy related information. The AMF configuration selects the PCF based on:

1. NRF: Here the AMF selects the PCF via NRF discovery procedure.
2. Local PCF configuration: Here the AMF selects a locally configured PCF per PLMN.

B. Authentication Server Function (AUSF)

AUSF is the Authentication Server Function that always resides in the home network, and it is an essential 5G NF in the 5G unified authentication framework. In addition to 5G Authentication and Key Agreement (5G-AKA) for 3GPP access authentication, the AUSF supports Extensible Authentication Protocol-Authentication and Key Agreement (EAP-AKA') for non-3GPP access. In the EAP framework, the Security Anchor Function (SEAF), which is collocated with the AMF, is considered a pass-through authenticator and the AUSF is a backend authentication server. Although, these role descriptions come from the Extensible Authentication Protocol (EAP) framework, it is particularly useful to apply the same terminologies when using 5G-AKA.

One of the key architectural advantages of having the AUSF as an independent entity, is to allow the home network to initiate secure communication with the UE using the UE specific home key (K_{AUSF}) without forcing a new UE round of authentication. For example, when the UDM updates the UE parameters. In addition to the UE authentication, the AUSF provides security material to ensure the steering information and allow the home network to update the UE parameters for a registered UE.

I. AUSF Service-Based Representation

Figure 4-14 outlines the position of AUSF in the 5G service-based architecture. The AUSF supports the following services (3GPP-TS.29.509, 2022):

- Nnrf_NFDiscovery service: AUSF uses this NRF service to select and discover the UDM.
- Nudm_UEAuthentication service: AUSF uses this UDM service to request UE authentication.

Additionally, the AUSF provides its Nausf services to other NFs. The AUSF provides the following three services: (3GPP-TS.29.509, 2022)

- Nausf_UEAuthentication service: This allows the consumer NFs, e.g., AMF/SEAF, to authenticate the UE over 3GPP access and non-3GPP access. As part of the output of this service, AUSF provides the master keying material, e.g., anchor key (K_{SEAF}).
- Nausf_SoRProtection service: The UDM uses this service to communicate steering material to the UE during the authentication procedure and to securely steer the UE to other Visited Public Land Mobile Network (VPLMN). The AUSF calculates the Steering of Roaming Message Authentication Code (SoR-MAC-I_{AUSF}) using UE specific home key (K_{AUSF}) along with the steering information received from the UDM and delivers the SoR-MAC-I_{AUSF} and CounterSoR (3GPP-TS.33.501, 2022)

- Nausf_UPUProtection service: the UDM utilizes this AUSF service to update the UE parameters after the UE has successfully been registered and authenticated in the network. The AUSF calculates the UE Parameter Update Message Authentication Code (UPU-MAC-I_{AUSF}) using UE specific home key (K_{AUSF}) along with the UE Parameters Update Data received from the UDM and delivers the UPU-MAC-I_{AUSF} and CounterUPU to the UDM. (3GPP-TS.33.501, 2022)

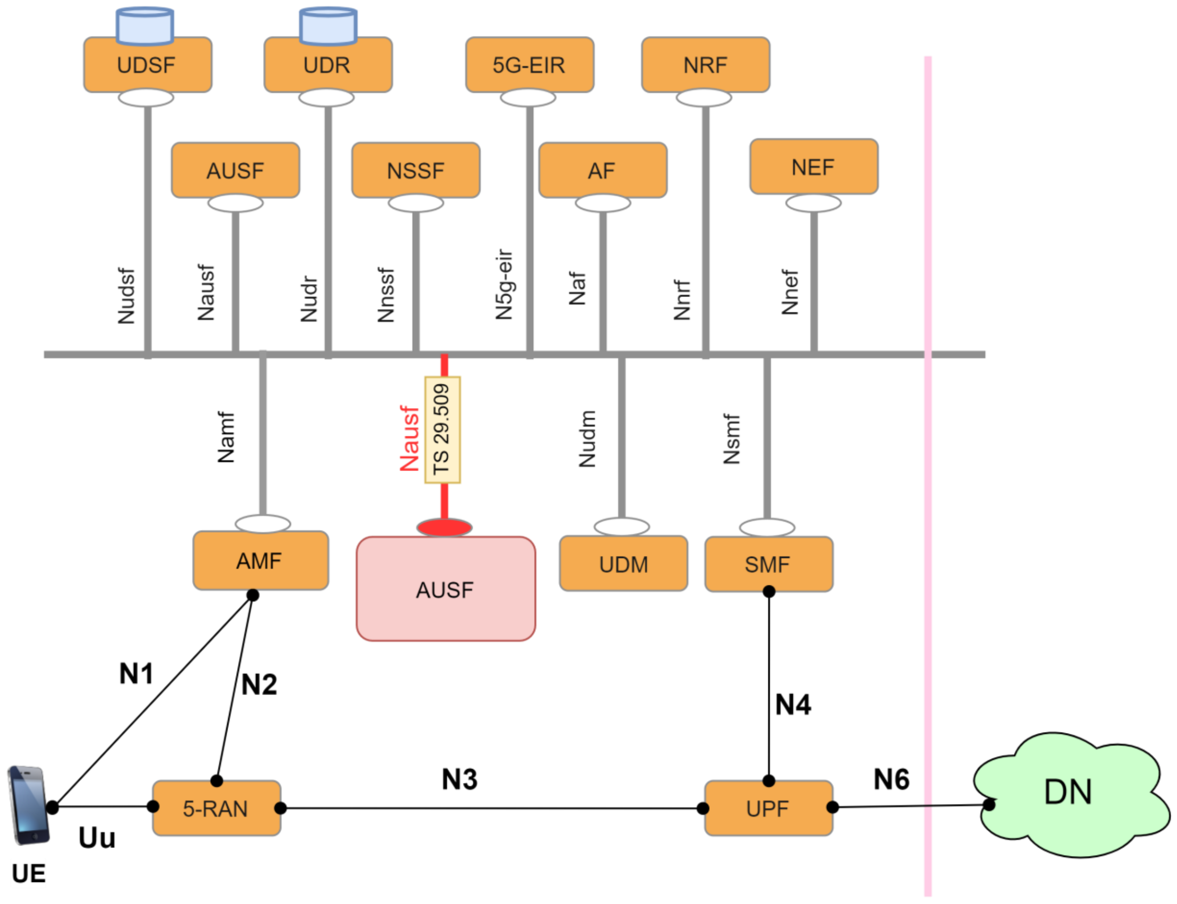


Figure 4-14: AUSF Service-Based Representation

II. *AUSF Reference Point Representation*

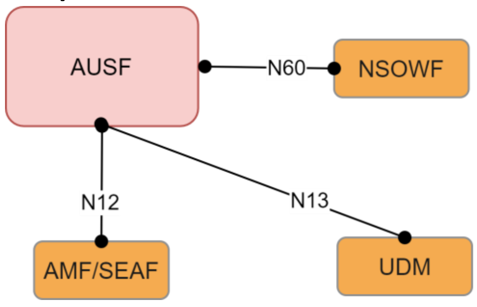


Figure 4-15: AUSF Reference Point Representation

Figure 4-15 represents the AUSF reference point architecture. The interface between the AMF and the AUSF is named N12 and AUSF communicates with UDM on the N13 interface.

III. *AUSF Key Functions*

AUSF supports the following functions and services (3GPP-TS.23.501, 2022):

- 5G-AKA Authentication and Key Agreement,
- EAP-AKA' Authentication and Key Agreement,
- Supports UE authentication over 3GPP access,
- Supports UE authentication over non-3GPP access,
- Delivers the UE SUPI to the serving network only after the UE authentication is complete,
- Supports informing the UDM that a successful or unsuccessful authentication of a subscriber has occurred,
- Maintains the UE KAUSF master key for the duration of the UE registration or until the next UE authentication procedure,
- Supports UDM request for providing secure material to secure the UE steering information during the UE registration procedure,
- Supports UDM request to provide security material to protect the home network, procedure for updating UE parameters during the duration of the UE registration,
- NF selection (UDM) via NRF or local config.

IV. AUSF Key Features

Some of the key features supported by AUSF are listed below (3GPP-TS.29.509, 2022) (3GPP-TS.23.501, 2022):

a. UDM Selection:

AUSF supports UDM selection for both local and through NRF discovery. For UDM selection based on local configuration, AUSF supports configuration of UDM instance IDs with UDM IPs or UDM FQDNs.

AUSF selects UDM through NRF using:

- SUPI,
- Routing ID part of SUCI,
- UDM Group ID,
- Operator defined query parameter (string).

b. Authentication for EAP:

In EAP procedure, the NF Service Consumer requests the authentication of the UE by providing UE related information and the serving network and the EAP-based authentication is then selected. EAP messages are exchanged between a UE acting as EAP peer, an NF Service Consumer (AMF/SEAF) acting as a pass-through authenticator and the AUSF acting as the EAP server. (3GPP-TS.29.509, 2022)

c. Authentication for 5G AKA:

In a 5G AKA procedure, the NF Service Consumer (AMF) requests the authentication of the UE by providing UE related information and the Serving Network Name to the NF Service Producer (AUSF), which retrieves UE related data and authentication method from the UDM. In this case the retrieved authentication method is 5G AKA. The NF Service Consumer (AMF) shall then return to the AUSF the result received from the UE (3GPP-TS.29.509, 2022)

C. Network Repository Function (NRF)

Network Repository Function (NRF) is a standalone NF and provides the base for the services framework defined for the 5G Core. NRF offers services to other NFs for registration and discovery, along with subscription and notification. The network functions register their services with NRF, and other network functions discover the services via the Nnrf service-based interface.

The NRF monitors the service health and updates the service registry. NRF starts monitoring the service health as soon as the service registers itself with NRF.

I. NRF Service-Based Representation

Figure 4-16 represents the NRF function and its position within the 5G service-based architecture. The NRF supports the following services:

- Nnrf_NFManagement: This service allows the Network functions, Service Communicaiton Proxy (SCP) or Security Edge Protection Proxy (SEPP) Instance to register, update or deregister its profile in NRF. Similarly, it supports operations such as subscribe, unsubscribe, or notify on the status of NF or SEPP instance registered in NRF. (3GPP-TS.29.510, 2022)

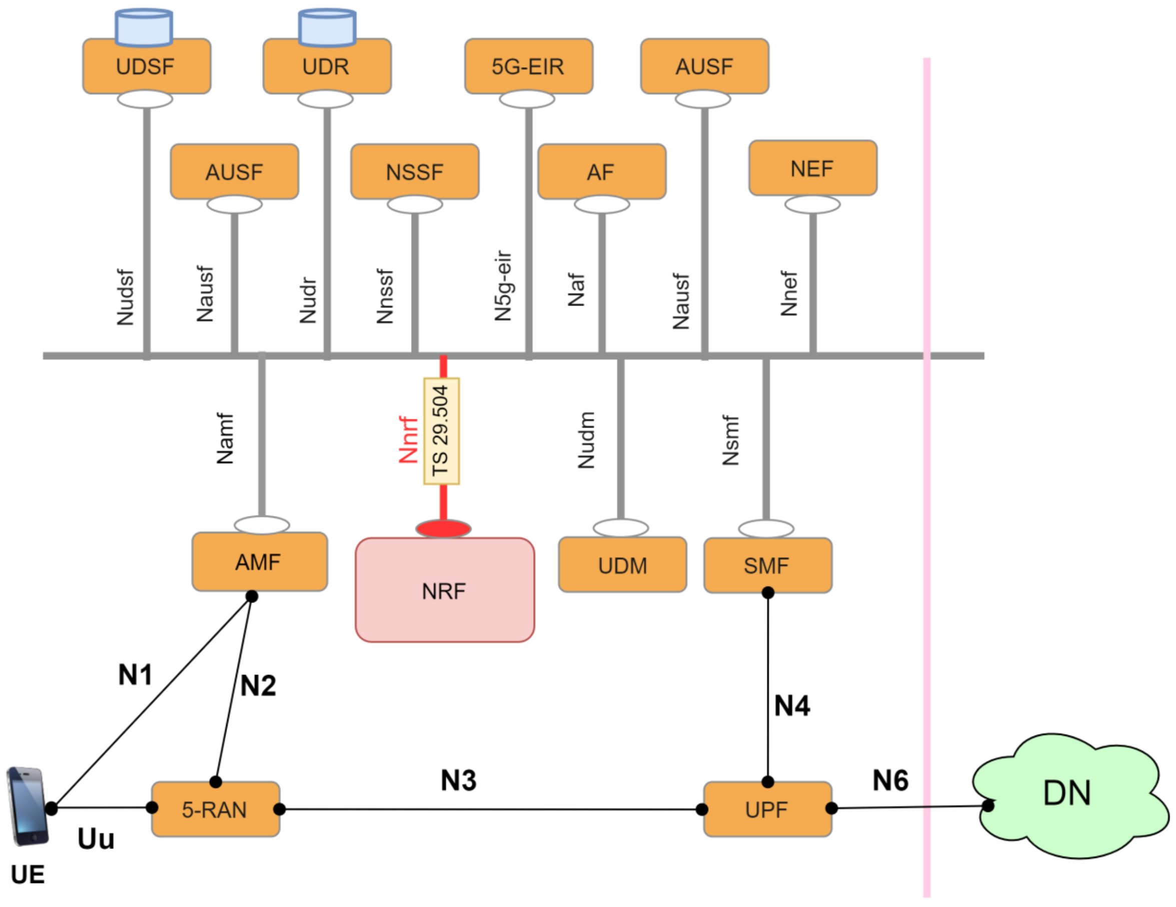


Figure 4-16: NRF Service-Based Representation

- Nnrf_NFDiscovery: This service allows a NF or SCP Instance to discover other NF instances with the offered services, by querying the local NRF.
- Nnrf_AccessToken: NRF offers this service (used for OAuth2[5] authorization) following the "Client Credentials" authorization grant, as specified in the 3GPP specification TS 33.501 (3GPP-TS.33.501, 2022). It exposes a "Token Endpoint" where the Access Token Request service can be requested by NF Service Consumers (3GPP-TS.29.510, 2022).
- Nnrf_Bootstrapping: This service enables the NF Service Consumer to get information on the supported services endpoints by NRF. (3GPP-TS.29.510, 2022)

II. NRF Reference Point Representation

The NRF interacts with every other NF in the 5GC, but it is never depicted in reference point representation figures. In the roaming case, the reference point between the visiting NRF and the home NRF is named as N27 (shown in Figure 4-17). The reference point name of N27 is used only for representation purposes, but its functionality is included in the services offered by the Nnrf Service-Based Interface (3GPP-TS.29.510, 2022)

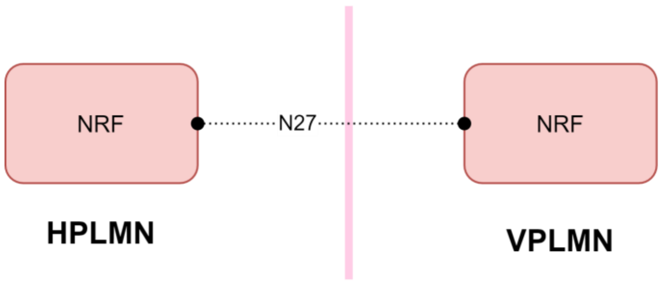


Figure 4-17: NRF Reference Point Representation

III. NRF Key Functions

NRF supports the following functions and services (3GPP-TS.23.501, 2022):

- Maintains the NF profile and services supported by the available NF instances.
- Allows other NF instances to subscribe to, and get notified about, any new NF instances registration in NRF.
- Supports service discovery function. The NRF receives NF Discovery Requests from NF instances and provides the information of the available NF instances.

IV. NRF Key Features

Key features supported by NRF are list below:

a. Health Monitoring:

NRF starts the health monitoring of NF instances immediately after the service registers itself with NRF. During the successful NF registration, the NRF provides a response, mentioning the heartBeatTimer field. The heartBeatTimer field specifies the time, in seconds, between two consecutive heart-beat messages from an NF instance to the NRF (3GPP-TS.29.510, 2022).

[5] OAuth authorization framework enables application to obtain limited access to an HTTP service. (RFC6749, 2012)

Each NF contacts the NRF periodically as mentioned in the heartBeatTimer field. When a NRF detects that the registered NF is not reachable, it updates the service registry and other consumer NFs about the unhealthy NF. In other words, if a NF has not updated its profile for the configured amount of time, the NRF considers the NF deregistered and its services are no longer discovered by the other registered NFs (3GPP-TS.29.510, 2022)

b. Flexible NRF:

The NRF can be a stateless network function that stores information in a database and maintains the NF profile. The NRF application stores the information in Unstructured Data Storage Network Function (UDSF).

c. Customized NF registration:

NRF supports registration of custom (i.e., non-standardized) Network Functions. This implies that NRF accepts registration of NF with unrecognized NF types and stores NF data associated with the custom NFs and returns it during the discovery result.

d. NF heartbeat:

When the NRF detects that a given NF has not updated its profile for a configurable amount of time (longer than the heart-beat interval), the NRF considers the NF as deregistered, and its services can no longer be discovered by other NFs via the NFDiscovery service. To avoid deregistration, the NF uses the PATCH method with the payload body of the PATCH request containing a "replace" operation on the "nfStatus" attribute of the NF Profile at the NF Instance and sets it to the value "REGISTERED". In addition, the NF Service Consumer also provides the load information of the NF, and/or the load information of the NF associated NF services (3GPP-TS.29.510, 2022).

e. Load balancing:

Load Balancing is an optional feature in NRF and can be enabled. When enabled, NRF sends a single producer NF profile to a consumer NF. When disabled, NRF returns all the NF instances in a discovery response.

D. Network Slice Selection Function (NSSF)

A distinct feature of the 5G core architecture is network slicing. Network slicing is a virtual concept where virtual networks are created to offer a specific service or groups of services over a common network infrastructure. In other words, a network slice is a logical network, serving a defined business purpose or customer with all the required network resources configured together.

A single network slice instance can be shared by multiple service instances. The network slice instance consists of none, one, or more sub-network instances shared by other network slice instances. The sub-network instance is a set of NFs, which run on physical or logical resources. The network slice is a complete logical network providing telecommunications services and network capabilities. Network slices vary depending on the services they need to support. Network slicing enables an operator to create logically partitioned networks at a given time, to provide optimized services for different market scenarios and use cases.

The NSSF offers services to the AMF via Nnssf service-based interface. The NSSF registers itself with all configured NRFs in an NRF to-AMF-set mapping. NSSF deregisters itself when an NRF is removed and registers when new NRF is added in the NRF mapping configuration.

I. NSSF Service-Based Representation

Figure 4-18 shows the NSSF service-based representation. Nnssf - NSSF offers services to the AMF via an Nnssf interface. This interface also offers services to the AMF and NSSF in a different PLMN. Some of the key services of network slicing are as follows (3GPP-TS.29.531, 2022):

- **Service isolation:** Network slices are created to provide tailored services to different verticals. Network slice isolation identifies the degree of resource sharing, which could be tolerable to an operator. Service isolation provides security of various levels and independent lifecycle management.

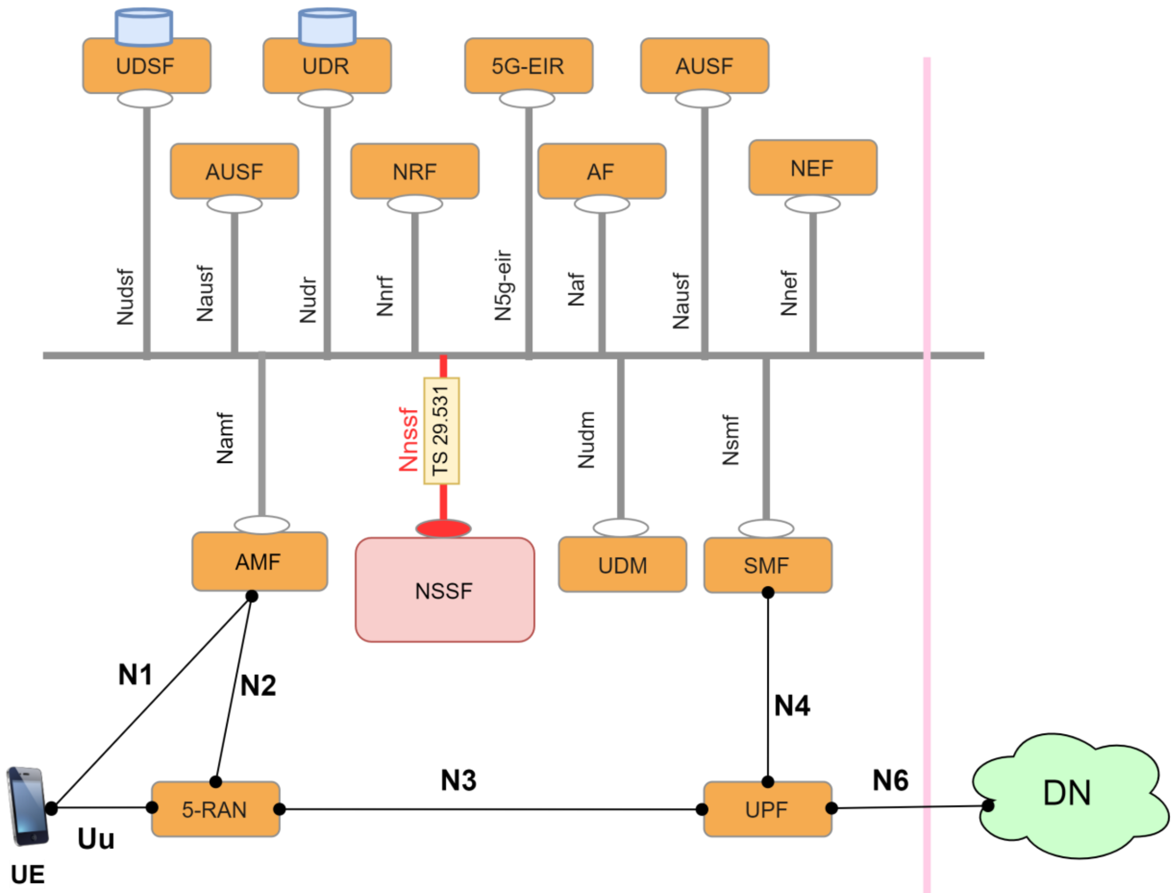


Figure 4-18: NSSF Service-Based Architecture

- **Slice coexistence:** Due to security constraints, the operator may need to prohibit certain set of network slices corresponding to different Single Network Selection Assistance Information Slices (S-NSSAIs) to actively serve the UE simultaneously. The operator needs to identify which set of network slice instances corresponds to an allowed S-NSSAI that can actively serve the UE simultaneously.
- **Reusability:** Reuse the network functions across slices by sharing or deploying dedicated instances of each NF and reuse underlying network resources.

- **Assurance:** Service assurance ensures complete end-to-end network slice performance. Service assurance guarantees that the network functions within a slice are monitored and automatically repaired or healed in case of degradation.
- **Scalability:** Dynamic scaling of the network functions within the slice ensures optimal resource usage. The network slice supports self-provisioning, with every NFs retrieving specific configuration from a centralized configuration store.

The Single Network Selection Assistance Information Slice (S-NSSAI) represents a Network Slice. Network Selection Assistance Information Slice (NSSAI) is a collection of S-NSSAIs, and one NSSAI can consist of a maximum of eight S-NSSAI. An S-NSSAI comprises of:

- **Slice/Service Type (SST)** - Refers to the expected Network Slice behavior that its features and services.
- **Slice Differentiator (SD)** - Is an optional information that allows the Slice/Service type(s) to be able to differentiate amongst multiple Network Slices of the same Slice/Service type.

II. NSSF Reference Point Representation

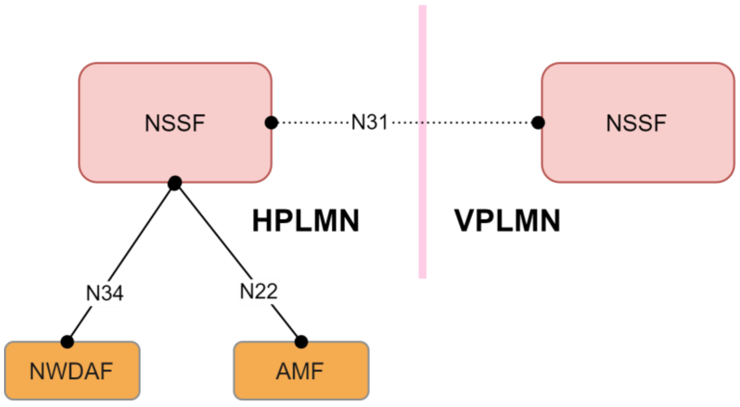


Figure 4-19: NSSF Reference Point Representation

Figure 4-19 represents the NSSF reference point architecture. The interface between the AMF and the NSSF is N22 and the NSSF offers services to different PLMN NSSF on the N31 interface (3GPP-TS.29.531, 2022).

III. NSSF Key Functions

NSSF supports the following functions and services (3GPP-TS.23.501, 2022):

- Selecting the set of network slice instances serving the UE,
- Determining the allowed NSSAI and, if needed, the mapping to the Subscribed S-NSSAIs,
- Determining the configured NSSAI and, if needed, the mapping to the Subscribed S-NSSAIs,
- Determining the AMF set to be used to serve the UE, or based on configuration, a list of AMFs, by querying the NRF.

IV. NSSF Key Features

NSSF supports the following features (3GPP-TS.23.501, 2022) (3GPP-TS.29.531, 2022) :

- NSSF registers with all NRFs configured in NRF-to-AMF-set mapping,
- NSSF deregisters itself if a NRF is removed and registers itself when a new NRF is added in the NRF mapping configuration,
- When a NSSF receives a subscription creation request (POST on nssai-availability/subscriptions), and NssfEventSubscriptionCreateData contains an expiry Information Element (IE), then NSSF checks if the received expiry date and time is earlier than the current date and time in addition to the configured nssai-availability subscription duration,
- If the received expiry time is an earlier date and time, the received expiry time is sent back in the response,
- If the received expiry time is a later date and time, the expiry IE in the response is the current date time +configured nssai-availability subscription duration.

E. Network Exposure Function (NEF)

The Network Exposure Function supports external exposure of capabilities of network functions in the 5G network. As per 3GPP (3GPP-TS.23.502, 2022), external exposure has three distinct categories based on the capability.

- **Monitoring capability:** Monitoring specific events for UEs in the 5G system, such as UE location, loss of connectivity, roaming status, UE reachability. Information of such character is made available for external exposure via the NEF.
- **Provisioning capability:** The provisioning capability enables the external party to provision for specific information which can be used for the UE in the 5G system.
- **Policy/Charging capability:** The policy/charging capability enables external party to manage QoS and charging policies for the UEs, based on the request.

NEF provides access to these network capabilities through homogeneous network Application Programming Interfaces (APIs) defined over a T8 interface. The NEF abstracts the services from the underlying 3GPP network interfaces and protocols.

I. NEF Service-Based Representation

Figure 4-20 shows the status of NEF in the 5GC service-based representation where NEF offers multitudes of services via the Nnef interface. As per the 3GPP TS 29.522 specification (3GPP-TS.29.522, 2022) (3GPP-TS.29.551, 2022) (3GPP-TS.29.591, 2022), the most important services related to NEF are as follows:

- Nnef_EventExposure service provides monitoring events features i.e., info on UE events like loss of connectivity, UE reachability or Location reporting,
- Nnef_Trigger service which helps the AF to send an application trigger (i.e., SMS) to the UE,
- Nnef_BDTPNegotiation service that helps with resource management of background data transfers to a set of UEs,
- Nnef_ParameterProvision service provides support to provision information which can be used by the UE in 5GS,
- Nnef_PFDManagement service provides support for Packet Flow Descriptions (PFDs) management,
- Nnef_TrafficInfluence service provides the ability to shape traffic routing,

- Nnef_ChargeableParty service enables to set the chargeable part for the session upon create request from Application Function (AF),
- Nnef_AFsessionWithQoS service enables the AF to have the connection with a UE with required Quality of Service (QoS),
- Nnef_Location provides the capability to deliver UE location to AF,
- Nnef_AnalyticsExposure provides support for exposure of network analytics.

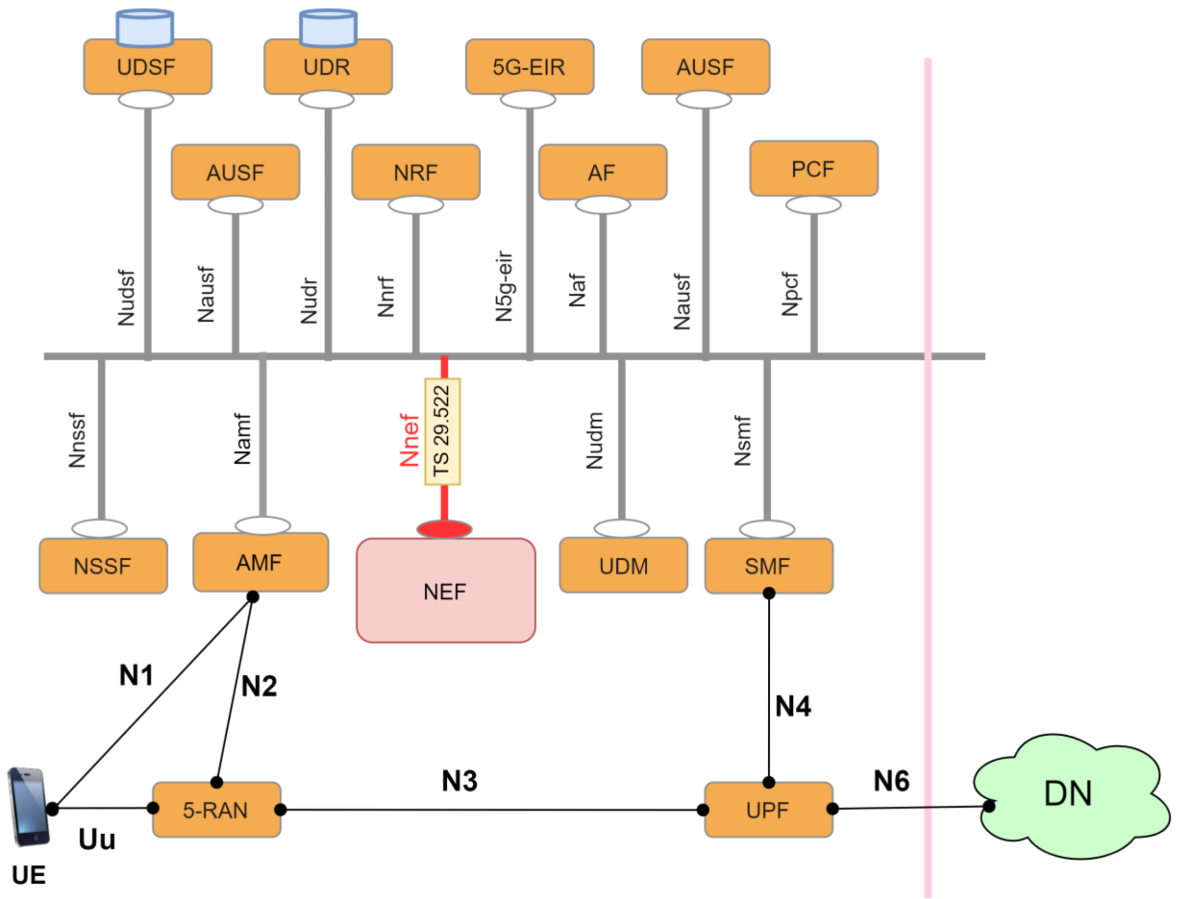


Figure 4-20: NEF Service-Based Architecture

II. NEF Reference Point Representation

Figure 4-21 represents the NEF reference point architecture. The N33 – NEF northbound interface enables the AF to access the services and capabilities provided by all NFs (3GPP-TS.23.501, 2022).

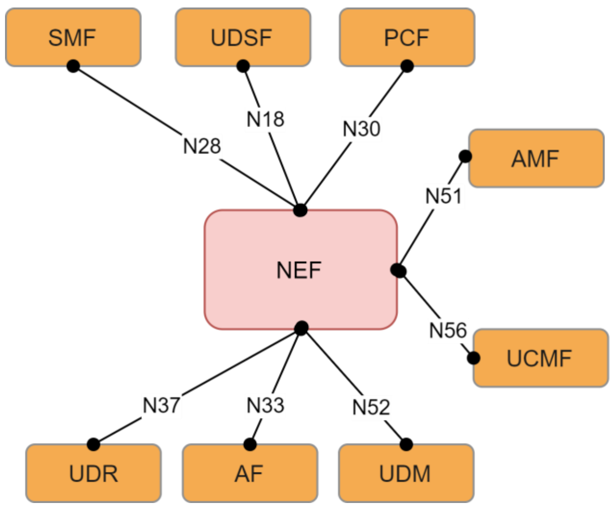


Figure 4-21: NEF Reference Point Representation

III. NEF Key Functions

NEF supports the following functions and services (3GPP-TS.23.501, 2022):

- NEF enables the secure exposure of NF capabilities and events towards third-parties, application functions, and edge computing.
- NEF employs network policies by permitting the masking of network and user sensitive information to external AFs.
- NEF securely provides information to 3GPP networks via application functions.
- NEF facilitates the translation of exchanged information with an AF and information exchanged with the internal network function.
- NEF receives information from other network functions. The information received is stored as structured data in the Unified Data Repository (UDR). Analytics can use the stored information inside the UDR. The stored information can also be accessed and "re-exposed" by the NEF to other network functions.

IV. NEF Key Features

NEF supports the following features (3GPP-TS.23.501, 2022):

- A Monitoring Events feature, which enables monitoring of specific events in 3GPP system and reports such monitoring events information via the NEF,
- Supports monitoring features in roaming scenarios, which requires a roaming agreement between the HPLMN and the VPLMN,
- NEF supports configuring Monitoring Events at the UDM or AMF and thus receives event reports from the UDM and/or AMF.

F. Policy Control Function (PCF)

The Policy Control Function (PCF) provides policy rules for control plane functions. This includes network slicing, roaming and mobility management. PCF accesses subscriber information from the UDR and takes policy decisions. PCF is the evolution of the Policy and Charging Rules Function (PCRF) in 4G LTE.

I. PCF Service-Based Representation

Figure 4-22 represents the position of PCF in a 5GC service-based architecture. PCF offers services through Npcf service-based interface. The PCF supports the following services as per 3GPP specification (3GPP-TS.29.512, 2022) (3GPP-TS.29.514, 2022):

- Npcf_SMPolicyControl provides session related policies.
- Npcf_AMPolicyControl provides access control, network selection and mobility management related policies, and UE Route Selection Policies.
- Npcf_UEPolicyControl provides management of UE Policy Association.
- Npcf_EventExposure provides support for event exposure.
- Npcf_BDTPolicyControl provides background data transfer policy negotiation.

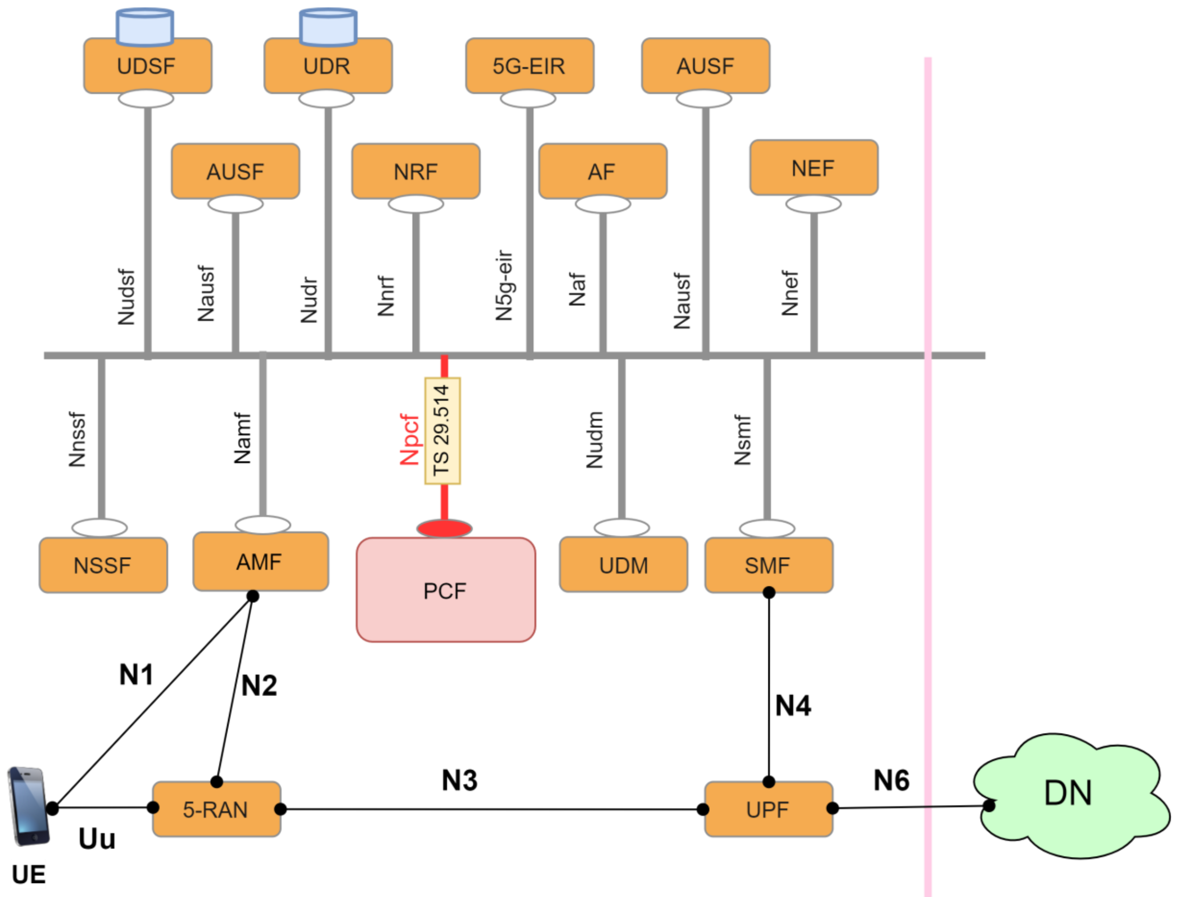


Figure 4-22: PCF Service-Based Architecture

II. PCF Reference Point Representation

Figure 4-23 represents the PCF reference point architecture. The PCF has the following reference points (3GPP-TS.23.501, 2022):

- **N7**: Interface between the SMF and PCF to retrieve the policy information about which SMF provides the appropriate information to the UPF for traffic detection and policy enforcement,
- **N15**: Interface between the PCF and the AMF in the case of non-roaming scenarios,
- **N20**: Interface between the PCF and Charging Function (CHF),

- **N24**: Interface between home network PCF and visiting network PCF,
- **N30**: Interface between the PCF and NEF,
- **N36**: Interface between the SMF and UPF. It supports the Packet Forwarding Control Protocol (PFCP) protocol,
- **N43**: Interface between two PCFs.

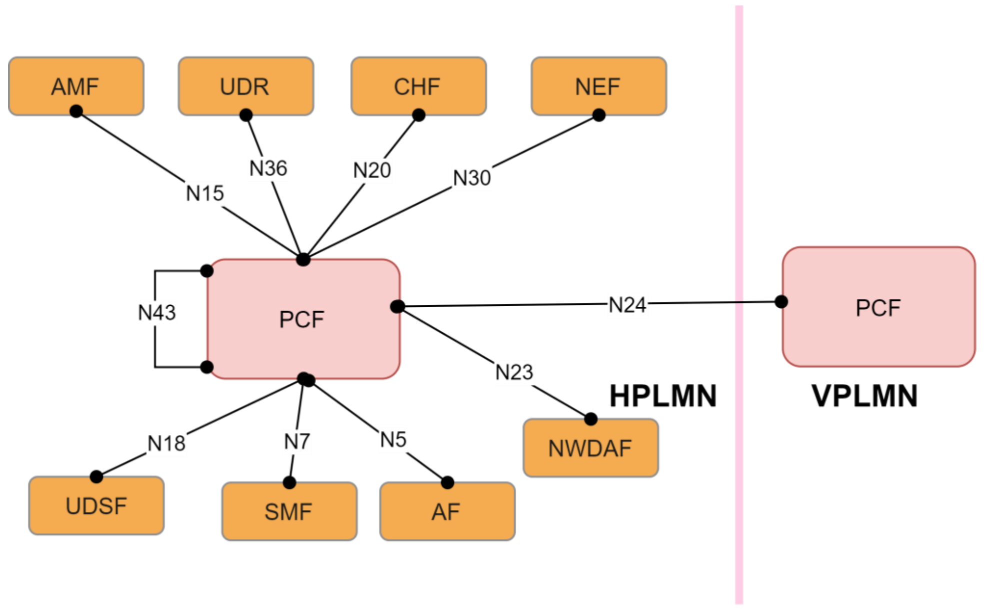


Figure 4-23: PCF Reference Point Representation

III. PCF Key Functions and Features

PCF supports the following functions and services as per the 3GPP specification TS 23.501 (3GPP-TS.23.501, 2022):

- Enables the management of the network behavior with a unified policy framework,
- Provides the network related policy rules to SMF and enforces such rules on the network,
- UE policy management,
- Provides Access and Mobility (AM) policy to AMF,
- Provides Session Management (SM) policy to SMF,
- Data volume monitoring,
- Session Aggregate Maximum Bit Rate (AMBR) control,
- Access subscription information relevant for policy decisions in the UDR.

G. Session Management Function (SMF)

After the UE registers successfully with the network, i.e., with AMF, the UE starts using the NAS connection to request the establishment of one or more sessions. As part of the Protocol Data Unit (PDU) session establishment, the AMF uses S-NSSAI, PLMN ID, and DNN to select a SMF via the NRF or via the combination of S-NSSF and NRF. For SMF to delegate the same PDU session for the length of its lifetime, the AMF ensures one-to-one mapping between the UE PDU session and the SMF. The AMF then forwards the NAS SM messages over the N11 interface for the SMF to establish a PDU session. SMF utilizes either locally configured static address pools or dynamic allocations by using DHCP to allocate an IP address/prefix for each PDU session. SMF may allocate a static IP address via the UDM, based on the subscriber profile. It also selects the Session and Service Continuity (SSC) mode, either by using local configuration or via UDM-based per subscriber profile (ETSI.123.501-R17, 2022).

The SMF also assigns a unique PDU session identifier, selects a single UPF or multiple UPFs, and maintains the CN tunnel information over reference points N3 and N9 if applicable. The SMF uses Packet Forwarding Control Protocol (PFCP) to configure forwarding and Policy and Charging Control (PCC) information at the UPF over reference point N4. SMF also instructs the UPF to proxy or forward all Address Resolution Protocol (ARP) or IPv6 Neighbor Discovery for Ethernet PDU Sessions.

I. SMF Service-Based Representation

Figure 4-24 represents the position of the SMF in a 5GC service-based architecture. The SMF offers services through the Nsmf service-based interface, SMF exposes its Nsmf services to other SMFs and other NFs. SMF provides the following services (3GPP-TS.29.508, 2022) (ETSI.123.501-R17, 2022):

- Nsmf_PDUSession service allows the consumer NFs, e.g., other SMFs and an AMF, to establish, modify and delete PDU session(s),
- Nsmf_EventExposure service exposes the events happening on the PDU session(s) to the consumer NFs, e.g., NEF, PCF, AMF.

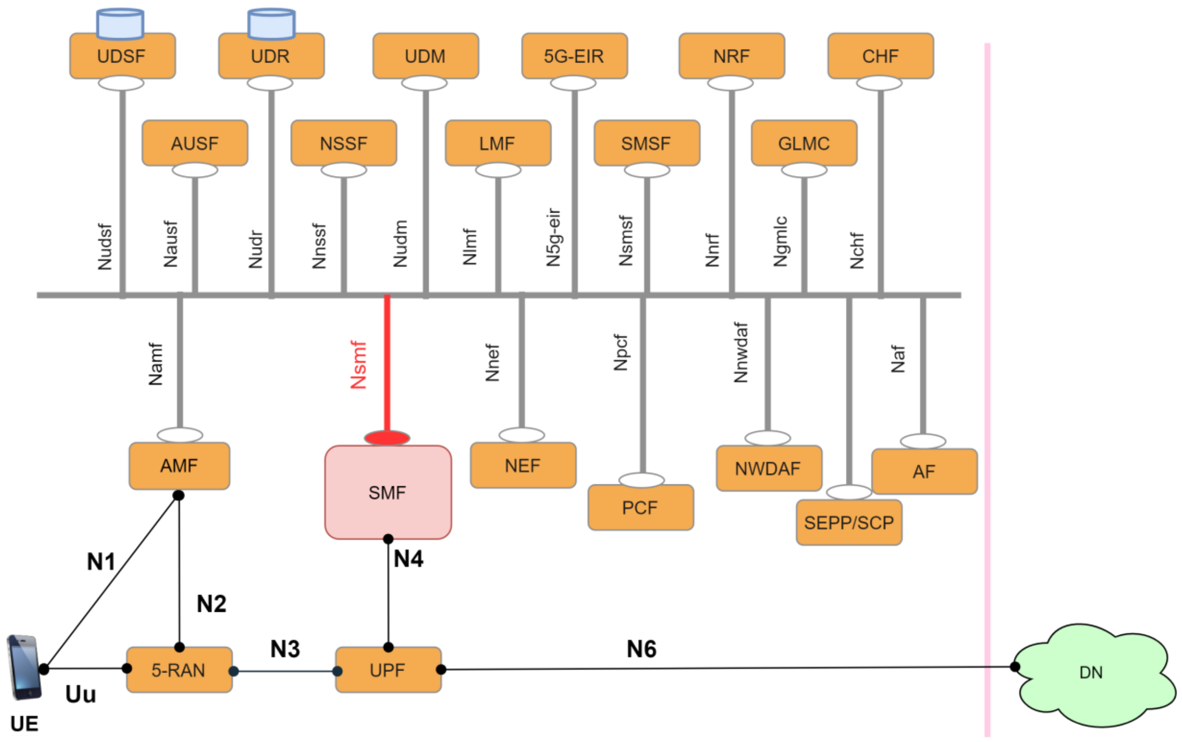


Figure 4-24: SMF Service-Based Architecture

II. SMF Reference Point Representation

Figure 4-25 figure represents the SMF reference point architecture. The SMF supports the following reference point for communication (ETSI.123.501-R17, 2022):

- **N4** - Reference point between SMF and UPF. This reference point is used for provisioning and configuring UPF network functions for the data plane. The data plane configuration information is exchanged between the SMF and UPF,
- **N11** – Reference point between AMF and SMF. This reference point is used to carry SM-related NAS messages from and to the UE (3GPP-TS.29.508, 2022),
- **N29** – Reference point between NEF and a home SMF. This reference point is used to deliver NEF anchored Mobile Terminated (MT) data for a given PDU session of a UE towards the SMF (3GPP-TS.29.542, 2022),
- **N7 –** Reference point between PCF and SMF. PCF uses this reference point to access Management Event Exposure Services at the SMF (3GPP-TS.29.512, 2022).

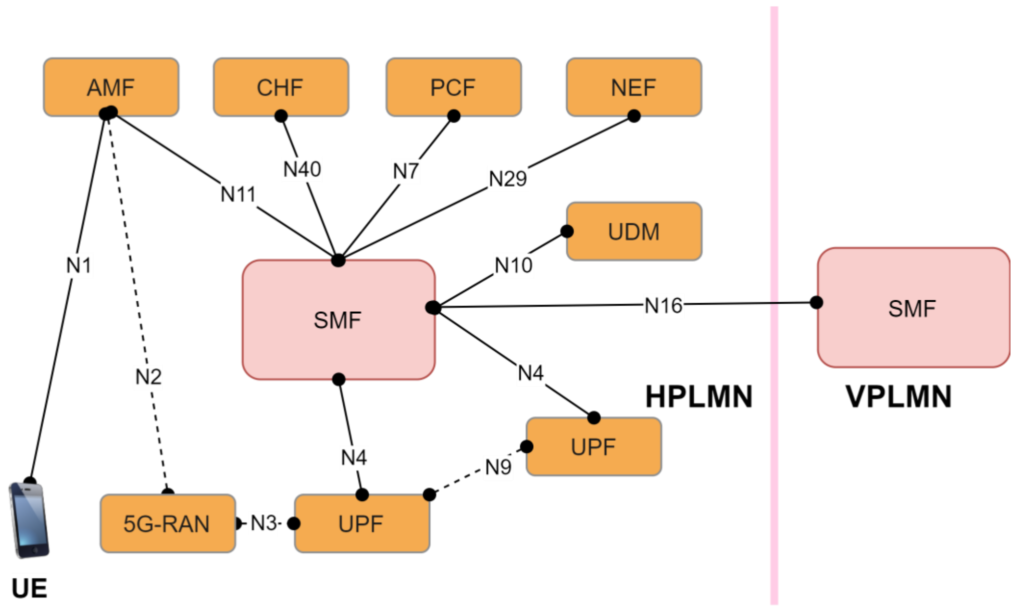


Figure 4-25: SMF Reference Point Architecture

III. SMF Key Functions

SMF supports the following functions and services (3GPP-TS.23.501, 2022):

- Session Management, i.e., session setup, modification & release, including maintaining CN tunnel info between a UPF and a 5G-AN node,
- UE IP address allocation & management,
- DHCPv4 and DHCPv6 (server and client) functions,
- ARP proxying for the Ethernet PDUs,
- Selection and control of user plane functions,
- Configuring traffic steering at UPF to route traffic to proper destinations,
- Termination of interfaces towards policy control function,
- Lawful interception,
- Charging data collection and support of charging interfaces,
- Control and coordination of charging data collection at UPF,
- Termination of SM parts of NAS messages,
- Downlink data notifications,
- Initiator of AN specific SM information, sent via AMF over N2 to AN,
- Roaming functionality,

- Enforcement of QoS,
- Supports interaction with external DNs for transport of signaling for PDU Session authorization/authentication by external DNs.

IV. SMF Key Features

SMF supports the following features (3GPP-TS.23.501, 2022):

a. UDM Discovery Function:

SMF performs UDM discovery to manage the user subscriptions in the HPLMN. The UDM discovery function utilizes the NRF to discover the UDM instance(s) and select a UDM instance based on the obtained UDM information.

b. UPF Selection Function:

SMF performs UPF discovery through NRF or via local provisioning.

c. UL Classifier UPF selection for a PDU Session:

In the case of PDU sessions of type IPv4 or IPv6 or IPv4v6, the SMF may decide to insert the data path of a PDU session an "UL CL" (Uplink classifier). The UL CL is a functionality supported by an UPF that aims at diverting (locally) traffic based on traffic matching filters provided by the SMF. SMF decides and controls the insertion and removal of an UL CL by using generic N4 and UPF capabilities.

d. UE IP Address Management:

SMF supports PDU Session Type "IPv4" or "IPv6" or "IPv4v6".

e. Change of PDU Session Anchor (PSA):

The SMF decides to change the original PDU session anchor (IP anchor point PSA) for a PDU session UE based on the following:

1. When SMF finds a better suited UPF to service the present UE due to UE mobility events or due to load conditions of the PSA, or
2. When it finds that re-allocating the UPF might be beneficial in cases such as:
    - Due to UE moving out of area served by the SMF,
    - Due to overload condition,
    - Due to Operations and Maintenance (O&M) intervention.

f. Converged Charging:

Converged charging is a process where online and offline charging combines. The charging information is utilized by Charging Function (CHF) in one converged charging service, which offers online charging with a quota, and offline charging without quota management, as well as generation of charging information records.

g. Roaming Detection and Multiple PLMN Support:

SMF supports multiple PLMN ID configurations. Each of the configured PLMN IDs is treated as a home PLMN-ID.

h. Paging Policy Differentiation:

Depending on the operator's configuration, a paging policy differentiation feature allows the AMF to apply different paging strategies for different traffic or service types, provided within the same PDU session.

H. Unified Data Management (UDM)

Unified Data Management (UDM) offers services to the AMF, SMF, SMSF, NEF and AUSF within the 5G Core. 3GPP defines the UDM as stateless or stateful. In case of a stateless UDM, all subscriber and dynamic data is stored in the UDR. Therefore, the UDM will make use of the Nudr services for data management, whereas a stateful UDM will store all the information in the local memory.

I. UDM Service-Based Representation

Figure 4-26 represents the UDM service-based architecture. The UDM offers services through Nudm service-based interface, UDM exposes its Nudm services to other NFs. Some of the main services provide by UDM are following (3GPP-TS.29.503, 2022) (ETSI 123 501-V15.5.0 -5G, 2019):

- Nudm_SubscriberDataManagement service allows the consumer NFs (AMF, SMF, NEF) to retrieve, subscribe, unsubscribe, modify or notify information related to the UE's individual subscription,

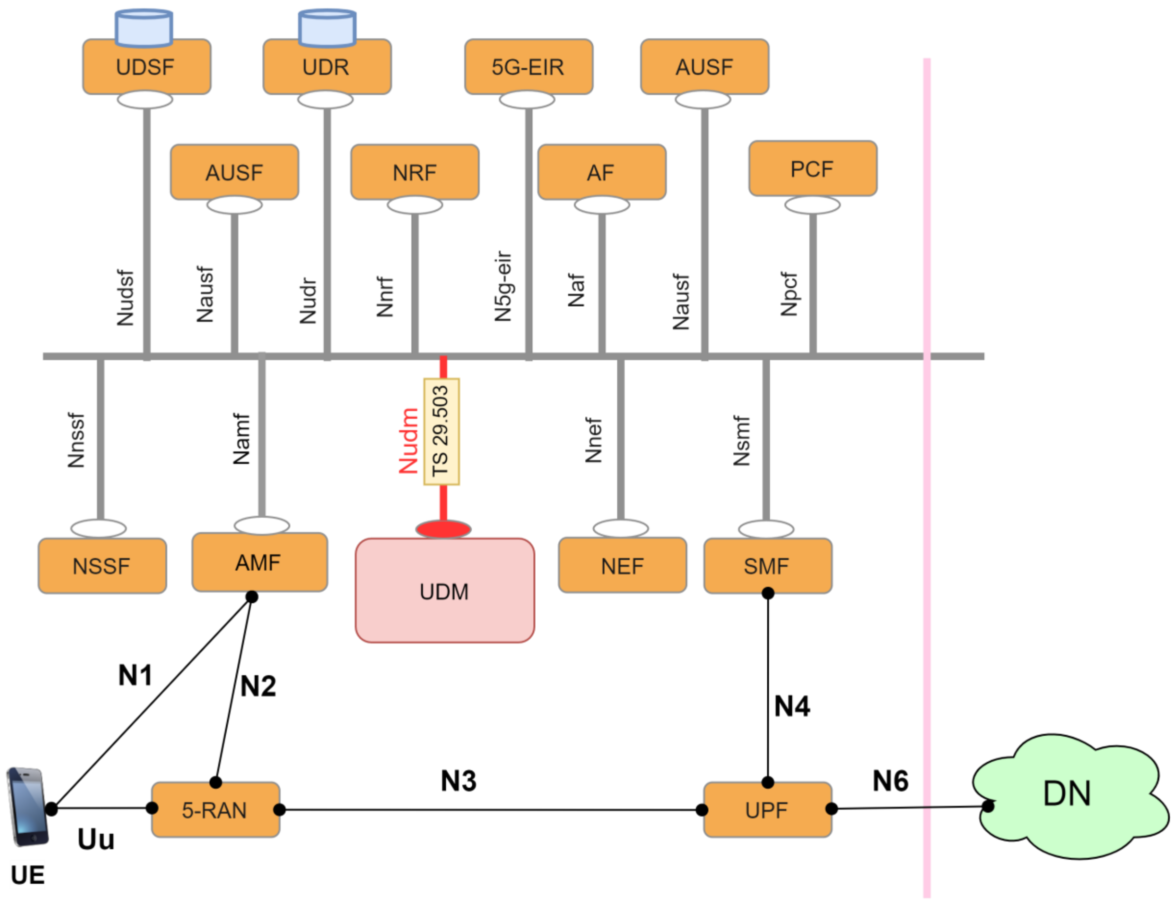


Figure 4-26: UDM Service-Based Architecture

- Nudm_UEContextManagement service exposes the UDM to consumer NFs to determine the provided service based on the subscription in the registration method,
- Nudm_UEAuthentication service allows the AUSF to select the authentication method,
- Nudm_EventExposure service allows the NEF to subscribe, unsubscribe, notify, or modify events related to the other NFs,
- Nudm_ParameterProvision service allows consumer NFs such as NEF to update subscription data for a UE or group of UEs.

II. UDM Reference Point Representation

Figure 4-27 represents the UDM reference point architecture. The UDM supports the following reference point for communication (ETSI 123 501- V15.5.0 -5G, 2019):

- **N8**: Reference point for SMS subscription data retrieval between AMF and UDM,
- **N10**: Reference point between the UDM and the SMF,
- **N13**: Reference point between the UDM and Authentication Server Function (the AUSF),
- **N35**: Reference point between UDM and UDR,
- **N52**: Reference point between NEF and UDM.

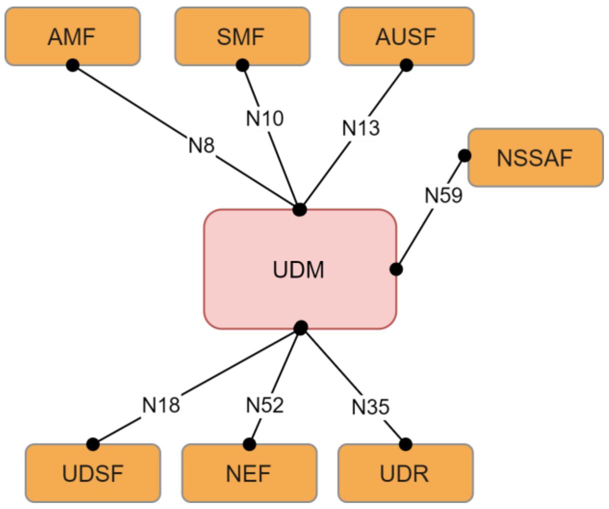


Figure 4-27: UDM Reference Point Architecture

III. UDM Key Functions and Features

UDM supports the following functions and services (ETSI 123 501- V15.5.0 -5G, 2019):

- Enables NFs to retrieve the UE's AMF registration information for 3GPP access,
- Exposes services to enable 5G and 4G interworking,
- Enables the NFs to retrieve the UE's SMF registration for 3GPP access,
- Lawful interception functionality,
- Subscription management,
- SMS management,

- Supports custom operations, such as when a consumer NF requests the AUSF to clear the Security Context[6], after the UE successfully re-authenticates in the same Serving Network, or has been successfully authenticated in another Serving Network, for example due to registration via another access type.

I. Unified Data Repository (UDR)

The Unified Data Repository is the repository to store the subscription data and policy data in the 5G core. NFs can store and retrieve structured data from the UDR. A UDR is deployed in each PLMN and can be accessed by UDM, PCF, NEF, which belong to the same PLMN.

I. UDR Service-Based Representation

Figure 4-28 represents the location of UDR in a 5GC service-based architecture. The UDR offers services through a Nudr service-based interface. UDR exposes its Nudr services to other NFs for storing in and retrieving data from it (3GPP-TS.29.504, 2022) (ETSI 123 501-V15.5.0 -5G, 2019).

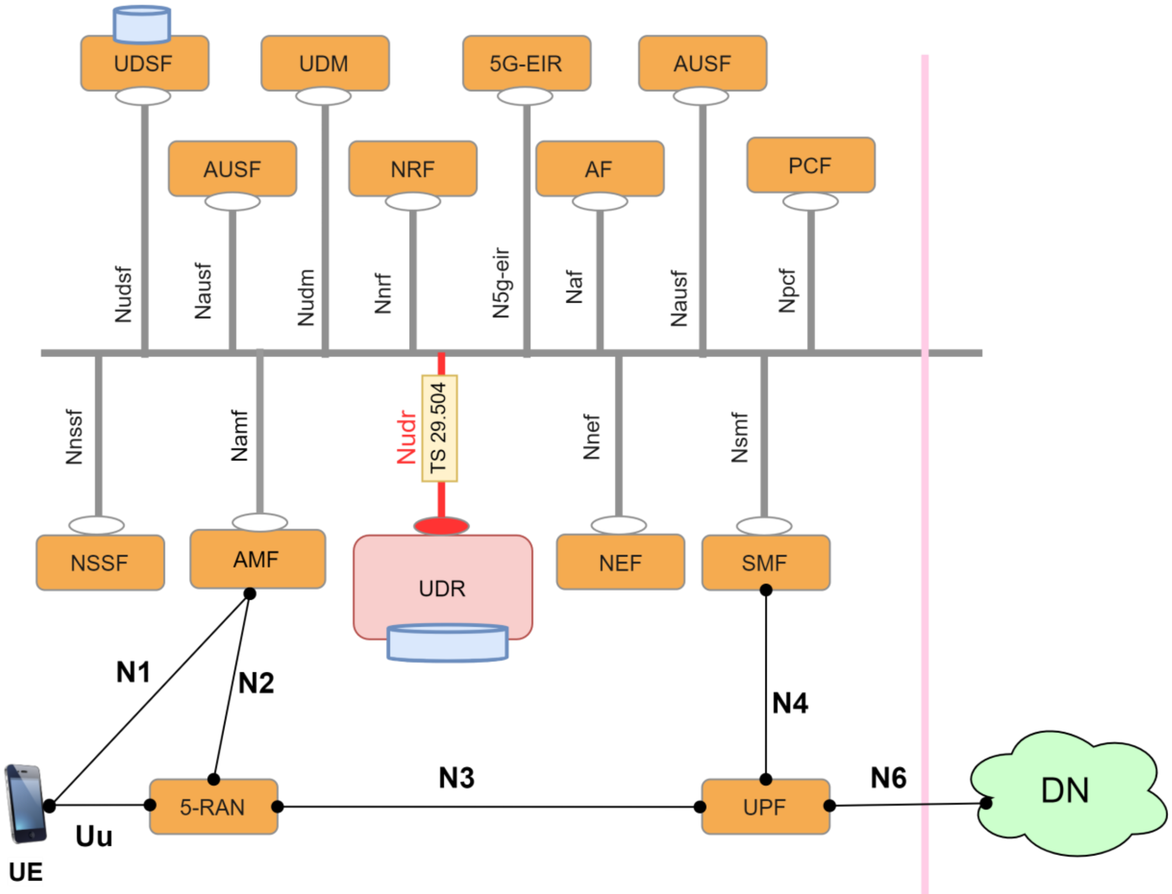


Figure 4-28: UDR Service-Based Architecture

II. UDR Reference Point Representation

Figure 4-29 represents the UDR reference point architecture. The UDR supports the following reference point for communication:

[6] security context is created in NAS procedure as the result of a primary authentication and key agreement procedure between the AMF and the UE

- **N35**: Reference point between UDM and UDR,
- **N36**: Reference point between PCF and UDR,
- **N37**: Reference point between NEF and UDR.

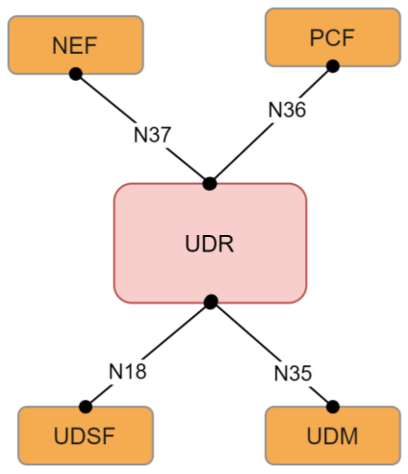


Figure 4-29: UDR Reference Point Architecture

III. UDR Key Functions and Features

The Unified Data Repository (UDR) supports the following functionalities (ETSI 123 501-V15.5.0 -5G, 2019):

- UDR manages the subscription data and provides storage for UDM to store and retrieve the data,
- UDR stores the policy data and enables PCF to store and retrieve the data,
- Storage and retrieval of structured data for exposure,
- UDR allows storage and retrieval of NF Group ID corresponding to a subscriber identifier.

J. User Plane Function (UPF)

The User Plane Function (UPF) is a fundamental component of the 3GPP 5GC architecture. As part of the evolution of the user plane and control plane separation architecture, the UPF is the decoupled user plane part while the SMF is the decoupled control plane. This advancement enables the data forwarding component (UPF) to be decentralized while the control plane (SMF) remains in the core network. The UPF can be deployed to process packets and aggregate traffic closer to the network edge or the location of the subscriber. Consequently, this approach increases bandwidth efficiency while reducing network complexity and at the same time reduces traffic latency to meet 5G requirements for various use cases, e.g., the URLLC use case requirements.

The UPF relies on the SMF to receive PDU session information to perform its functionalities. The UPF identifies user plane traffic flows based on information received from the SMF over N4 or based on a local policy.

I. UPF Service-Based Representation

Figure 4-30 represents the UPF service-based architecture. The UPF offers services through Nupf service-based interface. UPF exposes its Nupf services to other NFs (3GPP-TS.29.564, 2022):

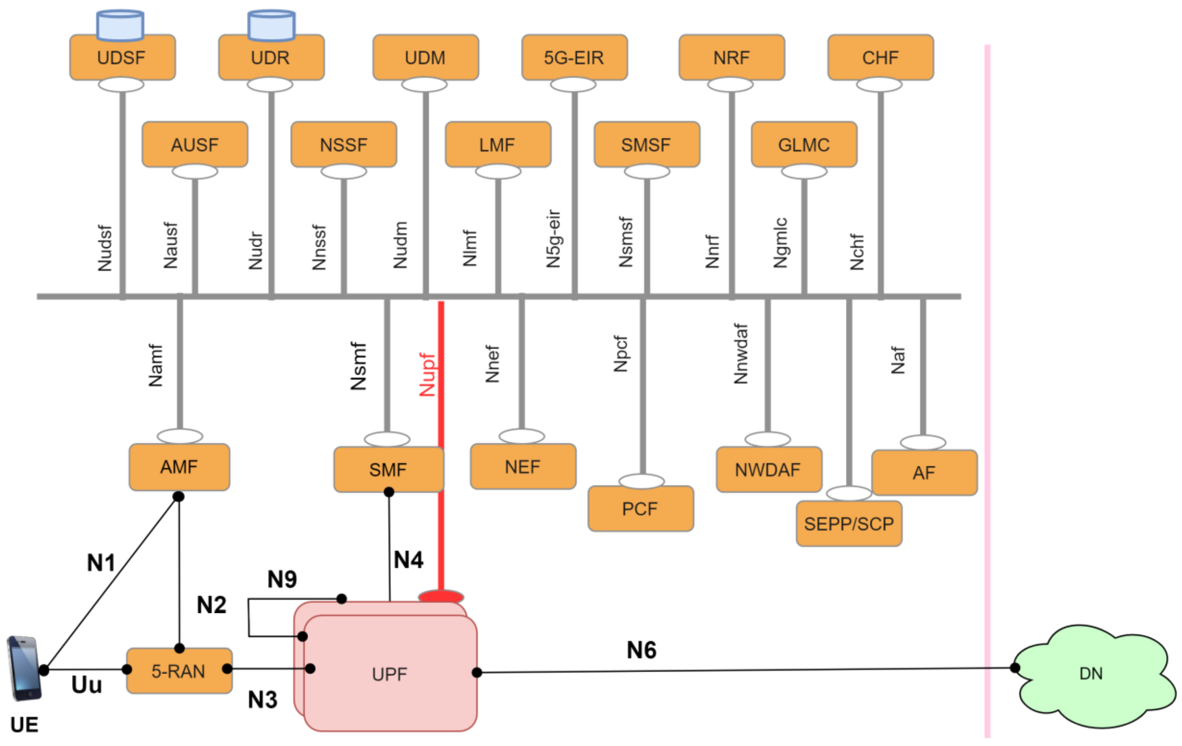


Figure 4-30: UPF Service-Based Architecture

II. UPF Reference Point Representation

Figure 4-31 represents the UPF reference point architecture. The UPF has four distinct reference points. Three are for the user plane and one is for the control plane (ETSI 123 501-V15.5.0 -5G, 2019):

- **N3**: Interface between the 5G-RAN and the (Initial) UPF. It uses GTP-U protocol to transport user plane traffic in the uplink and downlink direction,
- **N9**: Interface between two UPFs,
- **N6**: Interface between the UPF and the Data Network (DN). It is an IP-based interface,
- **N4**: Interface between the Session Management Function (SMF) and the UPF. It supports Packet Forwarding Control Protocol (PFCP) protocol.

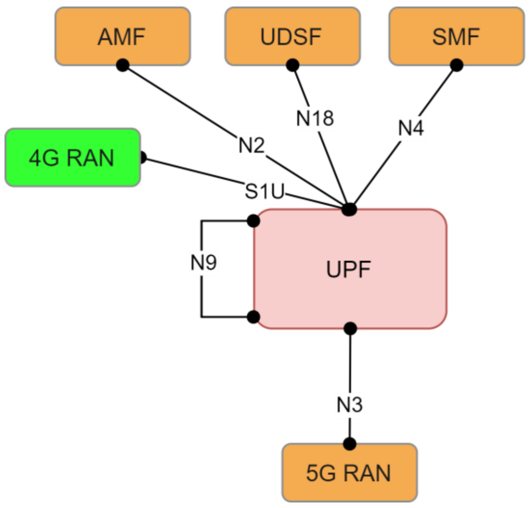


Figure 4-31: UPF Reference Point Architecture

III. UPF Key Functions

UPF supports the following functions and services (ETSI 123 501- V15.5.0 -5G, 2019):

- N4 session management, for example, N4 session establishment, modification & release,
- PDU session anchoring, branching point, and Uplink Classifier (UC),
- Support of IP type Protocol Data Unit (PDU) such as IPv4, IPv6, and IPv4v6,
- External PDU session point of interconnect to Data Networks (DNs),
- Anchor points for Intra/Inter Radio Access Technology (RAT) mobility,
- Sending and forwarding of one or more "end markers"[7] to the source NG-RAN nodes
- Packet flow processing, and uplink (UL) traffic verification,
- GTP-U path management,
- Explicit buffer management, downlink (DL) packet buffering and data notification triggering,
- Policy rule enforcement, for example, gating and routing,
- QoS management for the user plane, e.g., uplink (UL) / downlink (DL) rate enforcement, reflective QoS marking in DL,
- Transport level marking (Differentiated Services Code Point - DSCP),
- Local breakout,
- Multiple IP anchor for IPv6 multihoming,
- Lawful interception (user plane data collection).

IV. UPF Key Features

UPF supports the following features (ETSI 123 501- V15.5.0 -5G, 2019):

- UPF supports GTP-U path management messages over N3 and N9 interfaces,
- UPF supports GTP-U Tunnel management messages (Error Indication) over N3 and N9 interfaces,
- UPF supports path Maximum Transmission unit (MTU) discovery over the N3 interface,
- UPF supports IP fragmentation and reassembly over the N3 interface.

K. **Unstructured Data Storage Function** (**UDSF**)

UDSF is a component in the 5G network responsible for storing unstructured data. The 5G system supports stateless NFs where the computational resource is decoupled from the storage resource. When NFs are implemented to support stateless architecture, then NF can utilize the UDSF for storing/retrieving unstructured data into/from a storage. The unstructured data does not mean that data has no structure but rather it is pointing to the fact that 3GPP has not defined structure in specification and suppliers are allowed to define their own structure and then utilize UDSF to store the date into storage. The UDSF is deployed along with the other 5G NFs within the same PLMN and with the possibility having

[7] "end marker" packet is indicated in the GTP header and enables the eNB to forward the packet to towards the target eNB (Cisco, 2023a)

the common UDSF for all NFs to store/retrieve data or dedicated UDSF for particular NF depending on operator needs (ETSI.129.598, 2020).

I. UDSF Service-Based Representation

Figure 4-32 represents the 5G service-based architecture with UDSF highlighted as data store for any 5G NFs. The UDSF services are offered through Nudsf service-based interface. It exposes the services via Nudsf to other NFs.

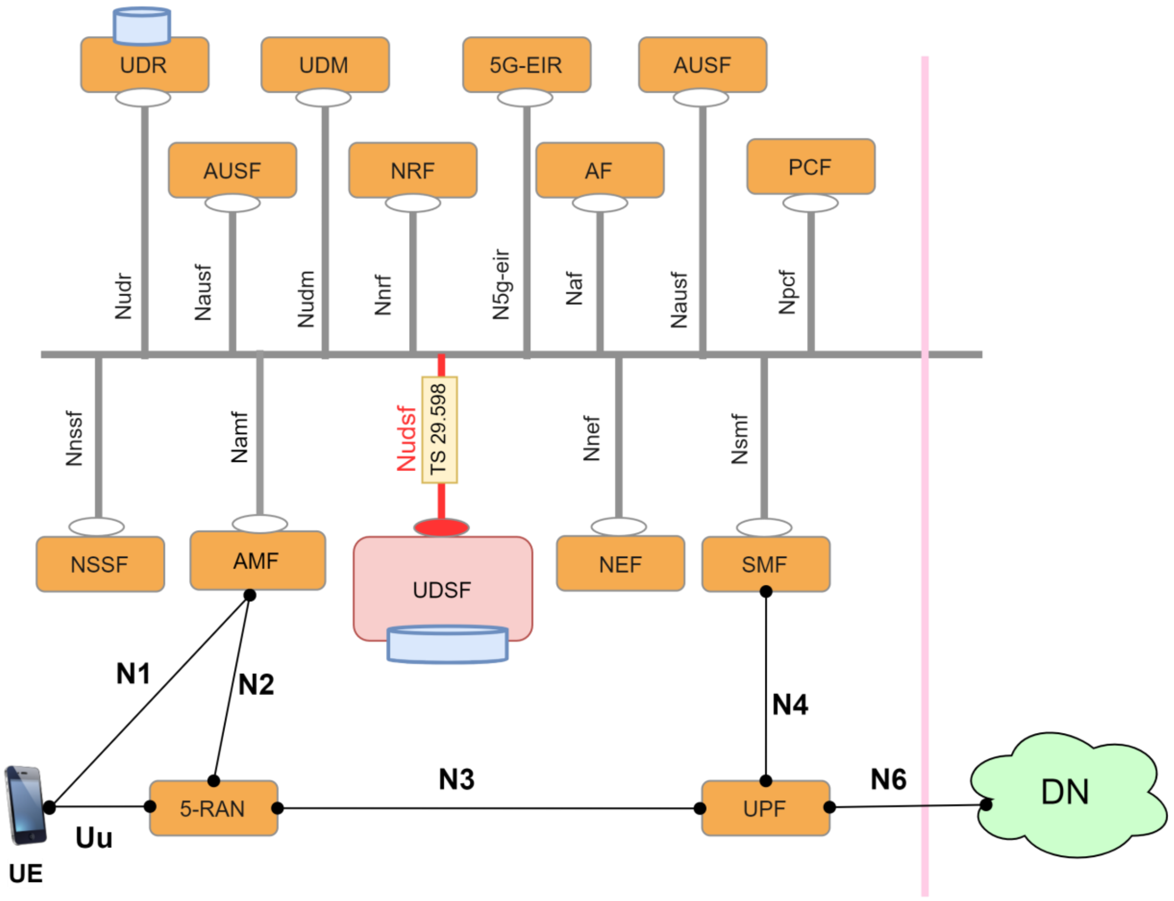


Figure 4-32: UDSF Service-Based Architecture

II. UDSF Reference Point Representation

Figure 4-33 represents the UDSF reference point architecture. The UDSF has single reference points known as N18. N18 is the interface for all the 5G network functions to store/retrieve data into/from UDSF. HTTP/2 protocol is used to implement N18 reference point. (ETSI.129.598, 2020)

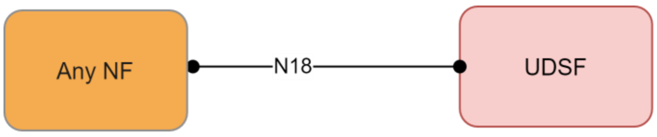


Figure 4-33: UDSF Reference Point Architecture

III. UDSF Key Functions

UDSF provides the functionality to any NF in the network to store/retrieve/delete/search unstructured data into storage via UDSF to make the NF stateless. HTTP/2-based API are exposed, and it allows storing data and retrieving/searching stored data primarily using a key-value style interface.

IV. UDSF Key Features

The UDSF is acting as an NF Service Producer to provide UDSF data repository service to the NF service consumer. Any NF can use the UDSF to store unstructured data. (ETSI.129.598, 2020)

L. Network Data Analytics Function (NWDAF)

The Network Data Analytics Function (NWDAF) is the analytics function for the 5G network. The NWDAF can collect data from all 5G network functions and OAM components. It provides statistical information on past events or predictive information based on the collected data. In summary, the performance management and fault management of the 5G network and OAM are governed by NWDAF. Including NWDAF in earlier 5G 3GPP specifications was fundamental, and the architecture enhancement to support NWDAF has been detailed in Release 16 and Release 17 of (ETSI.123.288, 2020). The enhanced architecture allows the NWDAF to extract and analyzes data from the network. The enhancements enable the NWDAF to improve the subscriber's quality of experience, reduce security risks and optimize network performance.

I. NWDAF Service-Based Representation

Figure 4-34 represents the 5G service-based architecture including NWDAF. The NWDAF services are offered to the other 5G network functions on Nwdaf interface. The 5G network functions can use the Nwdaf interface to request network analytics information from NWDAF. The NWDAF and 5G consumer network function belongs to the same PLMN. (ETSI.123.288, 2020)

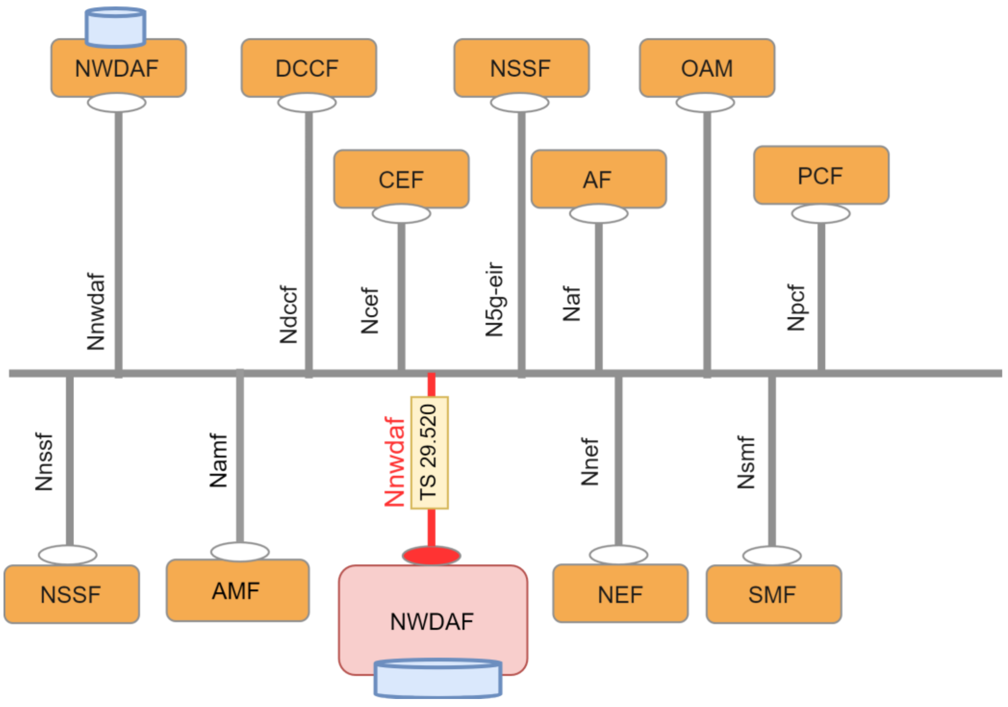


Figure 4-34: NWDAF Service-Based Architecture

Figure 4-35 shows the Nnf interface of NWDAF to collect data from all 5GC NFs. Nnf enables the NWDAF to request or cancel subscription to data delivery and a specific report of data for a particular context. (ETSI.123.288, 2020)

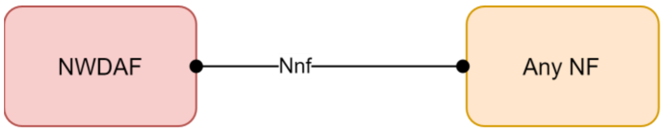


Figure 4-35: NWDAF Data Collection

II. NWDAF Reference Point Representation

Figure 4-36 represents the NWDAF reference point architecture. The NWDAF has reference points. The N23 and N24 are the main reference point mentioned in 3GPP (ETSI.123.501-R17, 2022) and descried below:

- **N23**: Interface between NWDAF and PCF, PCF uses N23 interface to get statistics or predictions on UE mobility from NWDAF.
- **N34**: Interface between NSSF and NWDAF, NSSF determines the load on the network slice instance retrieved from NWDAF using N34 interface.

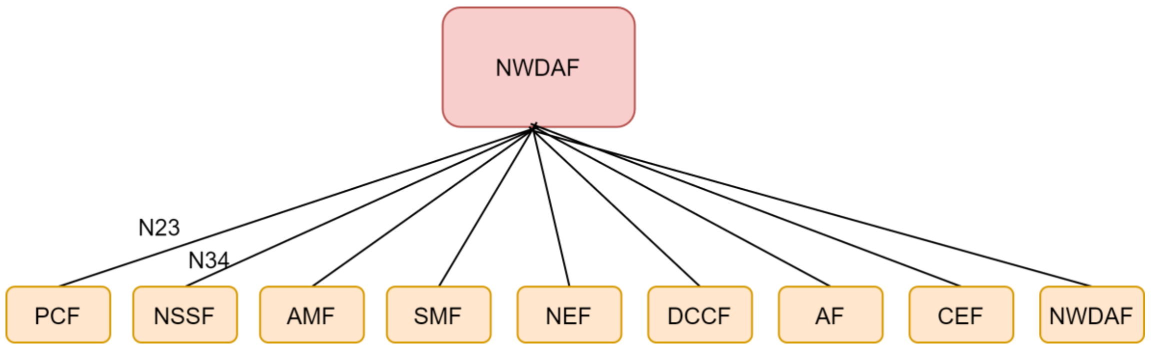


Figure 4-36: NWDAF Reference Point Architecture

III. NWDAF Key Functions and Features

NWDAF, a part of the 5GC architecture, acts as a central node that provides the statistical details of the past events or any predictive details. NWDAF supports the following functions and features (ETSI.123.501-R17, 2022):

- Support data collection from all 5G network functions based on subscription,
- Provides analytics information provisioning to network functions and application functions,
- Data collection from OAM based on subscription,
- Information retrieval from data repositories such as UDR/UDM,
- Information retrieval about NFs,
- Support Machine Learning (ML) model training and provisioning to NWDAFs.

NWDAF provides different analytic information to network functions based on the analysed events. NWDAF notifies the following event to NF consumers upon subscription:

- NF LOAD: The NF load analytics is used to request the load of a network function instance in the form of statistics, and prediction.

- Handling Network congestion in RAN: NWDAF supports to enable or disable the RAN congestion analytics functionality.

4.3.2 Next-Generation Radio Access Network (NG-RAN) Architecture

5G proposed the new architecture for the Radio Access Network (RAN) to accommodate the existing networks and support the implementation of the 5G use cases. Schematically, the 5G architecture looks like the LTE. Figure 4-37 shows the high-level 5G architecture and its components. It comprises of a User Equipment (UE), which includes a Mobile Station (MS) and a Universal Subscriber Identification Module (USIM)[8], Next Generation RAN and 5G Core Network (CN). NG-RAN provides data access to the 3GPP compliant devices via 5G CN, whereas devices connected to the public network can access the 5G core network via Non-3GPP Interworking Function (N3IWF). N3IWF is equivalent to the ePDG in the LTE network and provides the control plane connectivity to the AMF via N2 reference point, as well as user plane connectivity towards the UPF via N3 reference point. The 5G architecture provides complete separation of control plane and user plane traffic for 3GPP and non-GPP access. (ETSI.123.501-R17, 2022)

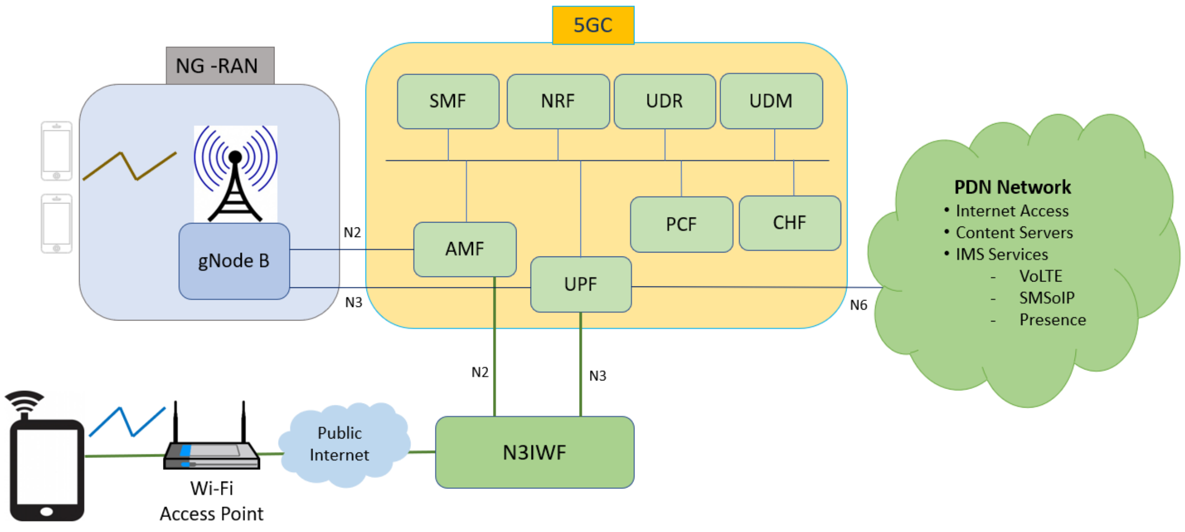


Figure 4-37: 5G Architecture

4.3.2.1 NG-RAN Architecture

The NG-RAN architecture is a pillar of the 5G system. It has been designed to operate in two modes to support various combinations of Dual Connectivity (DC) with LTE and different core networks (Lin and Lee, 2021):

- **Non-standalone (NSA):** In NSA mode, NG-RAN comprises of Next Generation Node-B (gNB) and Next Generation Evovled Node-B (ng-eNBs[9]). gNBs and ng-eNBs interoperate with one another to provide connectivity and they are connected to the same 5G core network, providing DC toward the same terminal.

[8] USIM is only the software part of the SIM, and the hardware part is now called Universal Integrated Circuit Card (UICC). USIM introduces many features advancement such as APN setting, MMS storage and so on.
[9] ng-eNB enables the 5G UE to connect to the 5G CN via 4G LTE air interface.

- **Standalone (SA):** In SA mode, a NG-RAN comprises of only gNBs, which connect directly to the 5G core network.

Figure 4-38 shows the NG-RAN logical architecture. A Next Generation (NG) reference point is introduced in the 5G network to provide the connectivity to the 5G core. NG-RAN includes gNB and gNB supports Frequency Division Duplex (FDD), Time Division Duplex (TDD) or dual mode operation. The interconnection between the gNB is achieved via Xn interface.

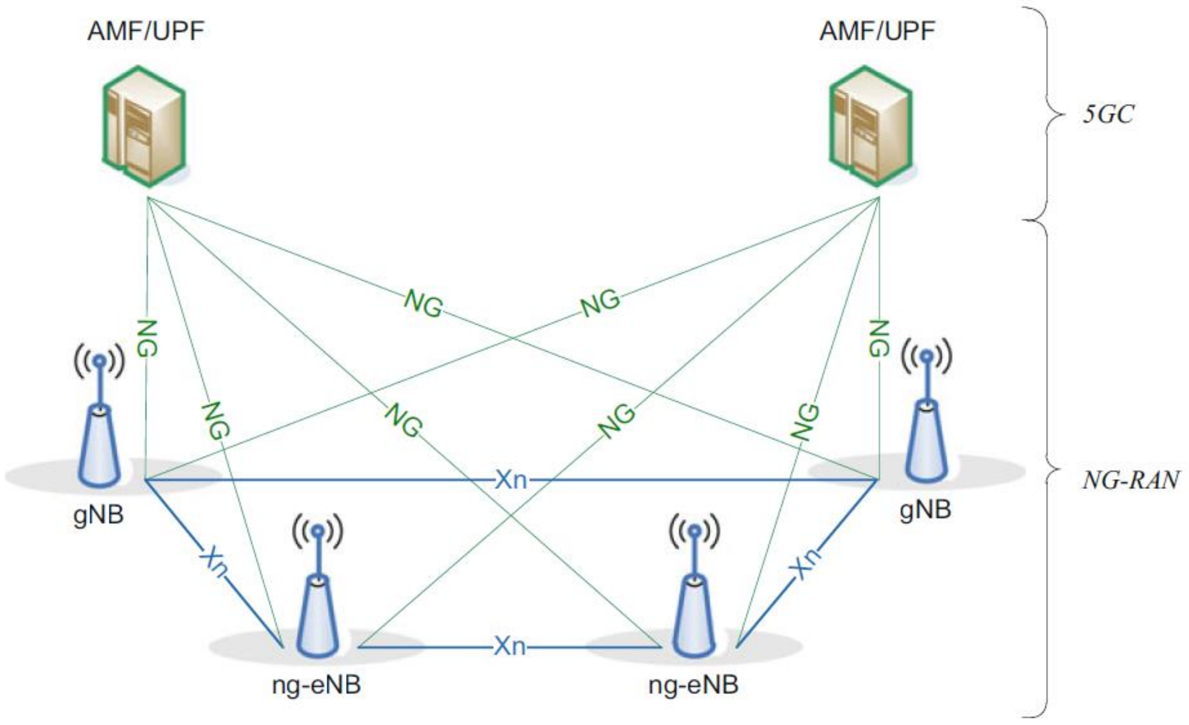


Figure 4-38: NG RAN Logical Architecture (ETSI.138.300, 2021) (Lin and Lee, 2021)

In Release-15, 3GPP added split functionality to gNB so that more flexibility can be achieved at the RAN level. gNB is split into three logical nodes (ETSI.138.401, 2018):

1. Central Unit (CU) or gNB-CU
2. Distributed Unit (DU) or gNB-DU
3. Radio Unit (RU)

Figure 4-39 depicts the high-level split architecture of gNB. It shows that multiple gNBs are interconnected through the Xn Interface, and the same gNBs are connected to the 5G CN through the NG interface. A gNB consists of a gNB-CU and one or more gNB-DU(s), and the interface between gNB-CU and gNB-DU is called F1. The F1 interface supports signaling exchange and data transmission between distributed and central units. It also separates the radio network layer and transport network layer and enables the exchange of UE-associated and non-UE-associated signaling. gNB-CU provides connectivity to the 5G CN via NG interface and connectivity to the gNB-CU via Xn interface. In 3GPP Release-15 (ETSI.138.401, 2018), one gNB-DU connects to only one gNB-CU, but it allows implementations to connect multiple gNB-CUs to a single gNB-DU for fault tolerance and resiliency. One or more cells can be supported by a single gNB-DU. The internal split of gNB is not visible to the 5G core network and other RAN nodes, it is seen in the network as a single gNB (ETSI.138.401, 2018).

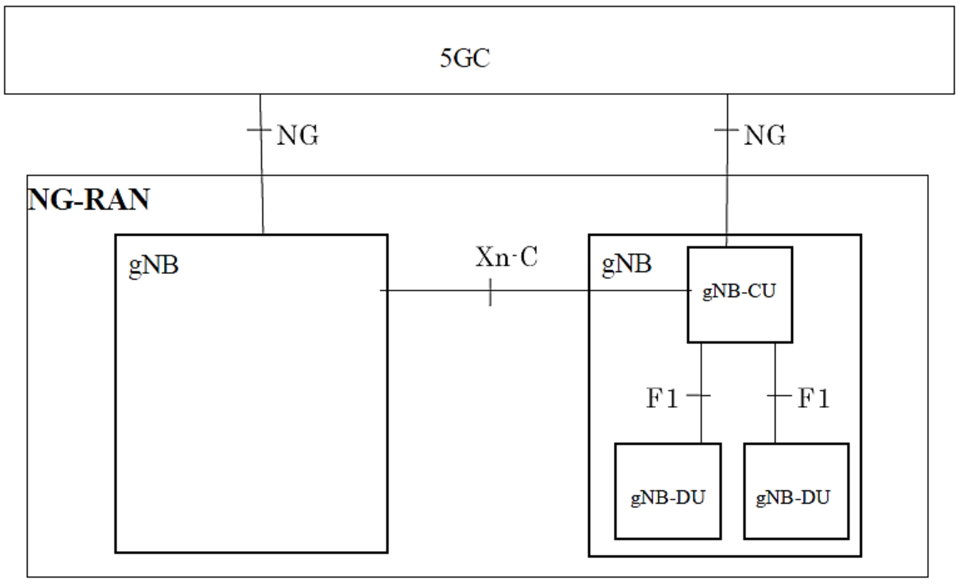


Figure 4-39: NG-RAN – gNB High Level Split Architecture (ETSI.138.401, 2018)

Figure 4-40 shows the details of the architecture split, as per the 3GPP specification TS 38.401 (ETSI.138.401, 2018). The CU is split further into control and user plane. The Central Unit Control Plane (CU-CP) is responsible for the control plane messages and the Central Unit User Plane (CU-UP) is responsible for the user plane messages. A gNB may consist of a single CUCP and multiple CUUPs and DUs. CUUPs and DUs can be connected to a single CUCP. The control interface between the CUCP and DU is called F1-C and user plane interface between CUUP and DU is called F1U. CUUP is connected to the CUCP on E1 interface.

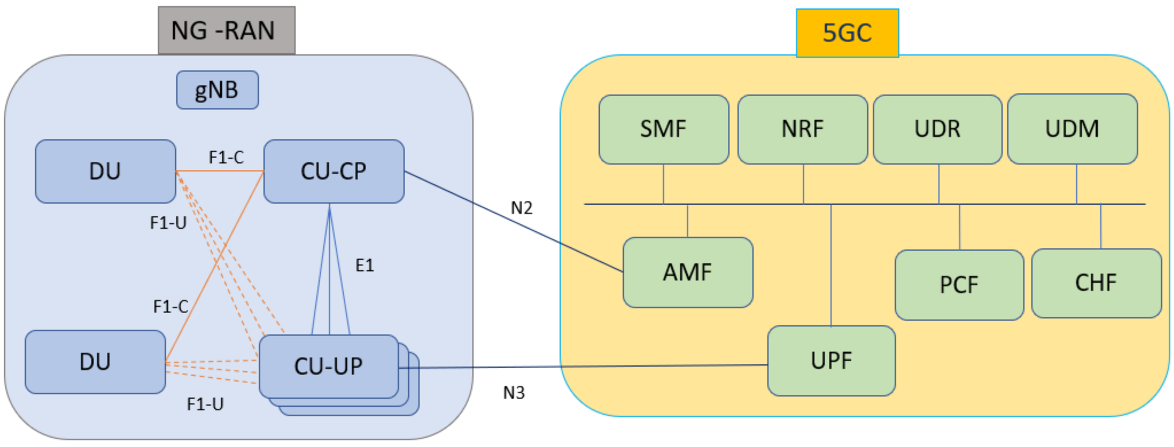


Figure 4-40: NG-RAN – Detailed Split Architecture

A. NG-RAN Interfaces

The NG-RAN has four main interfaces as shown in Figure 4-41. F1 interface is between the CU and DU components. E1 interface is between the CUCP and CUUP, whereas NG interface is towards the 5G CN from CU component of the gNB. Multiple gNBs can communicate with each other on the Xn interface.

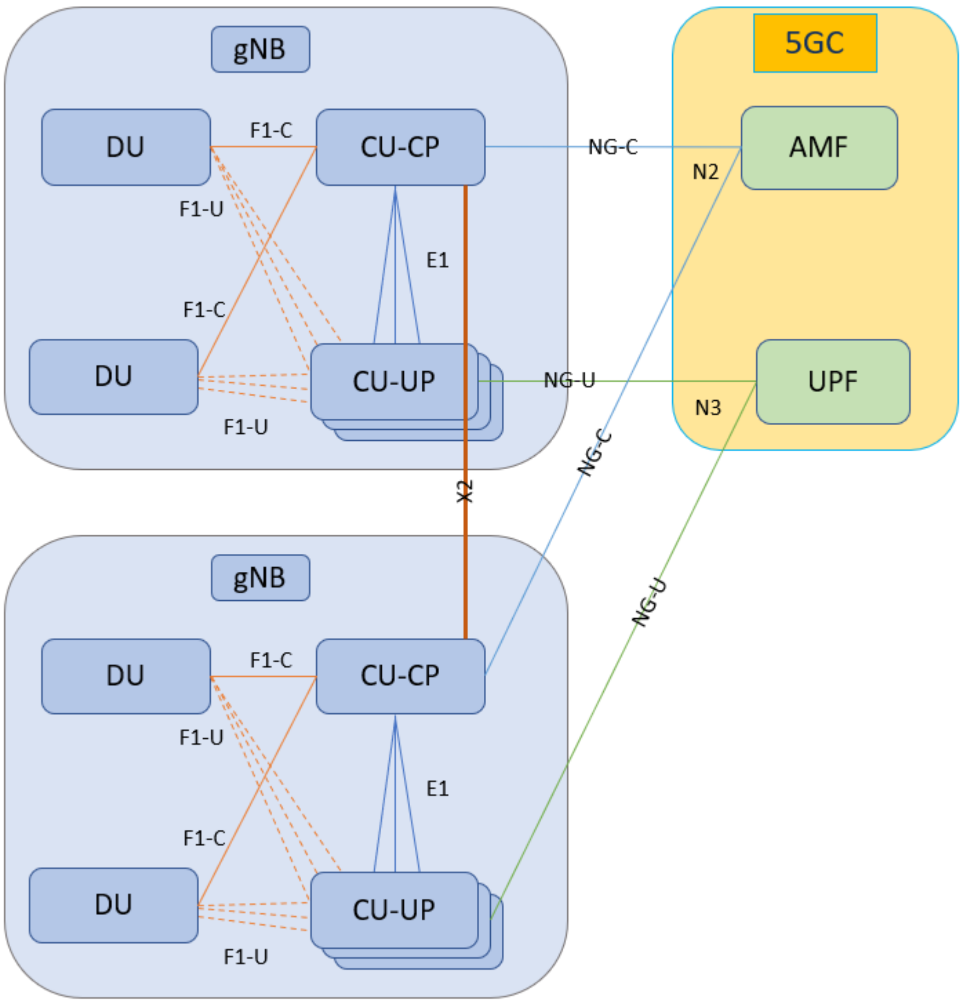


Figure 4-41: NG-RAN Interfaces

I. F1 Interface

F1 is an interface between the CU and DU. It has been divided into two interfaces based on control plane and user plane traffic. F1-C Interface is responsible for control plane connectivity between the DU and CUCP and it provides the following functions:

- **Management Functions:** It performs the F1 setup towards the CUCP, gNB-CU Configuration Update, gNB-DU Configuration Update and implementation of error indication and reset function (ETSI.138.470, 2020).
- **System Information Management Functions:** The gNB-DU is responsible for the scheduling, broadcasting of system information and the encoding of NR-MIB and SIB1, while the encoding of other SI messages is performed by the gNB-CU (ETSI.138.470, 2020).
- **UE Context Management Functions:** The F1 UE context modification request can be initiated by either the gNB-CU or the gNB-DU. It is responsible for the establishment and modification of the necessary UE context. When the establishment of the F1 UE context is initiated by the gNB-CU, the gNB-DU can accept or reject the establishment based on admission control criteria (e.g., depending on the resource availability the gNB-DU can reject a context setup or modification request). It is

responsibility of F1 UE context management function to establish, modify and release Data Radio Bearers (DRBs) and Signaling Radio Bearers (SRBs) (Bertenyi *et al.*, 2018).

- **RRC Message Transfer Function:** This function can be used by the gNB-CU and the gNB-DU for transferring of RRC in both directions (ETSI.138.470, 2020).

There are also other functions provided by the F1-C such as paging, warning messages information transfer, Remote Interference Management (RIM) message transfer, trace, load management and self-optimization support (ETSI.138.470, 2020).

The F1-U Interface is used for user plane connectivity between the DU and CUUP. The F1U interface provide two functionalities as listed below:

- **Transfer of User Data:** This function transfer user data between gNB-CU and gNB-DU (ETSI.138.470, 2020).
- **Flow Control Function:** It allows for controlling the downlink user data transmission towards the gNB-DU. In order to have improved performance on data transmission, several functionalities are introduced such as fast retransmission of Packet Data Convergence Protocol (PDCP) PDUs lost due to radio link outage, discarding redundant PDUs, the retransmitted data indication, and the status report (Bertenyi *et al.*, 2018).

II. E1 Interface

The E1 interface enables communication between the control and user plane of CU such as the CUCP and the CUUP. E1 is an open interface, and it allows the exchange of signaling information between the CUCP and the CUUP (user plane information is not being forward on E1 interface). This interface separates the radio network layer and transport network layer and enables exchange of UE associated information and non-UE associated information (ETSI.138.460, 2020).

The E1 interface provides the following three main functionalities:

- **Management function:** This function allows the E1 setup between the CUCP and the CUCP, implementation of error indication and reset function and the CUUP configuration update (ETSI.138.460, 2020).
- **Bearer Context management:** This function is used to setup and modify the QoS-flow to Data Radio Bearer (DRB) mapping configuration. It is used by the CUCP to send security information to the CUUP and if required, to send the parameters for header compression. The function is used by the CUUP to notify the CUCP about the DL data arrival detection to trigger the paging procedure over F1 or Xn. It is also used by the CUUP to notify the event of user inactivity to the CUCP and report data volume to the CUCP (ETSI.138.460, 2020).
- **Trace function:** Trace function provides means to control trace sessions for a UE over E1 interface (ETSI.138.460, 2020).
- **Load management function:** This function enables CUCP to request the load report to the DU and then, it is used by the CUUP to report the results of measurements admitted by the CUUP (ETSI.138.460, 2020).

III. Xn Interface

Xn is the open interface between the two NG-RAN nodes (gNB), and it allows the exchange of signaling information between NG-RAN nodes. The interface enables interconnection of NG-RAN nodes supplied by different vendors. Xn interface supports intra NG-RAN mobility and dual connectivity between the NG-RAN nodes. Xn interface has been further divided into control plane (Xn-C) and user plane (Xn-U) interface between NG-RAN nodes (ETSI.138.420, 2020).

Xn-C performs the signaling association management between NG-RAN nodes and provides the following principal functions:

- **Management function:** The initial setup of an Xn interface between the two NG-RAN nodes is performed over Xn-C interface. The functionalities such as error handling, Xn reset, configuration data update and removal are provided via Xn-C management functions (ETSI.138.420, 2020).
- **UE Mobility Management function:** The function of Xn-C interface enables the mobility of UEs between two NG-RANs by initiating the handover process. Hence, handover preparation, cancellation, success, RAN paging, data forwarding control function and retrieve UE context function are provide by UE mobility management function of Xn-C interface (ETSI.138.420, 2020).
- **Dual Connectivity function:** The additional resource in secondary node in the NG-RAN can be used by utilizing dual connectivity function (ETSI.138.420, 2020).
- **Energy Saving function:** This function uses the Xn interface to activate and deactivate the cell to decrease the energy consumption (ETSI.138.420, 2020).

The other functions provided by the Xn-C are trace function, load management and volume reporting.

Xn-U interface allows user plane traffic exchange between two NG-RAN nodes and provides the following functionalities:

- **Data Transfer:** This function allows the data transfer between two NG-RAN nodes to enable the dual connectivity and mobility (ETSI.138.420, 2020).
- **Flow Control:** This function enables the NG-RAN node to receive the user plane data from another NG-RAN node to provide feedback information associated with the data flow (ETSI.138.420, 2020).
- **Assistance information:** This function is allows the NG-RAN node to share the assistance information with other NG-RAN nodes, such as radio conditions (ETSI.138.420, 2020).

IV. NG Interface

The NG interface is used to connect gNB and ng-eNB with 5G core network. It is an open logical interface and provides separation for control plane as NG-C (also known as N2 reference point) and user plane as NG-U (also known as N3 reference point). NG interface supports procedures to establish, maintain and release NG-RAN parts of PDU sessions; perform intra-RAT handover and inter-RAT handover; the transfer of NAS signaling messages between UE-AMF and separation of each UE on the protocol level for user specific signaling management (ETSI-TS.138.410, 2020).

The NG-C interface resides between the CUCP and AMF over the N2 reference point, and the NG-U interface establishes communication between the CUUP and UPF over N3 reference point. There are functionalities provided by NG interface related to control-plane and user-plane. The main functions are listed below:

- **Paging function:** Sending paging requests to the NG-RAN nodes involved in the paging area is achieved via paging function (ETSI-TS.138.410, 2020).
- **Mobility Management function:** The mobility function implements the handover function and it includes handover preparation, execution and is completed via the NG interface (ETSI-TS.138.410, 2020).
- **PDU Session Management function:** This function is responsible for establishing, modifying and releasing of relevant PDU sessions (ETSI-TS.138.410, 2020).
- **UE Context Management function:** This function enables the AMF to establish, modify or release a UE context in the AMF and the NG-RAN (ETSI-TS.138.410, 2020).
- **NAS Node Selection function:** The selection of AMF is performed by the NG-RAN using NAS node selection. The function allows the selection of AMF, based on the UE's temporary identifier or slicing information (ETSI-TS.138.410, 2020).
- **NAS Transport function:** The transport or re-route of NAS messages for a specific UE is achieved via Non-Access Stratum (NAS) transport function using NG interface (ETSI-TS.138.410, 2020).
- **NG Interface Management function:** The NG interface management related functions are implemented via this function such as error indication, start or reset (ETSI-TS.138.410, 2020).
- **AMF Management function:** This function supports the AMF planned removal and auto-recovery (ETSI-TS.138.410, 2020).

4.3.2.2 NG-RAN Protocol Stack & Functional split

As mentioned in the previous sub-section, the control plane and user plane traffic in the NG-RAN have been separated to achieve better performance and more flexibility in the RAN for future expansion. Based on this concept, the NR radio protocol stack is segregated into the control plane and user plane stack, as shown in Figure 4-42. All the signaling messages are processed through the control plane stack, and user data is processed by the user plane stack. The control plane and user plane stacks are like the Packet Data Convergence Protocol (PDCP). In the user plane, the Service Data Adaptation Protocol (SDAP) layer at the top of the radio stack communicates with UPF. Whereas, in the control plane, Radio

Resource Control (RRC) and NAS are added on top of the radio stack to establish communication with AMF.

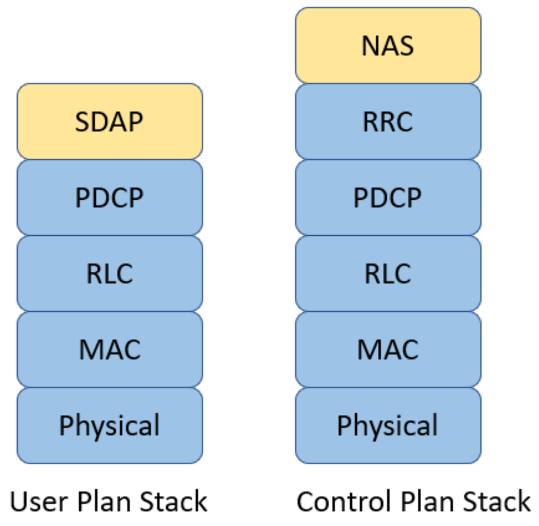


Figure 4-42: User Plan and Control Plan Protocol Stack

The functionalities provided by each protocol in the user plane and control plane protocol stacks are briefly explained below:

- **Service Data Adaptation Protocol (SDAP):** The SDAP protocol is added to the user plane protocol stack of the NG-RAN compared to LTE. SDAP has introduced a new flow-based QoS model of the 5G core network to allow the configuration of different QoS requirements for different IP flows of a PDU session in the core network. The SDAP layer provides mapping of QoS flows to radio bearers and marking QoS flow identifiers (QFI) in both DL and UL packets.
- **Non-Access Stratum (NAS):** The Non-Access Stratum (NAS) protocol enables communication between the UE and the AMF of the 5G core network. It is used for core network related functions such as registration, authentication, location updating and session management (Bertenyi *et al.*, 2018).
- **Radio Resource Control (RRC):** The RRC protocol is used between UE and the 5G-RAN and it is used for control and configuration of the radio related functions in the UE (Bertenyi *et al.*, 2018).
- **Packet Data Convergence Protocol (PDCP):** The PDCP provides functionalities and services to both the user plane and control plane. The main functions and services include data transfer, header compression and decompression through the use of ROHC (Robust Header Compression), security functions including ciphering / deciphering and integrity protection, duplication of transmitted PDCP PDUs, as well as reordering, duplicate detection of received PDCP PDUs and the introduction of integrity protection for user plane data (Bertenyi *et al.*, 2018).
- **Radio Link Control (RLC):** The RLC provides the Layer-2 functionalities such as error correction, reordering of 5G-RLC data PDUs, duplicate dedication, protocol error

handling, segmentation, re-segmentation and 5G-RLC re-establishment (ETSI.138.300, 2021).

- **Media Access Control (MAC):** The MAC layer operates at :ayer-2 to provide functionality such as beam management, random access procedure, mapping between logical and transport channels, scheduling information reporting, error detection, priority handing, transport format selection, padding and concatenation of multiple MACs (ETSI.138.300, 2021).
- **Physical Layer (PHY):** Provides error detection, encoding and decoding, rate matching, mapping of the coded transport channel onto physical channels, power weighting, modulation, frequency and time synchronization, frequency and time synchronization, radio characteristics measurements, digital and analog beamforming, Radio Frequency (RF) and Multiple Input Multiple Output (MIMO) antenna processing.

Based on the protocol stack, multiple functional split variants are possible as shown in Figure 4-43. Option-1 considers the RRC to be part of the CU and the rest of the protocols are implemented on the DU. Option-2 considers RRC & PDCP to be part of a CU and the rest should be included in the DU. Option-3 introduces intra RLC split in the low and high layers between the CU and DU, respectively. Option-4 splits the RLC and MAC between the CU and DU. Option-5 adds the intra MAC split in low and high between CU and DU, respectively. Option-6 splits the MAC and PHY between the CU and DU. Option-7 introduces intra-PHY split into low and high layers between CU and DU. Finally, option-8 consider the RF functionality on the DU and all the other protocols to be part of the CU (3GPP-TS.38.801, 2017).

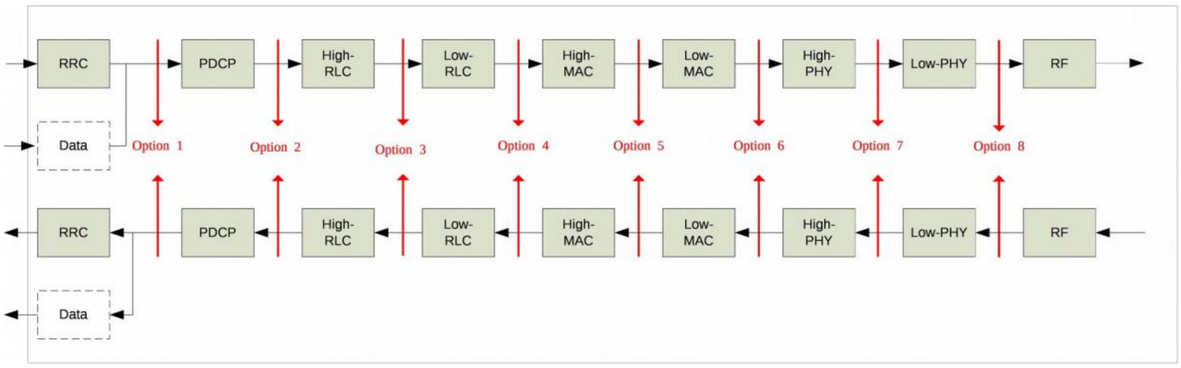


Figure 4-43: 3GPP Functional Split Option for CU and DU (3GPP-TS.38.801, 2017)

3GPP had extensive discussion on the split architecture so that the best split can be recommended to the industry. Finally, high level CU and DU split with option-2 was recommend by 3GPP addition to traditional monolithic RANs (3GPP, 2023).

5 ESTABLISHING A FUNCTIONAL 5G NETWORK

5GC network simulator is primarily prerequisite to build, implement and evaluate the framework to identify the monitoring points for the 5G network functions and data collection solution for associated metrics. 5GC simulator deployed at Oslo Metropolitan University cloud provides continuous access to facility and network to perform the research work with flexibility and more control. There are multiples open source 5GC simulators and projects available over the internet, few are listed below:

Simu5G: The Simu5G is an open-source network simulator and is developed in collaboration between Intel and the Department of Information Engineering at the University of Pisa (SIMU5G, 2022).

Free5GC: The free5GC is an open-source project to implement the 5GC network in Communication Service/Software Laboratory (CS Lab) managed by National Chiao Tung University (NCTU) (FREE5GC, 2022).

OpenAirInterface5G: The OpenAirInterface5G is a project developed by EURECOM, a French graduate school and a research center in communication systems based in the international science park of Sophia Antipolis within the new Campus Sophia Tech. This collaboration brings European renowned universities to work together on various research topics. Additionally, EURECOM is bringing big industry players such as BMW Group Research & Technology, SAP, ST Microelectronics, Orange, Symantec, Monaco Telecom to work together on the research and lab experiments. The strong administrative structure of the EURECOM has been beneficial to achieve the extensive relation between the academic's research and the market industry (EURECOM, 2022a).

OpenAirInterface5G having dedicated support by EURECOM and its wide usage in previously research subjects at the Oslo Metropolitan University, makes it prominent project to be utilized to build the 5G network at the Oslo Metropolitan University cloud for the experimentation phase of this thesis. This chapter covers the detailed implementation of a complete 5G environment creation via automation within OpenStack cloud powered by Open Air Interface (OAI) 5G.

5.1 Cloud Overview

The cloud environment at Oslo Metropolitan University was created to facilitate research work and enable building experimental testbeds in a real cloud environment. The OpenStack cloud infrastructure comprises of multiple controllers, compute, and storage nodes. The environment created for this thesis has been built in a separate cloud tenant with a quota dedicated to the project as follows:

Table 5-1: OpenStack Project Resources

Resources	Allowed Quota
Number of Instances	16
VCPUs	64
RAM (GB)	80

Number of Floating IPs	1
Number of Security Groups	10
Number of Volumes	6

5.2 Automation

The quota assigned to the project within the cloud environment is utilized to create multiple resources to onboard the 5G network in isolation settings. The process starts with manual deployment of a bastion virtual machine with the relevant tools required to automate the creation of infrastructure and container layer to deploy the OpenAirInterface5G. The automation process has been developed to speed up the infrastructure and container layer orchestration in case of rebuilding the network. The bastion virtual machine is deployed manually with "`Ubuntu-20.04-LTS`" as source image and assigned a floating IP to be accessible from the outside. The bastion virtual machine utilizes ana OpenStack flavor with 4 vCPUs, 8 GB RAM and 80 GB storage disk, it is protected with a firewall and access to the machine is only allowed from individuals having public keys installed on the bastion. By default, only a single key has been installed within the operating system named as "`Newkey`", described in the output below.

```
root@baston:~# openstack server show baston
+-----------------------------+----------------------------------------------------------+
| Field                       | Value                                                    |
+-----------------------------+----------------------------------------------------------+
| OS-DCF:diskConfig           | AUTO                                                     |
| OS-EXT-AZ:availability_zone | nova                                                     |
| OS-EXT-STS:power_state      | Running                                                  |
| OS-EXT-STS:task_state       | None                                                     |
| OS-EXT-STS:vm_state         | active                                                   |
| OS-SRV-USG:launched_at      | 2023-04-02T16:24:46.000000                               |
| OS-SRV-USG:terminated_at    | None                                                     |
| accessIPv4                  |                                                          |
| accessIPv6                  |                                                          |
| addresses                   | netsys_net=10.0.71.13, 128.39.121.194                    |
| config_drive                | True                                                     |
| created                     | 2023-04-02T16:24:18Z                                     |
| flavor                      | m1.large (4)                                             |
| hostId                      | 38d01b1779be3a2c07f2c13b7763167ee6680a79b7d70e253df13a14 |
| id                          | 1d4eb1df-6bb4-4170-9ea6-543288206ed7                     |
| image                       | Ubuntu-20.04-LTS (af6ebfa7-427c-40d0-9ca2-ad02a25a52c9)  |
| key_name                    | NewKey                                                   |
| name                        | baston                                                   |
| progress                    | 0                                                        |
| project_id                  | 85a74abdb9bb45b69b8080d64bf3348e                         |
| properties                  |                                                          |
| security_groups             | name='default'                                           |
| status                      | ACTIVE                                                   |
| updated                     | 2023-04-02T16:24:46Z                                     |
| user_id                     | d6bf86c10450434fb88d4dd3f8bf245d                         |
| volumes_attached            |                                                          |
+-----------------------------+----------------------------------------------------------+
```

5.2.1 Puppet

Puppet is an enterprise grade automation and configuration management open-source tool that manages routine tasks such as installation, patching, upgrading, configuration, updates or enforcing compliancy on the systems in an operational environment. Puppet supports declarative states and procedural capabilities, thus allowing support for scripts written in any language. It will make sure that the state of the system is kept as desired by applying the pre-configured tasks on the all the puppet agents (PUPPET, 2022) (PUPPET, 2023b). The Puppet architecture is based on the server and client model. A bastion virtual machine in the existing setup will function as a Puppet server and the rest of the nodes will be deployed as clients/agents.

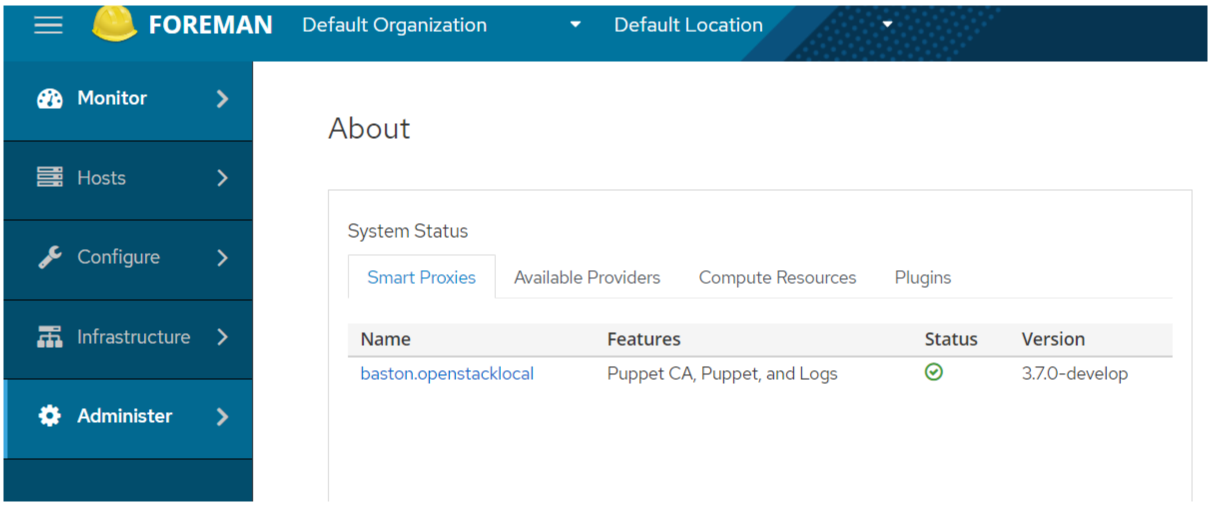
A Puppet server is deployed on the baston virtual machine, and this will be used to create the Kubernetes cluster described in the later sections of this chapter.

```
root@baston:~# puppetserver -v
puppetserver version: 6.20.0
root@baston:~#
```

5.2.2 Foreman

Foreman can be used in combination with puppet to enhance the automation process and to manage and deploy multiple hosts. It is a complete lifecycle management tool for physical and virtual servers. Foreman enables system administrators to easily automate repetitive tasks, quickly deploy applications, and proactively manage servers, on-premises or in the cloud. It provides a feature-rich graphical user interface (GUI) to perform different actions such as bootstrapping of physical server and virtual machines in any environment such as Amazon EC2, Google Compute Engine, OpenStack, and many other providers, as well as deliver monitoring of the registered host in via a dashboard along with the customized alerts created based on the requirement (FOREMAN, 2022).

Foreman version 3.7.0 has been deployed on the bastion virtual machine and the GUI is accessible on the floating IP assigned to the baston VM. Foreman also provides the Hammer command line interface (CLI) to manage the provision host.



Name	Features	Status	Version
baston.openstacklocal	Puppet CA, Puppet, and Logs	✓	3.7.0-develop

Figure 5-1: Foreman GUI

5.2.3 Docker

Docker is an open-source OS-level virtualization layer, it enables the code and application to run inside a container in isolated environment on the same operating system and the software that hosts the container is called docker engine. The bastion virtual machine requires container environment to accomplish the bootstrapping of the Kubernetes cluster via Puppet and Foreman, hence docker version 23.0.2 has been deployed on the bastion virtual machine.

```
root@baston:~# docker -v
Docker version 23.0.2, build 569dd73
root@baston:~#
```

5.2.4 Infrastructure Automation via MLN (Manage Large Networks)

MLN (Manage Large Networks) is an experimental tool to administrate the virtual machine building and run virtual machines and networks based on Amazon, OpenStack, VMware Server, and User-Mode Linux environments. The tool provides the automation to create virtual networks with multiple virtual machines and enables the user to control the bootstrap process of virtual machines to install the pre-defined packages, networks, security groups, storage, and many other (Begnum, 2009).

MLN version 1.0.8.8 has been deployed on the bastion virtual machine to enable the automation process of the infrastructure used to deploy the 5G network on the virtual machines.

```
root@baston:~# mln write_config
---> Config for root
mln version 1.0.8.8
Template directory is: /opt/mln/templates
Files directory is:    /opt/mln/files/root
Projects directory is: /opt/mln/projects/root
UML directory is:
User-Mode Linux kernel: /linux ()
Kernel modules are located: /
---> Other default variables:
BOOT_ORDER 99
CPUHOG 0.3
FILESYSTEM_SIZE 250M
FAMILY debian
LOCK_TIMEOUT 60
TERM_COMMAND xterm -bg black -fn fixed
SLIRP
TERM screen
SLIRP_IF_ADDR 10.0.2.15
SLIRP_DNS 10.0.2.3
COLOR lightgrey
SERVICE_HOST
MEMORY 64M
SCREEN_COMMAND screen -d -m -S
TEMPLATE Debian-4.0.ext3
MLN_VG mln-images
MODULE_PATH /
XEN_BRIDGE xenbr0
DAEMON_SOCKET 34001
PLUGIN_LOCATIONS ARRAY(0x557545ed8930)
MAC_BASE fe:fd:0:0:
---> Plugins:
vncplugin version 1.0
PUPPET:  Plugin version 0.1
OpenStack plugin version 0.9
Postfix plugin version 0.5
iSCSI backend plugin version 1.2
Apache plugin version 0.8
hostsFile version 1
winConfig plugin version EXPERIMENTAL
AMAZON EC2 plugin version 1.2
Fedora (users and groups) support plugin for MLN, version 1autoenum version 0.5
VMware Server plugin version 0.3
Xen extra plugin version 0.1
KVM plugin version 0.1
root@baston:~#
```

5.2.5 Bootstrapping

The bastion virtual machine has MLN installed, as well as Puppet and Foreman to initiate the bootstrapping of the required infrastructure for the deployment of a 5G core network. The

OpenAirInterface5G core network supports deployment in a container and Kubernetes environment, therefore the bootstrapping process comprises of the following steps:

a. Setup Puppet Modules,
b. Setup Kubernetes configuration,
c. Setup MLN configuration for infrastructure creation,
d. Kubernetes cluster creation via MLN Puppet and Foreman.

5.2.5.1 Setup Puppet Modules

The first step of the bootstrapping process is to make the Puppet-master and foreman machine ready on the baston virtual machine to implement the desired task to achieve end to end automation for the environment creation. Puppet has a Kubernetes module to install and configure the Kubernetes cluster automatically, thus allowing the deployment of a Kubernetes cluster to be managed via the open-source Puppet. Puppet also maintains the state of the cluster automatically and uses the “kubeadm” toolkit to bootstrap the Kubernetes cluster (PUPPET, 2023a).

The “puppetlabs-kubernetes” module version 6.2.0 is deployed on a bastion virtual machine for this purpose. The module was intended to work seamlessly, but it was shown during the boot strapping process that puppet modules may have multiple issues and several fixes were implemented to enable correct operation.

```
root@baston:~# puppet module list
/etc/puppetlabs/code/environments/production/modules
├── camptocamp-kmod (v2.5.0)
├── garethr-docker (v4.0.0)
├── herculesteam-augeasproviders_core (v2.7.0)
├── herculesteam-augeasproviders_sysctl (v2.6.2)
├── puppet-archive (v4.6.0)
├── puppetlabs-apt (v8.5.0)
├── puppetlabs-kubernetes (v6.2.0)
├── puppetlabs-stdlib (v6.6.0)
└── stahnma-epel (v2.0.0)
```

The list of fixes applied on the module to facilitate an end-to-end operation are list below:

A. Kubernetes key fix: The “puppetlabs-kubernetes” module has a wrong fingerprint key for the repository inside the manifest file with associated Kubernetes distribution. The Kubernetes fingerprint has been updated in the “repo.pp” with the correct value to enable the installation of the Kubernetes cluster:

```
      'Debian': {
        $codename = fact('os.distro.codename')
        apt::source { 'kubernetes':
          location => pick($kubernetes_apt_location,'https://apt.kubernetes.io'),
          repos    => pick($kubernetes_apt_repos,'main'),
          release  => pick($kubernetes_apt_release,'kubernetes-xenial'),
          key      => {
            'id'     =>
pick($kubernetes_key_id,'A362B822F6DEDC652817EA46B53DC80D13EDEF05'),
            'source' =>
pick($kubernetes_key_source,'https://packages.cloud.google.com/apt/doc/apt-key.gpg'),
          },
        }
```

B. The Docker repository and the key solution: The “puppetlabs-kubernetes” module has multiple issues to install Docker on the worker and master nodes. This issue is related to the repository and fingerprint key in the configuration files of the Puppet module, the default installation of the module contains a wrong repository and a fingerprint key, resulting in Docker installation failure on the worker and master nodes. The Docker repository and fingerprint have been updated in the “repo.pp” with the correct value, to enable the installation of Docker.

```
        if ($container_runtime == 'docker' and $manage_docker == true) or
            ($container_runtime == 'cri_containerd' and $containerd_install_method ==
'package') {
          apt::source { 'docker':
            location =>
pick($docker_apt_location,'https://download.docker.com/linux/ubuntu'),
            repos    => pick($docker_apt_repos,'stable'),
            release  => pick($docker_apt_release,"xenial"),
            key      => {
              'id'     => pick($docker_key_id,'9DC858229FC7DD38854AE2D88D81803C0EBFCD88'),
              'source' =>
pick($docker_key_source,'https://download.docker.com/linux/ubuntu/gpg'),
            },
          }
        }
```

C. Docker version fix: The “puppetlabs-kubernetes” module also has issues with the Docker versioning and package as well. The docker package and version comes in the configuration files of modules are obsolete and it is not possible to install those versions anymore, which results in failure of the Docker installation during the bootstrapping process of the worker and master nodes. The Docker package name and version have been updated in “init.pp” with the correct values to enable the installation of Docker.

```
  String $container_runtime                                        = 'docker',
  Optional[String] $containerd_version                             = '1.4.3',
  Enum['archive','package'] $containerd_install_method             = 'archive',
  String $containerd_package_name                                  = 'containerd.io',
  Optional[String] $docker_package_name                            = 'docker-ce',
  Optional[String] $docker_version                                 = $facts['os']['family'] ?
{
    'Debian' => '17.03.3~ce-0~ubuntu-xenial',
    'RedHat' => '17.03.1.ce-1.el7.centos',
  },
```

5.2.5.2 *Setup Kubernetes configuration*

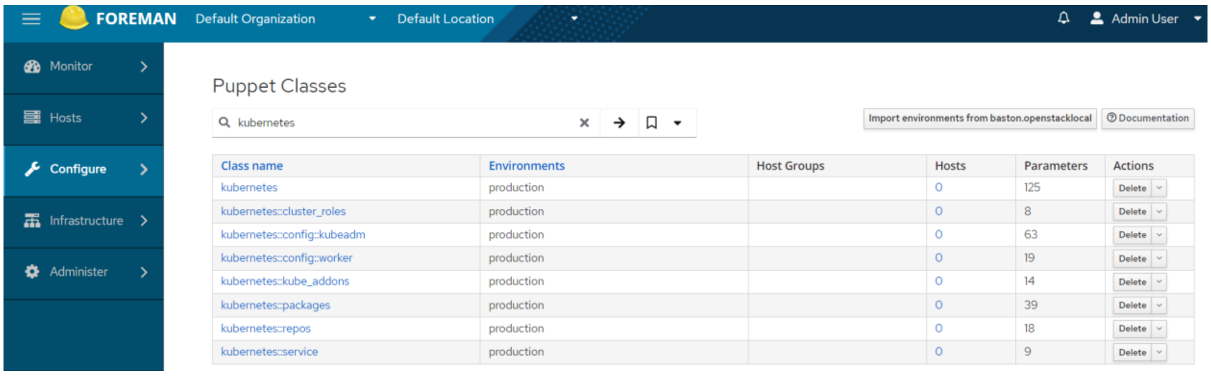
As we have seen before, Kubernetes is an open-source container orchestration system that provides the lifecycle management functionalities for containers such as deployment, scaling, rolling upgrade, configuration management and others. The Puppet modules deployed on the bastion virtual machine to install the Kubernetes cluster also require the configuration to be pre-loaded so that it can be applied during the bootstrapping process. The Kubernetes cluster for this environment comprises of one control/master and three worker nodes. The control/master node hosts the control plane and Etcd (ETCD, 2023), while the worker nodes run the workloads/applications. The control node role is applied on the virtual machine by assigning the “controller => true” label in the Puppet manifest code and a “worker => true” label is assigned for the worker nodes.

The setup comprises of one control node (master) and three worker nodes (worker0, worker1, worker2) hence the manifest code applied for the boot strapping process will be as below:

```
root@master:/etc/puppetlabs/code/environments/production# cat manifests/site.pp
node control {
```

```
class {'kubernetes':
  controller => true,
}
}
node worker0 {
class {'kubernetes':
  worker => true,
}
}

node worker1 {
class {'kubernetes':
  worker => true,
}
}

node worker2 {
class {'kubernetes':
  worker => true,
}
}

node default {}
```

The Puppet classes configured on the bastion for the Kubernetes control and workers nodes are imported into the Foreman within the production environment, and these classes ensure that the control node is installed with a control plane functionality of the Kubernetes cluster, whereas worker0, worker1 and worker2 are added as worker nodes in the cluster to host the relevant workloads. Figure 5-2 shows the list of Kubernetes classes imported to the Foreman for this setup in the production environment.



Puppet Classes

kubernetes

Class name	Environments	Host Groups	Hosts	Parameters	Actions
kubernetes	production		0	125	Delete
kubernetes::cluster_roles	production		0	8	Delete
kubernetes::config::kubeadm	production		0	63	Delete
kubernetes::config::worker	production		0	19	Delete
kubernetes::kube_addons	production		0	14	Delete
kubernetes::packages	production		0	39	Delete
kubernetes::repos	production		0	18	Delete
kubernetes::service	production		0	9	Delete

Figure 5-2: Puppet Classes in Foreman

The Puppet module includes a configuration tool called kubetool. Kubetool auto-generates the Hiera [10]security parameters, the discovery token hash, and other configurations for the Kubernetes cluster. The tool is available as a Docker image, and it generates the configuration files by taking input from the configuration file associated to that environment. The output of the tool generates configuration file for the control plane and the operating system, which are then copied to the Puppet production data directory so that it can be used during the bootstrapping of the Kubernetes cluster (PUPPET, 2023b) (PUPPET, 2023a).

In this setup, configuration files are generated by running the kubetool inside Docker on the bastion virtual machine, the configuration file provides the operating system details, Kubernetes version, runtime container, container network interface (CNI) provider, Etcd related configuration and the Kube API with an associated IP address. Etcd and Kube API IP

[10] Hiera looks up data by following a hierarchy – an ordered list of data sources.(PUPPET, 2023a)

are configured as control node IPs and the final input configuration looks as described below:

```
OS=Debian
VERSION=1.20.1
CONTAINER_RUNTIME=docker
CNI_PROVIDER=flannel
ETCD_INITIAL_CLUSTER=control:10.0.71.26
ETCD_IP=10.0.71.26
KUBE_API_ADVERTISE_ADDRESS=10.0.71.26
INSTALL_DASHBOARD=yes
```

These parameters are stored in the “env” configuration file and provided as an input to the kubetool container to generate the yaml files for bootstrapping of the cluster. The kubetool generates also “Debian.yaml” & “control.yaml” files which contains keys, certificates, relevant IPs, CNI provider, CIDR and Kubernetes version, the complete content of the file can reviewed in Appendix 1B & 1C.

```
docker run --rm -v $(pwd):/mnt --env-file /root/kubernetes/env puppet/kubetool:6.0.0
```

The configuration files are copied to the Puppet production data directory so that they can be used during the bootstrapping process of the Kubernetes cluster, while the “heria.yaml” configuration file is updated with the correct names of the newly created files (highlighted in a snapshot of the code below):

```
---
version: 5
defaults:
hierarchy:
  - name: "Family"
    path: Debian.yaml
  - name: "Host"
    path: control.yaml
  - name: "Per-node data (yaml version)"
    path: "nodes/%{::trusted.certname}.yaml"
  - name: "Other YAML hierarchy levels"
    paths:
      - "common.yaml"
```

5.2.5.3 Setup MLN Configuration for infrastructure

MLN installed on the bastion is used to create the control and worker virtual machines to host the Kubernetes cluster. MLN will also apply certain tasks at the bootstrapping process of each virtual machine via user data option. The complete configuration file prepared for the MLN to create the virtual machines can be read in Appendix 1E and it has the following parameters and properties:

- Declaration of the project name – 5GC,
- Supper class declaration with name agent for each virtual machine and following parameters:
    - Underlying cloud declaration to OpenStack,
    - Image used in virtual machine creation - Ubuntu-20.04-LTS,
    - Flavor used - m1.large,
    - Keypair installed during the boot strapping process – bastion,
    - List of tasks to be applied on each virtual machine via user data:
        - Update operating system,
        - Install repository for the puppet,

            - Install packages – “puppet-agent puppet-common augeas-tools facter” on virtual machine,
            - Update host file of virtual machine with baston IP,
            - Disable firewall and swap partitions on each VM,
            - Update the kernel parameter for bridge,
            - Pull puppet agent file from the git,
            - Update the hostname and IP to baston inside the puppet agent file,
            - Apply the puppet agent file to the virtual machine.
        - Network name used for the virtual machine.
- Apply the super class agent to employ the properties and tasks on each virtual machine such as control, worker0, worker1, worker2 with a host declaration.

```
host control {
     superclass agent
}
host worker0 {
     superclass agent
}
host worker1 {
     superclass agent
}
host worker2 {
     superclass agent
}
```

The logs of the tasks implemented via userdata can be verified on the virtual machine in the log file “cloud-init-output.log” stored inside the “/var/log” directory.

5.2.5.4 End-to-End Kubernetes Cluster Creation

The configuration explained and prepared in the previous sections for MLN, Puppet, Foreman, and Kubernetes are used to create the infrastructure and Kubernetes cluster. This process is automated with the bash program with code described in Appendix 1F and complete workflow described in Figure 5-3.

a. The program starts the deployment,
b. Import the source credential file to connect to the OpenStack cloud, this file includes the URL and login credentials to query the API access to the ALTO cloud.
c. Build the 5GC project via MLN by using the configuration file prepared in section 5.2.5.3 of this chapter,
d. The successful creation of the project will instantiate the creation of virtual machines. In this process, four virtual machines are created and loaded with the configuration, as well as the tasks supplied by the MLN during the bootstrapping process. In case of failure to power on any VM, the script will delete the VM and restart the creation process for that VM. This process will continue until all VMs are created successfully and powered on. OpenStack will show all the VM as “ACTIVE” after completion of this process.

```
root@baston:~# openstack server list -c Name -c Status -c Networks
+-------------------+--------+---------------------------------------+
| Name              | Status | Networks                              |
+-------------------+--------+---------------------------------------+
| worker2.5GC       | ACTIVE | netsys_net=10.0.71.29                 |
| worker1.5GC       | ACTIVE | netsys_net=10.0.71.28                 |
| worker0.5GC       | ACTIVE | netsys_net=10.0.71.27                 |
| control.5GC       | ACTIVE | netsys_net=10.0.71.26                  |
| basiton           | ACTIVE | netsys_net=10.0.71.13, 128.39.121.194  |
+-------------------+--------+----------------------------------------+
```

e. The IP of the control node will be stored inside the variable so that it can be used to create the configuration files for the Kubernetes cluster.

f. Environment file is created as input for the kubetool, this environment file will contain the necessary information for the Kubernetes cluster creation such as Kubernetes version, container run time and networking details and control plane details.

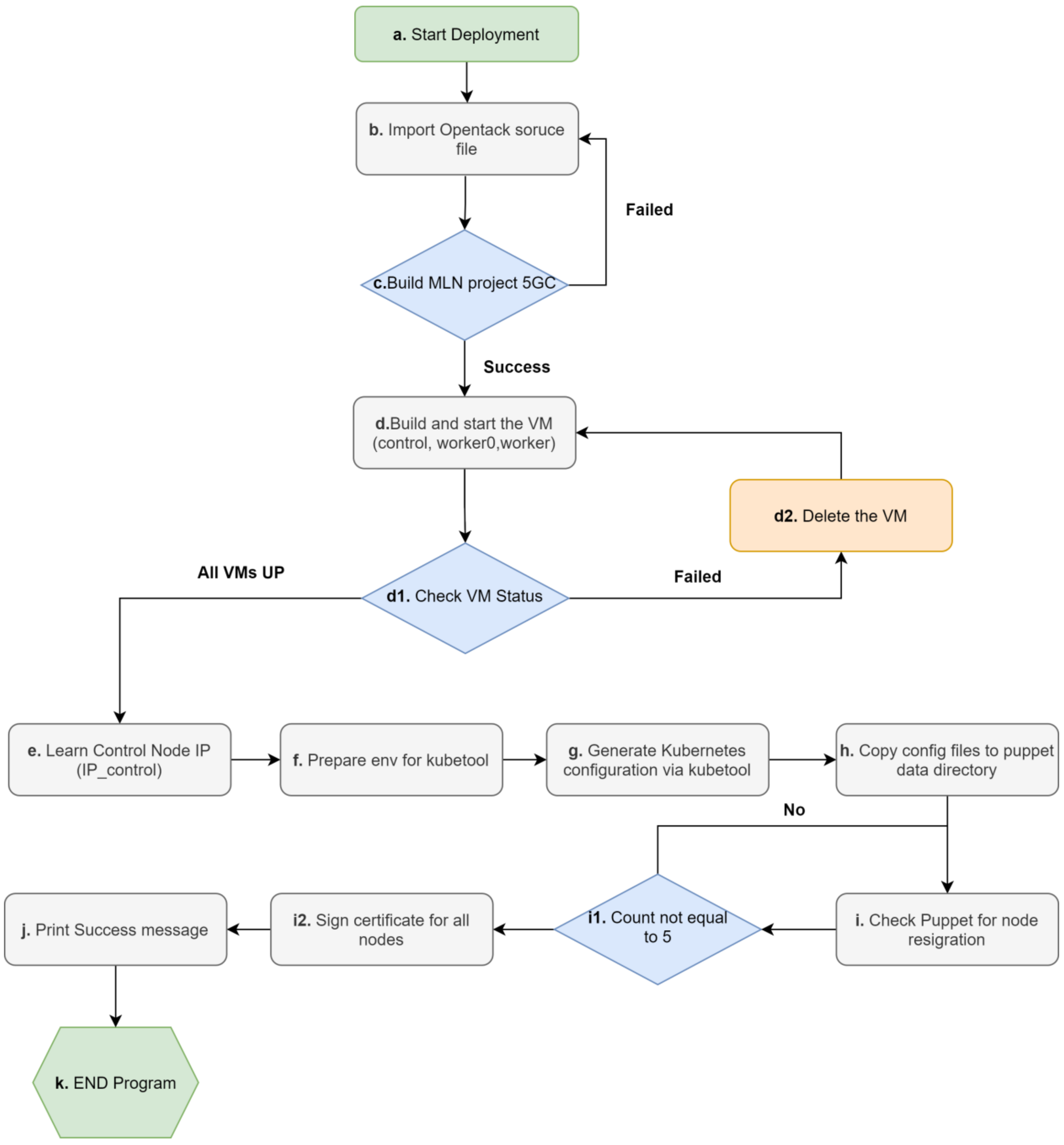

a. Start Deployment
b. Import Opentack soruce file
Failed
c.Build MLN project 5GC
Success
d.Build and start the VM (control, worker0,worker)
d2. Delete the VM
All VMs UP
d1. Check VM Status
Failed
e. Learn Control Node IP (IP_control)
f. Prepare env for kubetool
g. Generate Kubernetes configuration via kubetool
h. Copy config files to puppet data directory
No
i1. Count not equal to 5
i. Check Puppet for node resigration
i2. Sign certificate for all nodes
j. Print Success message
k. END Program

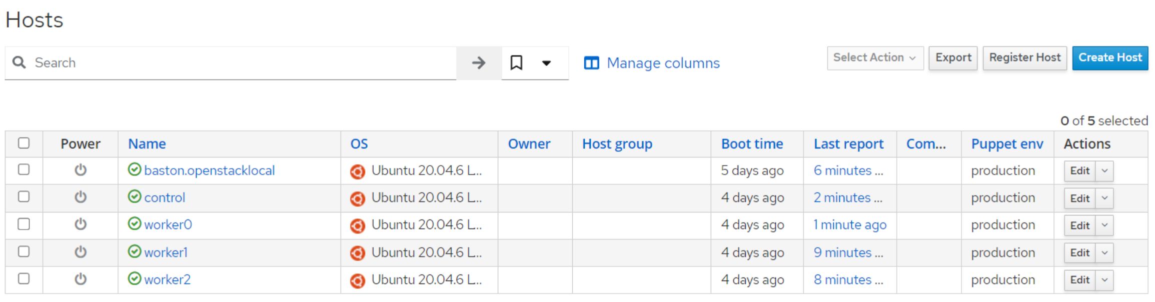
Figure 5-3: Complete Automation flow for Setup creation

g. The environment files are consumed by kubetool to generate the configuration files for the Kubernetes cluster creation in this environment, the files can be viewed in Appendix - Kubernetes – DEBIAN YAML FILE and Kubernetes – CONTROL YAML FILE.

h. Generated files are copied to the production data directory of the puppet, this ensures that puppet server utilizes these files during the cluster creation via puppet module.

i. MLN execute puppet agent installation with correct configuration on the all the virtual machines. The virtual machine will request a puppet master (baston) for registration on successful installation of the puppet agent. The program will wait till all the virtual machine requests for the registration and then it signs the certificates for all virtual machines.

j. When the signing process is complete, puppet server and foreman will start the installation of the Kubernetes cluster on the virtual machines based on the configuration prepared in previous section of this chapter. All the pre-loaded classes will be applied via puppet, when the classes are applied successfully then the nodes will be visible in the foreman GUI with the correct status. Foreman will also show the list of classes applied by puppet and it also shows the any task which are applied to make the nodes in the configured state.



	Power	Name	OS	Owner	Host group	Boot time	Last report	Com...	Puppet env	Actions
☐	⏻	✓ baston.openstacklocal	Ubuntu 20.04.6 L...			5 days ago	6 minutes ...		production	Edit
☐	⏻	✓ control	Ubuntu 20.04.6 L...			4 days ago	2 minutes ...		production	Edit
☐	⏻	✓ worker0	Ubuntu 20.04.6 L...			4 days ago	1 minute ago		production	Edit
☐	⏻	✓ worker1	Ubuntu 20.04.6 L...			4 days ago	9 minutes ...		production	Edit
☐	⏻	✓ worker2	Ubuntu 20.04.6 L...			4 days ago	8 minutes ...		production	Edit

Figure 5-4: VM Status on Foreman

k. Program will exit with the success message and Kubernetes cluster is ready for the installation of 5GC.

5.2.6 5G Core Network Deployment

Open Air Interface (OAI) 5G core network is an implementation of 3GPP compliant (3GPP-TS.23.501, 2022) and it comprises of the following network functions (EURECOM, 2022b):

- Access and Mobility Management Function (**AMF**),
- Authentication Server Management Function (**AUSF**),
- Network Repository Function (**NRF**),
- Session Management Function (**SMF**),
- Unified Data Management (**UDM**),
- Unified Data Repository (**UDR**),
- User Plane Function (**UPF**),
- Network Slicing Selection Function (**NSSF**).

These network functions can be deployed individually or at once on a common Kubernetes cluster. OAI supports three different deployment settings based on the order of network function deployment and functional requirements:

1. **Minimalist deployment:** This deployment includes MySQL (Subscriber Database), AMF, SMF, UPF, NRF only, hence constructing the 5G core network with minimum number of the network functions.
2. **Basic deployment:** Basic deployment includes MySQL (Subscriber Database), UDR, UDM, AUSF, AMF, SMF, UPF, NRF. This deployment has all main required network function without slicing functionality, UDSF is not included by OAI in their deployments, but it will be available in the future releases.
3. **Slicing support:** This flavor of deployment includes MySQL (Subscriber Database), NSSF, UDR, UDM, AUSF, AMF, SMF, UPF, NRF. This deployment supports the slicing functionality along with the other functionalities of the 5G core network.

The Basic and slicing deployment has been used in our testbed to build the 5G core network. The focus is to collect real-time information from the various interfaces and process the data for statistical needs. The deployment of 5G core networks requires an ordinary Kubernetes environment installed and configured properly, and this has been achieved already by the automation process explained in the former sections of this chapter. As a verification, a connection is made towards the control node of the Kubernetes cluster and the status of each node is confirmed as "Ready."

```
root@control:~# kubectl get nodes
NAME      STATUS   ROLES                  AGE   VERSION
control   Ready    control plane,master   6d    v1.20.1
worker0   Ready    <none>                 6d    v1.20.1
worker1   Ready    <none>                 6d    v1.20.1
worker2   Ready    <none>                 6d    v1.20.1
root@control:~#
```

Similarly, the "kubectl" utility is installed on the bastion host and the Kubernetes cluster "admin.conf" file is copied to the bastion so that the cluster can be managed from the bastion.

5.2.6.1 HELM setup

Kubernetes enables the orchestration of containers. It manages and simplifies the lifecycle of the containers by grouping multiple microservices into a single deployment. This approach solves the lifecycle management issue of the microservices, but brings version management, updating, rollbacks and resource allocation challenges to the microservice application. The Helm tool is a package manager for Kubernetes supporting applications and it enables the packaging, creation, deployment, and upgrade of different applications designed with a microservice architecture on a Kubernetes cluster. Kubernetes based applications are deployed using YAML configuration files and Helm helps to maintain a single deployment YAML file with version information by using helm charts. A Helm chart is created for the microservices, and it has all the necessary resources to deploy an application to a Kubernetes cluster. The charts include YAML configuration files for replica set, demon set, deployments, services, secrets, and config maps required for the Kubernetes based application (Schmitt, 2023).

The OAI 5G core network requires helm package to be installed on the bastion virtual machine, thus HELM v3.11.2 has been deployed on the bastion.

```
root@baston:~# helm version
version.BuildInfo{Version:"v3.11.2", GitCommit:"912ebc1cd10d38d340f048efaf0abda047c3468e",
GitTreeState:"clean", GoVersion:"go1.18.10"}
root@baston:~#
```

OAI 5G also requires helms spray to support the sequential deployment of the network function on the Kubernetes cluster. Helm spray v4.0.11 is installed on the baston to carry the installation of 5G core network (Thales, 2023).

```
root@baston:~# helm plugin list
NAME     VERSION          DESCRIPTION
spray    4.0.11-beta.3    Helm plugin for upgrading sub-charts from umbrella chart with
dependency orders
root@baston:~#
```

5.2.6.2 MULTUS Setup

Kubernetes pods by default are deployed with a single interface and all the communication happens via the single interface of the pod. The restriction of using a single interface for the pod can be overcome with the installation of additional plugin called Multus CNI, which is a container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods. OAI 5G requires multiple interfaces for the AMF and UPF network functions, hence MULTUS CNI version v3.9.3 is deployed on the lab Kubernetes cluster. Multus CNI allows multiple interfaces on the AMF and UPF pods, hence allowing the configuration of different IPs to be used for different communication. The additional interface is attached to the pod as net0 or net1 for communication to the other network, whereas eth0 connects Kubernetes cluster network to attach with Kubernetes a server/services as show in Figure 5-5.

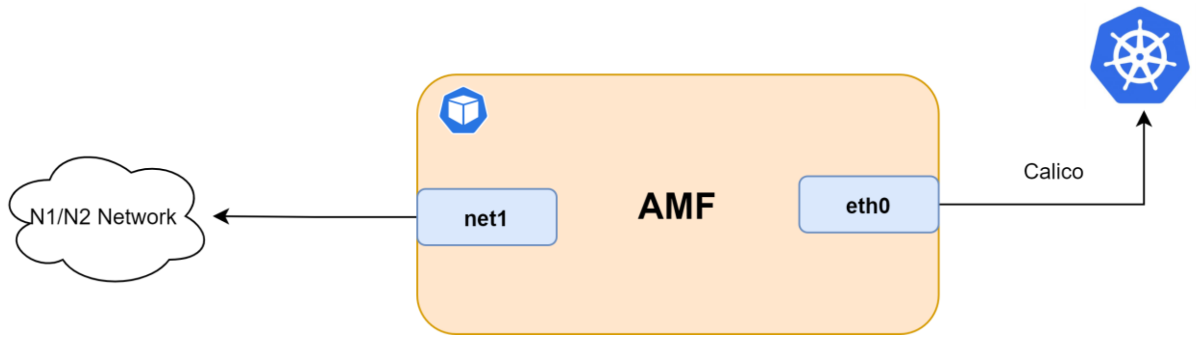


Figure 5-5 MULTUS Setup

5.2.6.3 OAI 5G core network charts

OAI 5GC is built based on Release-16 core network of 3GPP, but OAI has maintained its own versioning as well. The OAI 5G core network installed for this thesis is on version v1.5.0. This is delivered in the form of helm charts and these charts are cloned from git. All the OAI core network charts are present under oai-5g-core folder, which includes charts for the three different deployment flavors discussed above:

- oai-5g-mini for minimalist deployment,
- oai-5g-basic for basic deployment,
- oai-5g-slicing for slicing deployment.

OAI 5G core network charts are prepared to deploy the 5G service-based architecture instead of deploying the reference-based architecture. The basic and slicing core network is deployed for this thesis work, but this section only covers the deployment of the basic as the procedure is the same. The charts mainly consist of two parent files - OAI 5G – CHART.YAML and OAI 5G – VALUES.YAML, the chart.yaml file refers to the individual network charts path stored at the parent location of the folder "oai-cn5g-fed" as the individual folder contains all the information to deploy the network functions such as configmap, deployment, Multus, RBAC, service and service account details. All the configurable parameters required to deploy the network function are taken from the parent "values.yaml" file and it includes the image name, location to pull the image (docker hub can be used to pull original images), networks and other configuration parameters. There is a dedicated configuration in global value.yaml file for each network function with associated secret. The secret is created under a docker username **"furqanahmad"** with name "regcred" and it is utilized by all the network function during the installation. The snap for the AUSF deployment inside the "value.yaml" file is shown below:

```
oai-ausf:
  enabled: true
  weight: 3
  kubernetesType: Vanilla #Openshift/Vanilla Vanilla for Upstream Kubernetes
  nfimage:  # image name either locally present or in a public/private repository
    repository: docker.io/oaisoftwarealliance/oai-ausf           ## The image will be
pulled from dockerhub
    version: v1.5.0                                    ## The branch to be used to pull
from dockerhub
    # pullPolicy: IfNotPresent or Never or Always
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: "regcred"
  nodeSelector: {}
```

The deployment of the network functions needs to happen in a sequence as there is dependency of the network functions between each other. The deployment sequence to be followed is MySQL, NRF, UDR, UDM, AUSF, AMF, UPF and SMF so Helm Spray is used to implement this sequence during the installation. The OAI 5G – CHART.YAML file has section dependencies which includes the condition and sub-charts to deploy the network functions. The "value.yaml" file has a weight applied to each sub-chart as described below, to enforce the sequential installation of network functions.

Table 5-2: OAI 5G Deployment Sequence

Network Function	Weight
MySQL, NRF	0
UDR	1
UDM	2
AUSF	3
AMF	4
UPF	5
SMF	6

5.2.6.4 OAI 5G core network Instantiation

The OAI 5G workload should run on worker nodes of the Kubernetes cluster and this is achieved by labeling all the worker nodes with a label "cnf=load" which is being used in global "value.yaml" file with a "nodeSelector" parameter.

```
kubectl label no worker0 cnf=load
kubectl label no worker1 cnf=load
kubectl label no worker2 cnf=load
```

Additionally, this deployment is based on service-based architecture hence there is no need to configure the static IP for the network functions and the network functions will discover each other using NRF by using their FQDN. For example, SMF registers with NRF using NRF FQDN – "oai-nrf-svc.oai.svc.cluster.local". The OAI 5G core network is deployed in separate name space "oai", which is created on the Kubernetes cluster.

```
root@baston:~# kubectl get ns
NAME                STATUS   AGE
default             Active   6d5h
kube-flannel        Active   6d5h
kube-node-lease     Active   6d5h
kube-public         Active   6d5h
kube-system         Active   6d5h
oai                 Active   6d4h
root@baston:~#
```

The OAI 5G core network helm charts are deployed in the namespace "oai", and the installation is performed via "helm spray" to control the network function deployment in the sequence used from the weight assigned in the "value.yaml" file. The installation output is shown below:

```
root@baston:~/oai-cn5g-fed/charts/oai-5g-core/oai-5g-basic#  helm dependency update
Saving 8 charts
Deleting outdated charts
root@baston:~/oai-cn5g-fed/charts/oai-5g-core/oai-5g-basic# helm spray . -n oai
[spray] processing chart from local file or directory "."...
[spray] deploying solution chart "." in namespace "oai"
[spray] processing sub-charts of weight 0
[spray]   > upgrading release "mysql": deploying first revision (appVersion 8.0.31)...
[spray]     o release: "mysql" upgraded
[spray]   > upgrading release "oai-nrf": deploying first revision (appVersion v1.5.0)...
[spray]     o release: "oai-nrf" upgraded
[spray]   > waiting for liveness and readiness...
[spray] processing sub-charts of weight 1
[spray]   > upgrading release "oai-udr": deploying first revision (appVersion v1.5.0)...
[spray]     o release: "oai-udr" upgraded
[spray]   > waiting for liveness and readiness...
[spray] processing sub-charts of weight 2
[spray]   > upgrading release "oai-udm": deploying first revision (appVersion v1.5.0)...
[spray]     o release: "oai-udm" upgraded
[spray]   > waiting for liveness and readiness...
[spray] processing sub-charts of weight 3
[spray]   > upgrading release "oai-ausf": deploying first revision (appVersion v1.5.0)...
[spray]     o release: "oai-ausf" upgraded
[spray]   > waiting for liveness and readiness...
[spray] processing sub-charts of weight 4
[spray]   > upgrading release "oai-amf": deploying first revision (appVersion v1.5.0)...
[spray]     o release: "oai-amf" upgraded
[spray]   > waiting for liveness and readiness...
[spray] processing sub-charts of weight 5
[spray]   > upgrading release "oai-spgwu-tiny": deploying first revision (appVersion
v1.5.0)...
[spray]     o release: "oai-spgwu-tiny" upgraded
[spray]   > waiting for liveness and readiness...
[spray] processing sub-charts of weight 6
[spray]   > upgrading release "oai-smf": deploying first revision (appVersion v1.5.0)...
[spray]     o release: "oai-smf" upgraded
[spray]   > waiting for liveness and readiness...
[spray] upgrade of solution chart "." completed in 3m13s
root@baston:~/oai-cn5g-fed/charts/oai-5g-core/oai-5g-basic#
```

The installation process will deploy all the network functions from the corresponding charts, and the process deploys the charts for all network functions. This can be verified via the "helm" list command as below:

```
root@baston:~/oai-cn5g-fed/charts/oai-5g-core/oai-5g-basic# helm list -n oai
NAME            NAMESPACE       REVISION        UPDATED
STATUS          CHART                   APP VERSION
mysql           oai             1               2023-04-10 10:06:35.809788881 +0000 UTC
deployed        oai-5g-basic-v1.5.0     master-v1.5.0
```

```
oai-amf         oai             1               2023-04-10 10:08:15.629797395 +0000 UTC
deployed        oai-5g-basic-v1.5.0     master-v1.5.0
oai-ausf        oai             1               2023-04-10 10:07:53.456289258 +0000 UTC
deployed        oai-5g-basic-v1.5.0     master-v1.5.0
oai-nrf         oai             1               2023-04-10 10:06:36.783520766 +0000 UTC
deployed        oai-5g-basic-v1.5.0     master-v1.5.0
oai-smf         oai             1               2023-04-10 10:08:44.123511674 +0000 UTC
deployed        oai-5g-basic-v1.5.0     master-v1.5.0
oai-spgwu-tiny  oai             1               2023-04-10 10:08:32.563639801 +0000 UTC
deployed        oai-5g-basic-v1.5.0     master-v1.5.0
oai-udm         oai             1               2023-04-10 10:07:47.247959768 +0000 UTC
deployed        oai-5g-basic-v1.5.0     master-v1.5.0
oai-udr         oai             1               2023-04-10 10:07:35.754943219 +0000 UTC
deployed        oai-5g-basic-v1.5.0     master-v1.5.0
root@baston:~/oai-cn5g-fed/charts/oai-5g-core/oai-5g-basic#
```

The output above shows the version of the deployed charts for each network function, and it is possible to upgrade each network function individually via its own charts. The completion of the deployment will create all the network function pods in the namespace called “oai.” All the network function has the status “Running,” and the ready ones will show the health of the network function, so that it can accept the traffic. The complete list of network function deployed are listed below:

```
root@baston:~# kubectl get po -n oai
NAME                                  READY   STATUS    RESTARTS   AGE
mysql-56778b4976-mnf9p                1/1     Running   0          6d4h
oai-amf-5b77969675-mc426              2/2     Running   0          6d4h
oai-ausf-84f7898648-zdwl9             2/2     Running   0          6d4h
oai-nrf-697bb54847-8r882              2/2     Running   0          6d4h
oai-smf-dc7f64467-sjwdx               2/2     Running   0          6d4h
oai-spgwu-tiny-6d9b47986f-kbbb4       2/2     Running   0          6d4h
oai-udm-d6f55666b-rbhg4               2/2     Running   0          6d4h
oai-udr-54c6459774-ncf96              2/2     Running   0          6d4h
root@baston:~#
```

The OAI 5G core network deployment will also create the list of services for each network function and these services are associated with the network function pod, which is used for the communication within the network function. The list of services created as part of this deployment are listed below:

```
root@baston:~# kubectl get svc -n oai
NAME                   TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)             AGE
mysql                  ClusterIP   10.100.57.72   <none>        3306/TCP            6d4h
oai-amf-svc            ClusterIP   None           <none>        80/TCP,8080/TCP     6d4h
oai-ausf-svc           ClusterIP   None           <none>        80/TCP,8080/TCP     6d4h
oai-nrf-svc            ClusterIP   None           <none>        80/TCP,8080/TCP     6d4h
oai-smf-svc            ClusterIP   None           <none>        80/TCP,8080/TCP     6d4h
oai-spgwu-tiny-svc     ClusterIP   None           <none>        8805/UDP,80/TCP     6d4h
oai-udm-svc            ClusterIP   None           <none>        80/TCP,8080/TCP     6d4h
oai-udr-svc            ClusterIP   None           <none>        80/TCP,8080/TCP     6d4h
root@baston:~#
```

6 DATA GENERATION AND ACQUISITION IN 5G NETWORKS

The 5G core network has multiple network elements to provide the corresponding 5G feature and functionality to enable the network to function efficiently. As seen before, these network elements are referred to as Network Functions (NFs) in the 5G core network. The core network function must communicate with other network functions within and outside the home network, to provide certain functionality, which also requires the network functions to have certain interoperability features. The communication between the network functions is based on the standard protocols defined by 3GPP, which enables the communication between network functions from different vendors within the same network or other networks. The design decision on the interfaces used to establish diverse types of communication by network functions differs across various providers, and the standard does not enforce any compliance on this.

This communication storm within the network generates an excessive number of messages from each network function. Subsequently, the following chapter identifies and addresses the metrics across various network functions, associated protocols, and interfaces, for each network function to be probed to gather information on the functioning of a 5G core network.

6.1 Data Generation in 5G Networks

As we have deployed a reduced 5G core network to provide basic functionality, other unnecessary network functions are not included since we aim to demonstrate the deployment of a data collection and monitoring framework. The solution shall work for every function since the NFs use a common method for exchanging information.

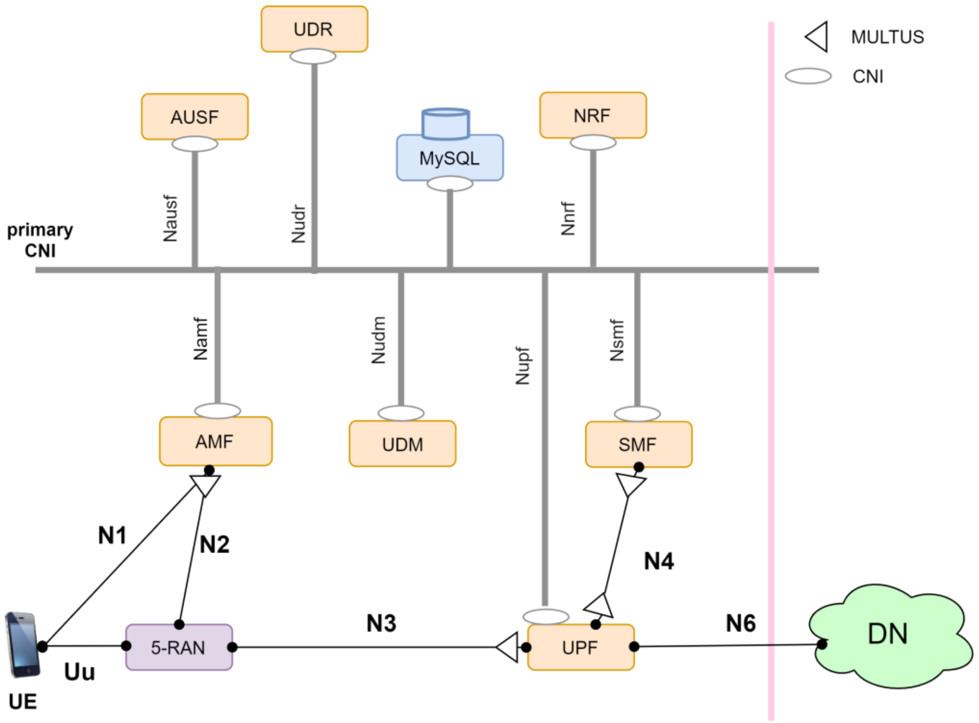


Figure 6-1: OAI 5G Core Network

Figure 6-1 shows the 5G core network functions deployed in scope of this thesis. The network functions are divided into two categories based on the interface's requirements:

- Single interface: Network functions requiring only the single interface inside pod for all internal and external communication via default Container Network Interface (CNI).
- Multiple interfaces: Network functions requiring multiple interfaces inside the pod for all internal and external communication via Multus plugin installed on the Kubernetes cluster.

The network functions support various protocols and accordingly process immense number of messages in the network. The flow includes processing of messages at the network function, retrieving the information from other network functions, forwarding the messages to other network functions, sharing the information with other network functions and more.

Figure 6-2 shows the marking of each interface along with protocol mapping inside the 5G core network.

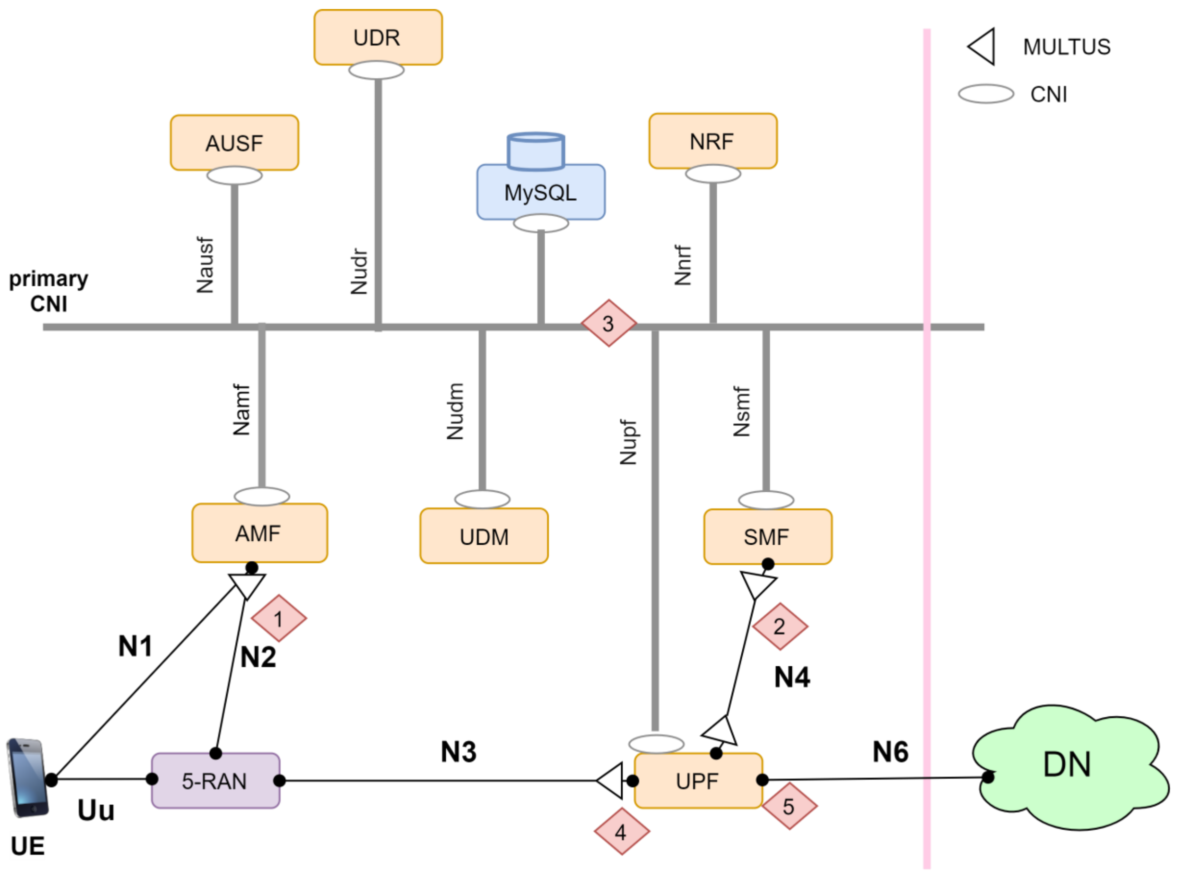


Figure 6-2: 5GC Protocol Categorization

The protocol-centric communication fits into five main categories:

1. Next-Generation Application Protocol (NGAP) over N2 reference point, with only NG signaling traffic (NG-C) between the AMF and gNB.
2. Packet Forwarding Control Protocol (PFCP) over N4 reference point between the SMF and UPF.

3. HTTP2 over various reference points, uses the default CNI between all the network functions.
4. General Packet Radio Service Tunnelling Protocol User (GTP-U) over N3 reference point with only NG user plane (NG-U) between gNB and UPF.
5. Traffic towards the Data Network (DN) from UPF over N6 reference point.

The AMF and SMF participate in the signaling messages and the UPF for the user plane traffic, hence this thesis includes the analysis of protocols relevant to AMF, SMF and UPF to be studied in the following sub sections.

6.1.1 NG Application Protocol (NGAP) over N2

The 5G architecture has a New Radio (NR) and Next Generation (NG) network interfaces. NR network interfaces covers the communication within the radio network and includes E1 interface, F1 interface, Xn interface, and F2 interface. NG interfaces facilitate the inter-connection of the NG-RAN to the 5G core, or specifically AMF, with separation of NG interface Radio Network functionality and Transport network functionality (ETSI-TS.138.410, 2020).

The communication between 5G RAN and 5GC is over NG interface and it splits into signaling (NG-C) and user plane (NG-U) protocols. The NG-C interface provides the control plane functionality between NG-RAN and the 5G Core network. The network function from the 5GC network involved in this communication is AMF and the protocol used is NGAP over reference point N2, whereas the NG-U interface provides the user plane functionality between NG-RAN and the 5G Core network. The network function from the 5GC involved in this communication is AMF, and the protocol used is General Packet Radio Service Tunnelling Protocol User-Plane (GTP-U) over reference point N3.

6.1.1.1 NGAP Stack

The protocol stack is the layered collection of protocols that work together and provide communication services. Each layer is responsible for a specific task and each layer can be mapped to the OSI model described before. The OSI model creates more robust and reliable communication. The layered architecture is essential for completing the NGAP stack and the functionality of the entire protocol is broken down into component protocols that are separately managed and can be treated and designed independently from one another. The NGAP layered stack works in such a way that the lower layer always provides the service to the upper layer, therefore the packet traverses through all the layers and the layer at the source communicates with the same layer at the destination that contains the peer-to-peer protocols (Afneh, 2023).

As stated above, the NGAP stack also relies on the lower layer, and peer-to-peer protocol communication is established between the source and destination system across layers. Figure 6-3 shows the protocol stack for NGAP between the gNB and AMF over a N2 interface. The physical layer provides the physical connectivity between the gNB and AMF and then IP provides the layer-3 connectivity. The transport level connectivity is achieved via Stream Control Transmission Protocol (SCTP) that ensures robust and sequential transport of data. SCTP provides multihoming support hence multiple IP addresses can be associated to a gNB and an AMF, for enabling the transport failover between the redundant paths.

NGAP is an application layer protocol between the gNB and AMF that enables the multiple application-specific procedures between a gNB and an AMF, as defined in the subsection that follows.

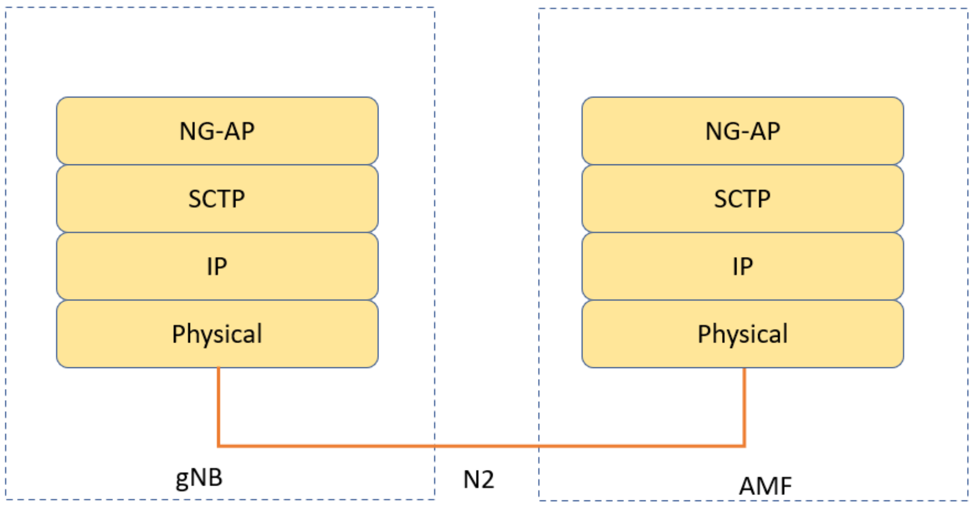


Figure 6-3: NGAP Protocol Stack

6.1.1.2 NGAP Procedure

NGAP provides non-UE associated, and UE associated signaling services between a gNB and an AMF. The non-UE associated services are related to the Next-Generation (NG) interface between the NG RAN and AMF, by utilizing non UE associated signaling connection such as NG setup, AMF configuration update, RAN configuration update and other similar procedures defined in Table 6-1 below. The UE associated signaling services are only connected to specific UEs and the NGAP procedures are only applicable on given UEs, such as initial context setup/resume/suspend, PDU session resource setup/modify/release, UE context setup/modification/release and other similar procedures defined in Table 6-1 (ETSI-TS.138.413, 2020).

Table 6-1: NGAP procedures

Procedures	Description
NG Setup	The procedure is initiated by NG RAN towards AMF. It uses non-UE-associated signaling and exchange application-level data needed for the NG-RAN node and the AMF to correctly interoperate on the NG-C interface.
AMF Configuration Update	The procedure is initiated by AMF towards NG RAN. It uses non-UE-associated signaling and update application-level configuration data needed for the NG-RAN node and AMF to interoperate correctly on the NG-C interface.
RAN Configuration Update	The procedure is initiated by NG RAN towards AMF. It uses non-UE-associated signaling and update application-level configuration data needed for the NG-RAN node and AMF to interoperate correctly on the NG-C interface.

NG Reset	The procedure is initiated by AMF towards NG RAN. It uses non-UE-associated signaling and initialize or re-initialize the RAN in the event of a failure in the 5GC.
Initial Context Setup	The procedure is initiated by AMF towards NG RAN. It uses UE-associated signaling and establish the necessary overall initial UE context at the NG RAN node.
UE Context Suspend	The procedure is initiated by NG RAN towards AMF. It uses UE-associated signaling and suspend the UE-associated logical NG-connection and the NG-U transport bearer with the 5GC.
UE Context Resume	The procedure is initiated by NG RAN towards AMF. It uses UE-associated signaling and to resume the UE context, the suspended UE-associated logical NG-connection, and the related NG-U transport bearer in the 5GC.
PDU Session Resource Setup	The procedure is initiated by AMF towards NG RAN. It uses UE-associated signaling and assign resources on NG-U for PDU sessions and the corresponding QoS flows.
PDU Session Resource Modify	The procedure is initiated by AMF towards NG RAN. It uses UE-associated signaling and enables the configuration modification of already established PDU session for UE.
PDU Session Resource Modify Indication	The procedure is initiated by NG RAN towards AMF. It uses UE-associated signaling and enables the NG RAN to request modification of established PDU session for UE.
PDU Session Resource Release	The procedure is initiated by AMF towards NG RAN. It uses UE-associated signaling and release already established PDU session resources for a given UE.
UE Context Setup	The procedure is initiated by AMF towards NG RAN. It uses UE-associated signaling and establish the necessary overall initial UE context at the NG RAN node.
UE Context Modification	The procedure is initiated by AMF towards NG RAN. It uses UE-associated signaling and modify the established UE context.
UE Context Release	The procedure can be initiated by AMF or NG RAN. It uses UE-associated signaling and enable the release of the UE-associated logical NG-connection.
Uplink RAN configuration Transfer	The procedure is initiated by NG RAN towards AMF. It uses non-UE-associated signaling and transfer RAN config information from the NG-RAN node to the AMF.
Downlink RAN configuration Transfer	The procedure is initiated by AMF towards NG RAN. It uses non-UE-associated signaling and transfer RAN config information from the AMF to the NG-RAN node.
UE Radio Capability Check	The procedure is initiated by AMF towards NG RAN. It uses UE-associated signaling and enables AMF to request the NG-RAN node to derive and provide an indication on

	whether the UE radio capabilities are compatible with the network configuration for IMS voice.
UE Radio Capability ID Mapping	The procedure is initiated by NG RAN towards AMF. It uses non-UE-associated signaling and request from the AMF UE Radio Capability information mapped to the UE Radio Capability ID.
Handover Preparation	The procedure is initiated by NG RAN towards AMF. The purpose is to request the preparation of resources at the target side via the 5GC.
Handover Cancellation	The procedure is initiated by NG RAN towards AMF. It uses UE-associated signaling and purpose is to cancel an ongoing handover preparation.
Handover Resource Allocation	The procedure is initiated by AMF towards NG RAN. The purpose reserve resources at the target NG-RAN node for the handover of a UE.
Path Switch Request	The procedure is initiated by NG RAN towards AMF. It uses UE-associated signaling and establish a UE associated signaling connection to the 5GC and request the switch of the downlink termination point of the NG-U transport bearer towards a new termination point.
Transport of NAS Messages	Transport of NAS messages include Initial UE Messages, Downlink NAS Transport, Uplink NAS Transport, NAS non delivery indication and Reroute NAS Request

6.1.1.3 NGAP Counters

The counters related to the N1 and N2 reference points can be collected from AMF. The messages processed by the AMF are captured and the relevant metrics can be stored to understand the network condition. The N1 reference point counters can be divided into seven categories:

- AMF NAS Registration,
- AMF NAS Service Request,
- AMF NAS Deregistration,
- AMF NAS Authentication counters,
- AMF NAS Security mode counters,
- AMF NAS uplink and downlink transport,
- AMF NAS PDU Session.

The Table 6-2 describes the list of recommended counters for N1 reference points to be stored and utilized by an AI system. The counters are associated with messages received, sent, and rejected based on the response and request of the procedure.

Table 6-2: N1 Interface Counters

Category	Recommended counters
AMF NAS Registration	• Initial Registration request received. • Mobility Update request received. • Periodic Update request received.

	• Registration Accept sent. • Registration Complete received. • Registration Reject sent. • registration rejected due to 5GMM cause #.
AMF NAS Service Request	• Service Request received. • Service Reject sent. • Service Accept sent. • Service request rejected due to 5GMM #
AMF NAS Deregistration	• Deregistration Accept sent. • Deregistration Request sent. • Deregistration Request received. • Deregistration Accept message received.
AMF NAS Authentication counters	• Authentication Request sent. • Authentication Response received. • Authentication failure received. • Authentication request rejected due to 5GMM #
AMF NAS Security mode counters	• Security mode command Request sent. • Security mode command Request received. • Security mode command request rejected due to 5GMM #
AMF NAS uplink and downlink transport	• UL NAS transport received. • DL NAS transport sent. • DL NAS transport received.
AMF NAS PDU Session	• PDU session establishment request received. • PDU session establishment release request received. • PDU session establishment release complete received. • PDU session establishment modification request. • PDU session establishment modification reject. • DL NAS transport sent. • DL NAS transport received.

Based on the NGAP procedure defined in the section 6.1.1.2 NGAP Procedure, counters can be accumulated from the messages received on the AMF. The NGAP counter can be divided into seven main categories:

- NGAP PDU Session Management,
- NGAP UE Context Management,
- NGAP UE Mobility Management,
- NAGP NAS Transport,
- NGAP Location Reporting,
- NGAP UE Radio Capability Management,
- NGAP Configuration Transfer.

Table 6-3 lists the recommended NGAP counters for each NGAP procedure category. The counters are associated with messages received, sent, and rejected based on the response and request of the NGAP procedure.

Table 6-3: NGAP counters

Category	Recommended counters
NGAP PDU Session Management	• PDU SESSION RESOURCE SETUP RESPONSE RCVD messages received. • PDU SESSION RESOURCE RELEASE RESPONSE messages received. • PDU SESSION RESOURCE SETUP REQUEST messages sent. • PDU SESSION RESOURCE RELEASE COMMAND messages sent. • PDU SESSION RESPONSE MODIFY INDICATION messages received. • PDU SESSION RESOURCE MODIFY CONFIRM messages sent. • PDU SESSION RESOURCE MODIFICATION RESPONSE messages received. • PDU SESSION RESOURCE MODIFY REQUEST messages sent.
NGAP UE Context Management	• UE_CONTEXT RELEASE REQUEST messages received. • UE CONTEXT RELEASE COMPLETE messages received. • INITIAL CONTEXT SETUP RESPONSE messages received. • UE CONTEXT RELEASE COMMAND messages sent. • INITIAL CONTEXT SETUP REQUEST SENT messages sent. • UE CONTEXT MODIFICATION REQUEST messages sent. • UE CONTEXT MODIFICATION RESPONSE messages received. • UE CONTEXT MODIFICATION FAILURE messages received.
NGAP UE Mobility Management	• PATH SWITCH REQUEST messages received. • PATH SWITCH REQUEST ACK messages sent. • PATH SWITCH REQUEST FAILURE messages sent. • HANDOVER REQUIRED messages received. • HANDOVER REQUEST messages sent. • HANDOVER FAILURE messages received. • HANDOVER CANCEL messages received.
NGAP NAS Transport	• INITIAL UE MESSAGE messages received. • UPLINK NAS TRANSPORT messages received. • DOWNLINK NAS TRANSPORT messages sent.

NGAP Location Reporting	• LOCATION REPORTING CONTROL messages sent. • LOCATION REPORTING FAILURE INDICATION messages received. • LOCATION REPORT messages received.
NGAP UE Radio Capability Management	• UE RADIO CAPABILITY CHECK REQUEST messages sent. • UE RADIO CAPABILITY CHECK RESPONSE messages received. • UE RADIO CAPABILITY INFO INDICATION messages received.
NGAP Configuration Transfer	• UL RAN Configuration Transfer messages received.

6.1.2 Packet Forwarding Control Protocol (PFCP) over N4

The Packet Forwarding Control Protocol (PFCP) is a defined protocol by the 3GPP and resides between the control and user plane network function described in the TS 29.244 specification (ETSI.129.244, 2020). In a 5G core network, PFCP provides the control means for SMF to manage packet processing, as well as means for UPF to perform forwarding. PFCP was also used as a protocol in the Control and User Plane Separation (CUPS) architecture of the Evolved Packet Core (EPC) between the Serving Gateway Control plane function (SGW-C) and Serving Gateway User plane function (SGW-U) as Sxa interface, as well as an between the PDN Gateway Control plane function (PGW-C) and PDN Gateway User plane function (PGW-U) as Sxb interface. In a 5G core network, the Control Plane (CP) is managed by the SMF, while the Uuer plane is managed by the UPF. The SMF controls the packet processing and forwarding in UPF by establishing, modifying or deleting PFCP sessions (ETSI.129.244, 2020) as follows:

- Provision Packet Detection Rules (PDR): PDRs include the rules for matching the data packets so that additional rule can be applied.
- Forwarding Action Rules (FAR): FARs are applied on the packets with mathcing PDRs and it decides for dropping, forwarding, buffering or duplicating a packet.
- Buffering Action Rules (BAR): BARs determine the amount of data to be buffered on UPF. The UPF is instructed to forward the DL packets to the SMF when applying buffering in the SMF (ETSI.129.244, 2020).
- QoS Enforcement Rules (QER): QoS rules enforced such as gating and QoS control, flow, and service level marking.
- Usage Reporting Rules (URR): URRs provide the rules to monitor the usage at the UPF and generate the report for SMF to apply the charging functionality.

6.1.2.1 PFCP Protocol Stack

As stated, the protocol stack is the layered collection of protocols that work together and provide communication services. Every layer is responsible for a specific task, and it is referenced to the OSI model. The PFCP stack relies on the lower layers and ensures peer-to-peer protocol communication is established between the source and destination systems within the same layer. Figure 6-4 shows the protocol stack for PFCP between the SMF and

UPF over the N4 reference point. The physical layer provides the physical connectivity between the SMF and UPF and then the IP provides the layer-3 connectivity. The transport layer connectivity is achieved via the User Datagram Protocol (UDP) that uses a connectionless communication model. PFCP is an application layer protocol between the SMF and UPF to enable the multiple application specific procedures between the network functions, as defined in the consequent section below.

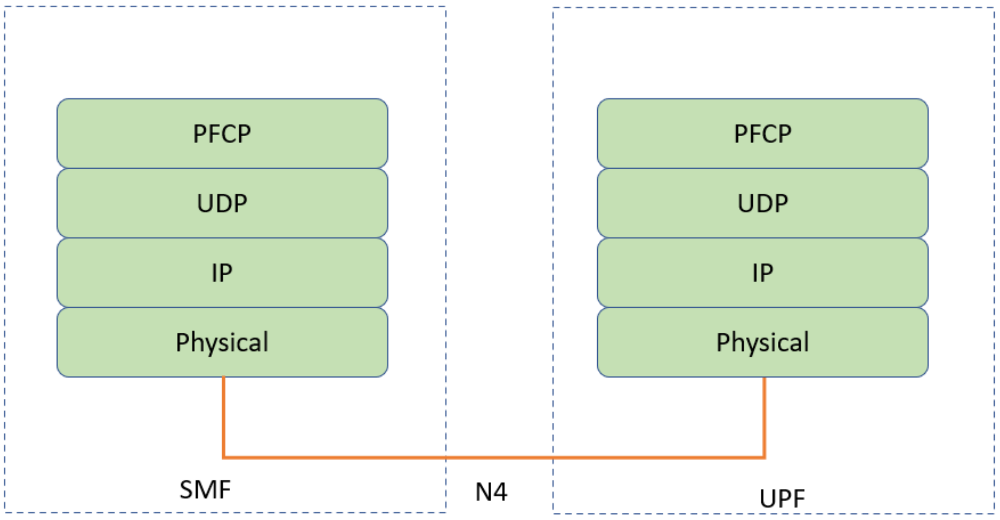


Figure 6-4: PFCP Protocol Stack

6.1.2.2 PFCP Procedure

The PFCP provides node-related and session-related procedures between the SMF and the UPF in a 5G core network over the N4 reference point. Node-related procedures are associated to the SMF and UPF exchanging received and sent messages related to the nodes such as Heartbeat, Load control, overload control, PFCP Packet Flow Description (PFD) management and PFCP association setup/update/release explained in Table 6-4. Session related procedures are associated to the messages received and sent related to the session between SMF and UPF such as PFCP Session Establishment / Modification / Deletion and PFCP Session Report explained in Table 6-4.

Table 6-4: PFCP Procedures

Procedure	Description
Heartbeat	The procedure can be initiated by SMF or UPF, as both nodes have capability to send and receive heartbeat. This procedure is node-related and helps peer node to find out if the peer PFCP entity is alive.
Load Control	The procedure is initiated by the UPF towards the SMF. This procedure is node-related and enables the UPF to send its payload information to the SMF to adaptively balance the PFCP session load across all the provisioned UPFs and according to their effective load.
Overload Control	The procedure is initiated by the UPF towards the SMF. This procedure is node-related and enables the UPF to enable protection in conditions of overload, by informing the SMF

	to reduce its incoming signaling load so that the traffic is reduced according to the available signaling capacity at the UPF.
PFCP PFD Management	The procedure can be initiated by the SMF or the UPF. This procedure is node-related and used by the SMF and UPF to provision PFDs.
PFCP Association Setup	The procedure is between the SMF and the UPF and can be initiated by either node. This procedure is node-related and enables the SMF to use the resources of the UPF subsequently.
PFCP Association Update	The procedure is between the SMF and the UPF and can be initiated by either node. This procedure is node-related and enables modification of existing PFCP associations.
PFCP Association Release	The procedure is between the SMF and the UPF and is initiated by the SMF. This procedure is node-related and enables the SMF to terminate a PFCP association.
PFCP Node Report	The procedure is between the SMF and the UPF and it is initiated by the UPF. This procedure is node-related and used by the UPF to report information unrelated to the session, such as path failures.
PFCP Session Establishment	The procedure is initiated by the SMF towards the UPF. This procedure is session related and sets up a PFCP session between the SMF and the UPF to configure rules in the UPF so that the UPF can manage incoming packets accordingly.
PFCP Session Modification	The procedure is initiated by the SMF towards the UPF. This procedure is session related and enables the SMF to modify the existing PFCP session with new rules or alter existing rules to process the packet.
PFCP Session Deletion	The procedure is initiated by the SMF towards the UPF. This procedure is session related and deletes the existing PFCP session.
PFCP Session Report	The procedure is between the SMF and the UPF and it is initiated by the UPF. This procedure is session related and used by the UPF to report information related to the PFCP session to SMF.

6.1.2.3 PFCP Counters

The counters related to N4 reference point can be collected from the SMF and UPF. The messages processed by the SMF are captured in the simulations and the relevant metrics are stored to understand the network condition related to the nodes and sessions. The N4 reference point counters can be divided into four categories:

- PFCP Request Messages
- PFCP response Messages
- PFCP Error Messages
- PDU Session Messages

Table 6-5 denotes the list of recommended counters for the N4 reference point, to be stored and utilized by an AI system. The counters are associated with messages received, sent, and rejected based on the response and request of the procedure.

Table 6-5: PFCP Counters - N4 Reference point

Category	Recommended counters
PFCP Request Messages	• PFCP Establishment message sent. • PFCP Modify message sent. • PFCP Delete message sent. • PFCP Report Failure message sent. • PFCP Association Setup Message sent. • PFCP Association Setup Message received. • PFCP Association update Message sent. • PFCP Association update Message received. • PFCP Association release Message sent. • PFCP Node Report Message received. • PFCP Heartbeat requests sent. • PFCP Heartbeat requests received.
PFCP response Messages	• Successful PFCP Establishment Response received. • Unsuccessful PFCP Establishment Response. • Successful PFCP Modify Response received. • Unsuccessful PFCP Modify Response received. • PFCP Report Response sent. • Successful PFCP Association Setup Response sent. • Unsuccessful PFCP Association Setup Response sent. • Successful PFCP Association Setup Response received. • Failed PFCP Association Setup Response received. • Successful PFCP Association update Response sent. • Unsuccessful PFCP Association update Response sent. • Successful PFCP Association update Response received. • Failed PFCP Association update Response received. • Successful PFCP Association release Response received. • Failed PFCP Association release Response received. • PFCP Heartbeat response sent. • PFCP Heartbeat response received.
PFCP Error Messages	• Unsupported PFCP version • Unknown Message Type value • Unexpected request message • Mandatory IEs are missing
PDU Session Messages	• PDU Session Establishment procedure attempts. • Successful PDU session establishment procedures. • Failed PDU session establishment procedures.

	• PDU Session Release procedure attempts. • Successful PDU session Release procedures. • Failed PDU session Release procedures.

6.1.3 HTTP/2 Protocol

HTTP/2 supports all the core features of predecessor HTTP releases such as TCP connection with multiple request and responses, authentication mechanism, caching, and status code. Additionally, HTTP/2 provides an optimized transport for HTTP semantics by introducing the advanced features such as:

a. Multiplexing: Multiplexing of requests by having each HTTP request/response exchange associated with its own stream (RFC7540, 2015).
b. Flow control and prioritization: flow control and prioritization enable the use of multiplexed stream efficiently (RFC7540, 2015).
c. Server Push: Server can push responses to a client which server anticipates the client will need, this approach will utilize the network but it will result in reduce the potential latency (RFC7540, 2015).
d. Header compression: Header compression allows connection can contain large amounts of redundant data, frames in compressed format hence allowing many requests to be compressed into one packet. (RFC7540, 2015).

The 5G core network is designed based on the service-based architecture and adopts the HTTP/2 as the application layer protocol (ETSI.129.500, 2018). HTTP/2 is used at the control plane for communication between the 5G core network functions. Consequently, the AMF then oversees the connection and mobility management tasks, while all messages related to session management are forwarded to the SMF over the N11 interface via HTTP/2.

6.1.3.1 HTTP/2 Protocol Stack

The HTTP/2 is an application layer protocol, and it is constituted with other communication layers such as physical, IP, TCP, and TLS to provide the full functionality as shown in Figure 6-5. Figure 6-5 represents the HTTP/2 protocol stack to enable 5G network functions to communicate within each other.

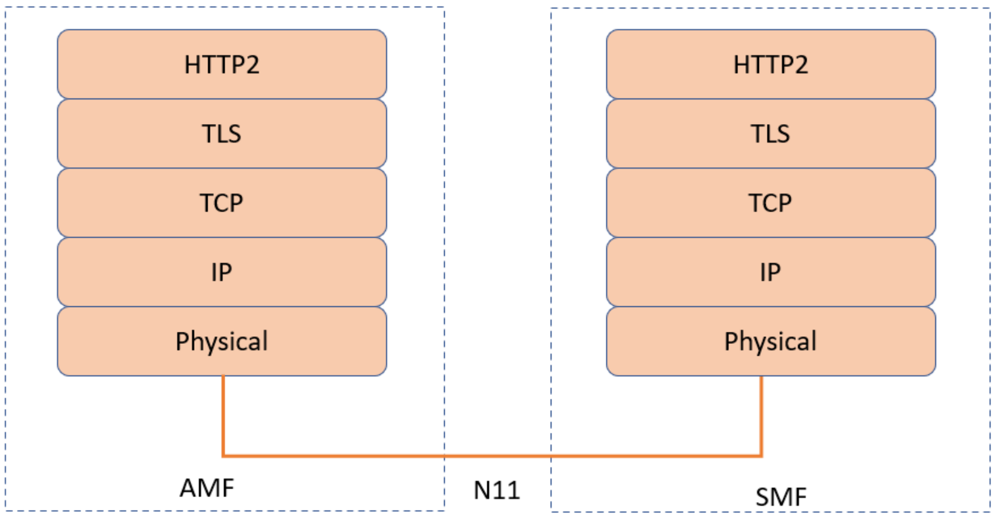


Figure 6-5: HTTP/2 Protocol Stack

The physical layer provides the physical connectivity between the network functions, the stack below shows the physical connectivity between the SMF and AMF with reference point N11. The IP protocol provides the IP source and destination address for the communication between the two network functions within the 5G core network. The TCP protocol as transport layer is responsible for the end-to-end conversion between the source and destination, such as AMF and SMF in Figure 6-5, and this communication is connection-oriented to ensure reliability. The end-to-end communication between the NFs can be encrypted via Transport Layer Security (TLS) on top of the TCP protocol. Encryption can be implemented depending on the location of the network function. For the sake of simplicity, the 5G network functions deployed for this thesis in the university lab do not implement encryption of HTTP/2 protocol, which shall not alter the outcome of the data collection and monitoring solution design. The HTTP/2 operates at the application layer and enables the communication between the 5G network functions by implementing the standard headers for HTTP requests and responses, as defined in (ETSI.129.244, 2020).

6.1.3.2 HTTP/2 Methods

HTTP/2 uses the set of methods defined in HTTP/1. The methods supported by HTTP are list in Table 6-6. These methods are being utilized by 5G core network functions to establish end to end communication.

Table 6-6: HTTP/2 Methods (MOZILLA, 2023)

Methods	Description
GET	GET method allows the retrieval of the data from the target.
PUT	PUT method replaces the content at target with the request payload.
PATCH	PATCH method allows the partial modification of the content at target.
POST	POST method allows the change at the
HEAD	HEAD method is like the GET but without the response body
OPTIONS	OPTIONS method describes the communication options.
CONNECT	CONNECT method enable the TLS communication by establishing tunnel
DELETE	DELETE method deletes the specified resources from the target

6.1.3.3 HTTP/2 Counters

The counters related to N11 reference point are service related and originate from the SMF. The messages processed by the SMF from AMF are then captured in the simulations and the relevant metrics are stored to facilitate the understanding of the network condition. N11 reference point counters can be as follows:

- Namf – EBI (EPS bearer ID) Assignment Service
- Namf - N1N2 Transfer Service
- Namf - N1N2 Transfer Failure Notification Service

Table 6-7 proposes the list of recommended counters for the N11 reference point to be stored and utilized by an AI system. The counters are associated with service messages being received, sent, and rejected based on the consumer and provider procedures.

Table 6-7: HTTP/2 Counter - N11 reference point

Category	Recommended counters
Namf – EBI (EPS bearer ID) Assignment Service	• Communication EBI Assignment service operation sent by SMF. • Unsuccessful responses received by SMF for the Communication EBI Assignment service. • EBI Assignment 403 responses received. • Rejects received by SMF for the Communication EBI Assignment service.
Namf - N1N2 Transfer Service	• Communication N1N2 Message Transfer service operation sent by SMF. • Successful responses received by SMF for the Communication N1N2 Message Transfer service operation. • Rejects received by SMF for the for the Communication N1N2 Message Transfer service.
Namf - N1N2 Transfer Failure Notification Service	• Communication N1N2 Message Transfer failure notification received by SMF. • Unsuccessful responses received by SMF for the Communication N1N2 Message Transfer service operation. • Rejects received by SMF for the Communication N1, N2 Message Transfer service

6.1.4 End to End Protocol Stack

The User Equipment (UE) needs to be registered into the 5G core network to start using provisioned 5G services. The UE registration process is performed on the control plane of the 5G core network, but it also involves the communication towards the user plane (UPF) to complete the PDU establishment procedure. Communication from the UE to the 5G core network involves multiple network functions and multiple protocols. Figure 6-6 shows the complete control plane protocol stack along with the network function, protocol stack supported by the network function and the relevant reference point as defined by 3GPP (3GPP-TS.23.501, 2022).

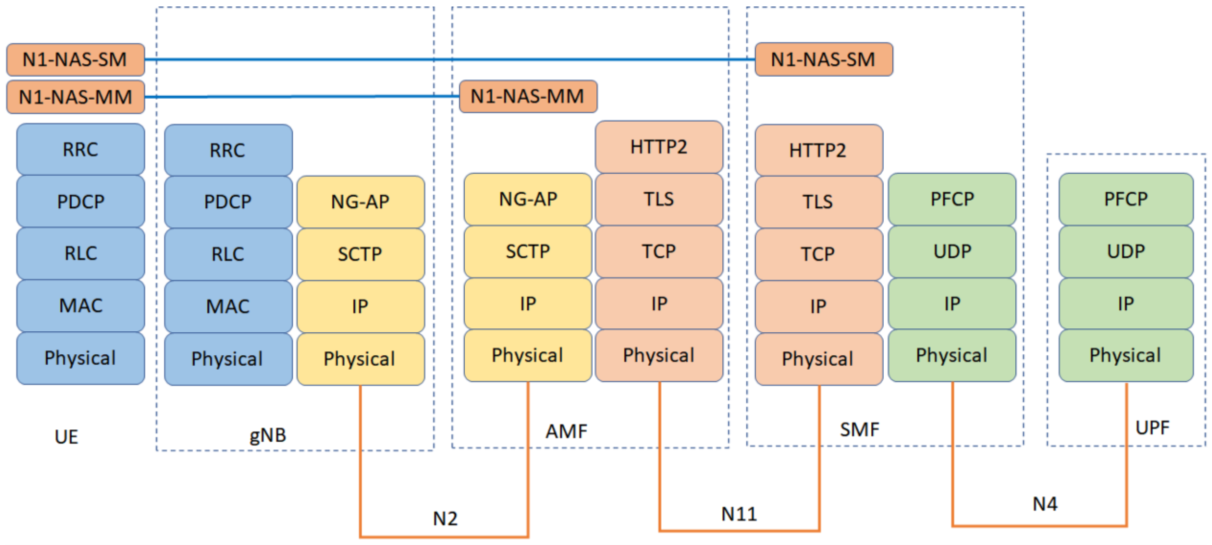


Figure 6-6: Control Plane Protocol Stack

As shown in Figure 6-6, UE and gNB protocol stack comprises of physical layer (PHY) to provide error detection, encoding and decoding, rate matching, mapping of the coded transport channel onto physical channels, power weighting, modulation, frequency and time synchronization, frequency and time synchronization, radio characteristics measurements, digital and analog beamforming, Radio Frequency (RF) and Multiple Input Multiple Output (MIMO) antenna processing. The Media Access Control (MAC) layer operates at Layer-2 to provide function such as beam management, random access procedure, mapping between logical and transport channels, scheduling information reporting, error detection, priority handing, transport format selection, padding and concatenation of multiple MAC. The Radio Link Control (RLC) provides layer-2 functionalities such as error correction, reordering of 5G-RLC data PDUs, duplicate dedication, protocol error handling, segmentation, re-segmentation and 5G-RLC re-establishment. The Packet Data Convergence protocol provides the services to RRC and user plane such as transfer of control plane and user plane data, ciphering, compression, and integrity protection. The RRC operate at layer 3 and provides functions such as transfer of upper layer PDUs, error detection, duplicate detection, protocol error detection, segmentation, re-segmentation, broadcasting of system information to Non-Access Stratum (NAS) or Access Stratum, Security, establishment, maintenance & release of Radio Resource Control (RRC) connection, mobility functions, UE measurements reporting, control of UE reporting, UE based mobility and NAS direct message transfer (RF-Wireless, 2023). The other protocol stack along with the reference points shown in Figure 6-6 has been explained earlier in this chapter.

6.2 Data Capture in 5G Networks

The telecommunication network provides communication services to the consumer with single click of button but the network behind to provide these services are built on complex infrastructure, multiple technologies, and applications. Technology is evolving hence making the network more complicated with the passage of time and increases the challenge for operators to monitor network with different technologies at the same time. The telecommunication networks are divided into Radio Access Network (RAN) and core network to have separation between the access and core. The RAN network provides connectivity and core network provide the provisioned services to the consumer. Additionally, network complexity increases with the network expansion and having the various technologies being operational at the same time such 2G, 3G, 4G and 5G makes the network difficult to manage and monitor. These various network technologies working together introduces the complication such as handover of UE to move in/out of one technology into another hence putting more pressure on the operators to monitor Key Indicator Performance (KPI) to deliver better performance to the consumer.

The networks need to be monitored continuously for the below main reasons. (UTEL, 2023)

- Visibility: The visibility of the full network can be achieved by real-time monitoring.
- Performance: Measurement of performance of the network against the baselines KPI. This ensures the quality of the network for the consumer.
- Improved security: The fraud activities are growing exponentially hence real time monitoring of the network helps in combatting bad actors and provides secure network to the consumers.
- Planning: The data traffic is increasing aggressively. The minoring of the traffic helps proper expansion of the network within the required time plan.
- Alerting and Notification: The real time monitoring of the network enables the operator to avoid downtime by implementing the alerting mechanism in case of any failure.

Monitoring of the networks can only be achieved by collecting the data from the various network elements. This sub chapter focuses on capturing the data from the 5G network functions deployed on cloud native and highlights the challenges related to legacy capturing procedures implemented for applications deployed on physical server and virtual machines.

6.2.1 Legacy Data Capturing

The telecommunication core network applications are deployed on the physical server and virtual machine hosted on the physical servers. These servers are deployed inside the big datacenters inside racks and each rack is equipped with Top of the Rack (ToR) switch. The ToR from each rack is connected to the datacenters core switches to provide the connectivity to application deployed within the rack with the rest of the network. All the traffic traverses through the top of the rack switch even if the communication is happening between the application within the rack or application needs to communicate with another application residing inside different rack at different location.

The legacy capturing has been shown in Figure 6-7, where monitoring probs are implemented at the switches.

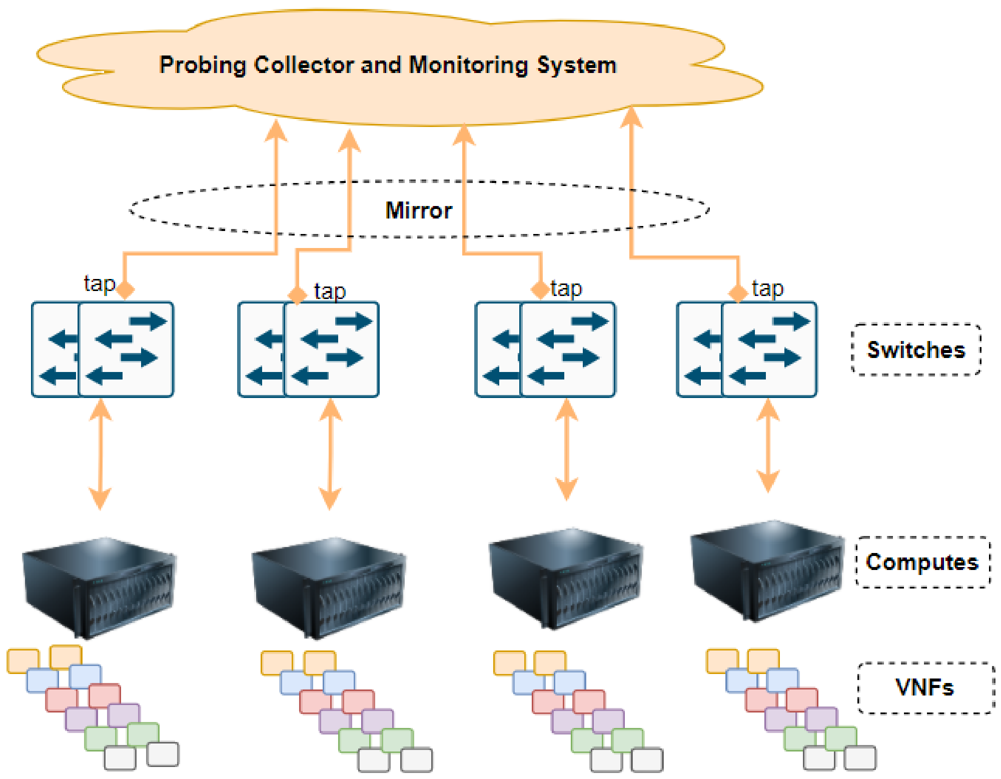


Figure 6-7: Legacy Probing

The computes nodes are running the workloads as Virtual Network Functions (VNFs) and each VNFs can be deployed with different application. The VNFs send all the traffic to the switches to achieve communication between the VNFs or outside the VNFs at different locations. In such cases, switches are probed to capture all the traffic going in and out of the application by implementing the port mirroring at switch level. The mirroring procedure will copy all the traffic to the probe collector system to achieve network monitoring. Switches and routers from different provider has the port mirroring functionality by default hence this seems to be most obvious way to implement the data capturing without any complexity (Cisco, 2023b).

6.2.2 Cloud Native Data Capturing

The 5G core network supports deployment on cloud native environment hence 5G core can be deployed on the Kubernetes platform as container. As explained in chapter 2.5, the containers network is different compared to the VM and physical servers. Containers running on the same worker nodes will communicate with each other freely without going to the top of rack switch, this reduces the latency and results in better performance, but it introduces the challenged for the legacy data capturing implementation. The traffic does not leave the worker nodes to top of rack switch hence all the traffic cannot be captured by probing the switches deployed inside the rack. This requires some new probing solution or change in existing probes to be placed at different location in the network instead of the top of rack switch. The thesis proposes two different probing solutions to solve this problem.

- Virtual tap at pod

- Capturing at worker interface

6.2.2.1 Capturing at worker node

The worker nodes are hosting all the pods hence all the communication happening between the pods residing on the same worker nodes will be visible at the worker network interface. Prob can be implemented on the network interface to mirror all the traffic to the probing collector so that data capturing all the network/application communication can be forwarded to the monitoring system. Figure 6-8 shows the implementation of the probing at the network interface of the worker node, the worker nodes interface can see all the communication between the pods hence control plane traffic is mirrored to the probing collector.

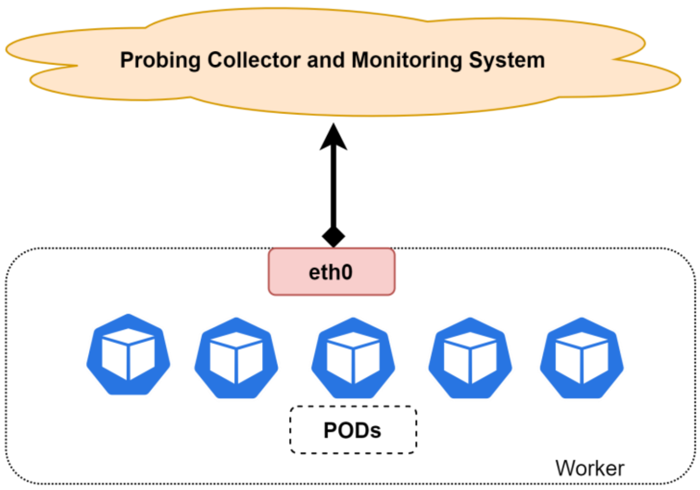


Figure 6-8: Capture at worker node

Tshark is an open-source network packet analyzer. It has libraries to capture the data on the specific interface of the worker nodes, capture the precise protocols on specific IP and port (TSHARK, 2023). In the lab, Tshark version 4.0.3 has been deployed to capture the pod communication on the worker nodes. The captured files are stored on the worker nodes for further processing.

```
root@baston:~# tshark -version
TShark (Wireshark) 4.0.3 (Git v4.0.3 packaged as 4.0.3-1~ubuntu20.04.0~ppa1).
```

6.2.2.2 Virtual tap at pod

The pods are deployed with the default Container Network Interface (CNI), default CNI is used to communicate with other pods. In case pod must send the traffic to outside network then it uses the worker IP to send that traffic to the destination node. The concept can be utilized to send the traffic the from the pod to the probing collector IP directly including all the necessary information related to the control plane traffic. This provides the flexibility to enable and disable the tapping at the interface or protocol level and it has very minimal dependency at the infrastructure as mirroring is enabled at application level. The tapping implementation on the network function level requires code change at application so that control plane traffic processed by application is mirrored to the probing collector directly. Figure 6-9 shows the high-level implementation of this solution where all the pods on the worker node are mirroring the control plane traffic to the probing collector.

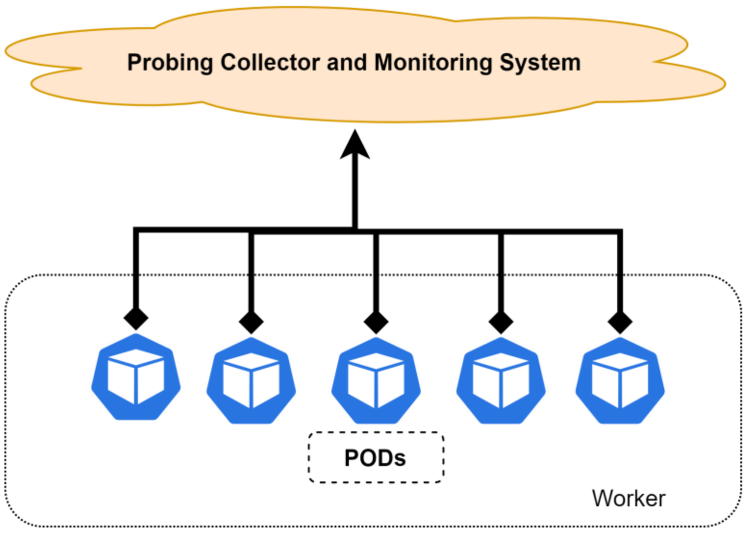


Figure 6-9: virtual tap at POD

In the university lab for this thesis, a solution with capturing at worker nodes has been implemented to capture the data. The capture data is being processed further so that it can consumed by machine learning system, and this has been covered in the next seventh chapter.

7 DATA COLLECTION AND PROCESSING IN 5G NETWORKS

The 5G core network has various network function to provide the functionalities and services as defined in 3GPP specification (3GPP-TS.23.501, 2022). Each network function supports different protocol to communicate with other network function and 5G NG-RAN, there have been defined reference points such as N2 for gNB - AMF, N11 for AMF – SMF, N4 for SMF – UPF and others. The messages processed by network functions in the 5G core network are related to the registration, policy enforcement, PDU establishment, charging and many more messages defined in the 3GPP specification to provide the designated services and functionalities (3GPP-TS.23.501, 2022). The data collection and processing become vital from each of the network functions to have visibility on the actual traffic flowing through the network function. Furthermore, it helps to monitor the network in real time against the defined Key Performance Indicator (KPI), detection of the any fault or failure, scaling of network with traffic and protecting the network against hackers.

The previous chapter of thesis has explained various data capturing techniques for all the interfaces & protocols along with implementation at the network functions and worker level. This chapter will be focusing on the collection of the data capture from the previous chapter from 5G core network and then, further processing of the data to make it available for machine learning system in real time. It is mandatory to have the data available to the machine learning system in real time as this will help the Artificial Intelligence (AI) system to take the proper decision on network related to the scale out/in and protection against the ongoing attack on the network.

As explained in earlier chapters, 5G core network is designed on the principle of scalability, high throughput, low latency, fault tolerance, reliability, and durability. Secondly, the monitoring and machine learning system requires the data to be present in real time to perform the action quickly on the network. Tool selection becomes important so that data processing can happen from 5G core network at real time. Hence, Kafka is used in this thesis to allow the real time streaming of the data and support 5G core network principles in general.

7.1 Solution Overview

The data capturing and processing solution has been deployed in the lab facility of Oslo Metropolitan University. The goal of the solution is to collect the data from the network function on specific interfaces/protocols, process the data and then make the clean data available to machine learning system to be consumed in real time. The overall solution has been divided into the three main blocks for simplicity as shown in Figure 7-1.

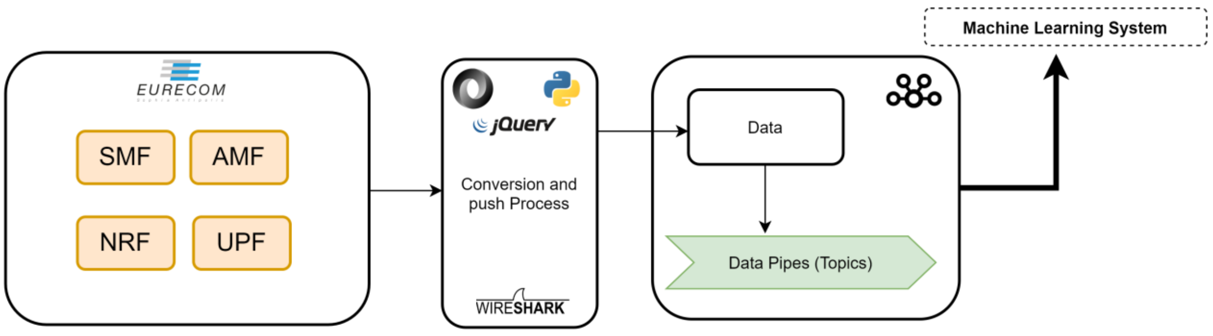


Figure 7-1: Solution Overview - Data Collection and Processing

The first block of the solution is to have functional 5G core network deployed in lab facility of Oslo Metropolitan University. To accomplish the first block, OpenAirInterface5G (OAI) has been deployed and operational as explained in fifth chapter. The OAI 5G core is deployed with other network functions such as UDM, UD and AUSF but only few network functions are being shown in Figure 7-1. The thesis focuses on the data collection and processing from SMF, AMF and UPF on specific reference points. The same process of data capturing, collection and processing can be applied to the other network function as well.

The second main block of the solution focuses on the data capturing, separating the data with respect to the 5G reference points such as N2, N4 and N11/HTTP2. The separation of the data with respect to the reference points ensures that machine learning systems can load the data instantly without implementing further parsing algorithm related to data separation. Furthermore, captured separated reference point data is converted into JSON format (JSON, 2023) with tshark and then cleaned up with jQuery (JQUERY, 2023). Finally, Kafka producer written in python (PYTHON, 2023) is used to push the data to Kafka server.

The final module of the solution comprises of the Kafka server (KAFKA, 2023) setup in the lab facility of Oslo Metropolitan University. The purpose of the Kafka server is to make the collected and separated data available for the machine learning system to be consumed in real time. The data is loaded to the Kafka server by the Kafka producer written in the python for each reference point.

7.1.1 Processing Raw Data

The network functions are residing on the worker nodes and all the traffic associated to the network functions passes through the Network Interface Card (NIC) of the worker nodes. The data captured on the worker node is raw as it captures all the packets passing through the worker NIC. The separation and cleaning process of the data has been divided into three sub processes P-1, P-2 and P-3 as shown in Figure 7-2. The purpose of each sub-process is to perform a certain action on the raw data and then forward the data to the next sub-process for further processing.

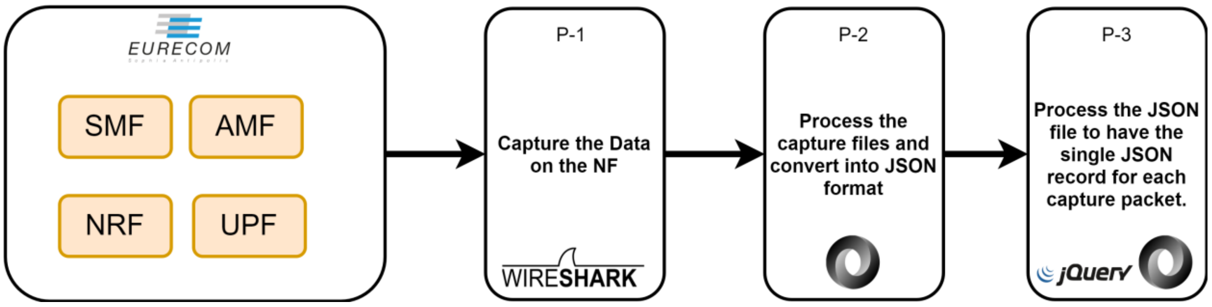

EURECOM
SMF
AMF
NRF
UPF
P-1
Capture the Data on the NF
WIRESHARK
P-2
Process the capture files and convert into JSON format
P-3
Process the JSON file to have the single JSON record for each capture packet.
jQuery

Figure 7-2: Raw Data Separation Process

The actions performed by each sub-process has been detailed as below:

- **P-1**: The data is captured on the worker nodes/network functions via tshark utility using specific filter related to the protocols. NGAP filter has been used to separate the packets relevant to the N2 reference point, PFCP filter has been deployed to capture packets related to the N4 reference point and HTTP2 filter is used for N11 packets between AMF and SMF (this filter might capture HTTP packets from other network functions, but this can be separated by applying the filters on the URI within the HTTP messages). The captured data is then stored in a separate file with timestamp included in the name of the file. The P-1 sub-process then forwards the data to P-2 for further processing.
- **P-2**: This sub-process will receive three captured files for each reference point such as N2(NGAP), N4(PFCP) and N11(HTTP/2). The sub-process will convert these files to JSON format via tshark utility hence this will produce three JSON files and forward the files to the next sub-process for further processing.

```
tshark -r <input>.pcap -T json > <output>.json
```

- **P-3**: P-3 sub-process will receive three JSON files from the P-2 sub-process. Ideally, these files should be ready to be processed further but received files by P-2 sub-process does not represent the single message per line as shown below and complete message can be check in Appendix - 1K

```
"pfcp": {
  "pfcp.flags": "0x21",
  "pfcp.flags_tree": {
    "pfcp.version": "1",
    "pfcp.spare_b4": "0",
    "pfcp.spare_b3": "0",
    "pfcp.fo_flag": "0",
    "pfcp.mp_flag": "0",
    "pfcp.s": "1"
  },
  "pfcp.msg_type": "56",
  "pfcp.length": "18",
  "pfcp.seid": "0x7248934023107441",
  "pfcp.seqno": "44534",
  "pfcp.spare_oct": "0",
  "Report Type : ": {
    "pfcp.ie_type": "39",
    "pfcp.ie_len": "2",
    "pfcp.spare_b7": "1",
    "pfcp.report_type.uisr": "0",
    "pfcp.report_type.sesr": "0",
    "pfcp.report_type.tmir": "0",
    "pfcp.report_type.upir": "0",
    "pfcp.report_type.erir": "0",
    "pfcp.report_type.usar": "0",
    "pfcp.report_type.dldr": "0",
     "IE data not decoded by WS yet": {
       "_ws.expert": {
         "pfcp.ie_data_not_decoded": "",
         "_ws.expert.message": "IE data not decoded by WS yet",
         "_ws.expert.severity": "4194304",
         "_ws.expert.group": "83886080"
       }
     }
   }
 }
```

P-3 sub process will convert the messages inside the received file from P-2 into single JSON record via jQuery and then file is ready to be used for further processing and loading into the Kafka.

```
jq -c .[] <output>.json > <clean>.json
```

jQuery version “jq-1.6” is deployed in the lab to perform the conversion.

```
root@worker2:~# jq --version
jq-1.6
```

7.1.2 Loading Data into KAFKA

The files produced from the previous processing modules contain a single JSON record for each message processed by the network functions in the 5G core network. The files are clean and ready to be processed for further loading onto the Kafka server. Data loading into Kafka is achieved by following steps:

- Set up Kafka broker and manager along with the zookeeper.
- Setup topics for each reference point.
- Setup Kafka producer to load data into Kafka topics.

7.1.2.1 KAFKA broker and manager setup

The Kafka broker and manager has been deployed in the lab facility of Oslo Metropolitan University. Separate worker node with Ubuntu operating system “20.04 TLS” has been setup along with the latest version of docker and docker compose in the lab environment. Kafka is installed using the docker compose file in Appendix - 1L, the compose file has been provided with the IP of the worker node on which Kafka broker will run as container along with the port number “9092”. Similarly, Kafka manager has been deployed as container on port “9000” as highlighted in below configuration snapshot.

```
  kafka:
    image: wurstmeister/kafka
    container_name: kafka
    ports:
      - 9092:9092
    environment:
      KAFKA_ADVERTISED_HOST_NAME: 10.0.71.29
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181

  kafka_manager:
    image: hlebalbau/kafka-manager:stable
    container_name: kakfa-manager
    restart: always
    ports:
      - "9000:9000"
    depends_on:
      - zookeeper
      - kafka
    environment:
      ZK_HOSTS: "zookeeper:2181"
      APPLICATION_SECRET: "random-secret"
    command: -Dpidfile.path=/dev/null
```

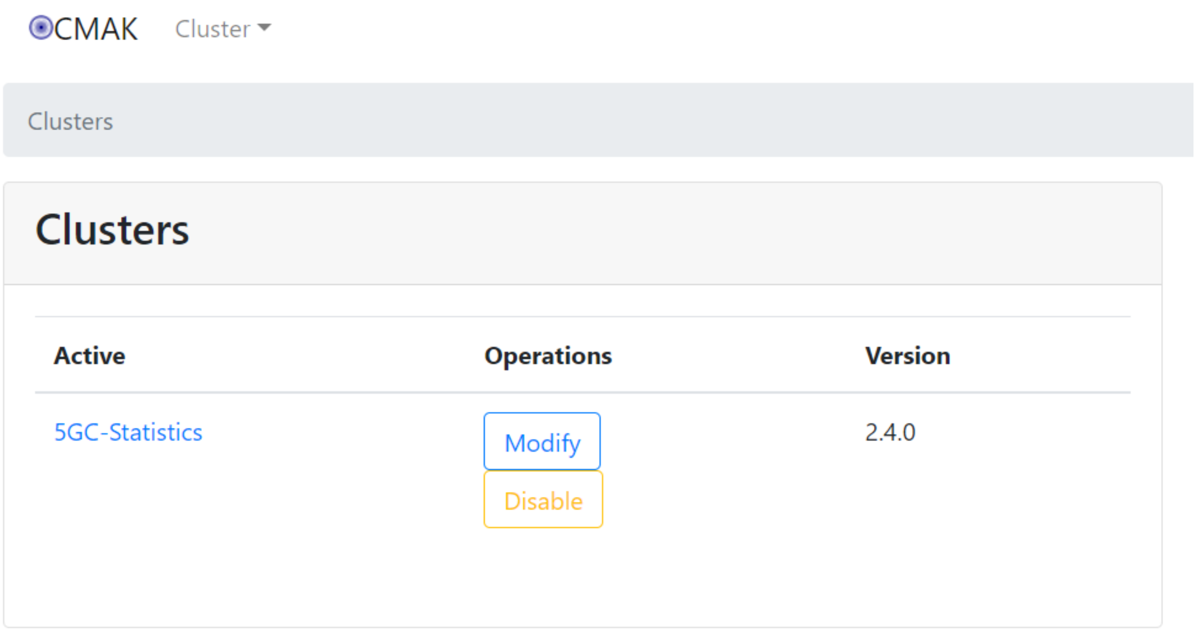
The docker compose file has been deployed using the below command to run in the background on the worker node.

```
docker-compose -f kafka-docker-compose.yaml up &
```

The installation of Kafka broker and manager is verified from the command line as shown below. The Kafka broker should be listening on port “9092” and Kafka manager should be accepting the connection on port “9000”.

```
root@worker2:~# docker ps | grep -i Kafka
c101aa4db46d   hlebalbau/kafka-manager:stable                         "/kafka-manager/bin/…"
6 days ago   Up 6 days   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp
kakfa-manager
4928e7fe9ae4   wurstmeister/kafka                                     "start-kafka.sh"
6 days ago   Up 6 days   0.0.0.0:9092->9092/tcp, :::9092->9092/tcp
kafka
root@worker2:~#
```

When Kafka manager is up and running then it is possible to access the GUI of the Kafka server on URL http://worker_IP: 9000 to manage the Kafka broker. The Kafka manager is used to create the Kafka cluster name “5G-Statistics” with Kafka version “2.4.0” as shown in Figure 7-5.



Clusters

Active	Operations	Version
5GC-Statistics	Modify Disable	2.4.0

Figure 7-3: Kafka Cluster

The Kafka cluster is ready with a single broker now and it is possible to create the required Kafka topics for this experimentation project.

7.1.2.2 KAFKA pipelines/topics setup

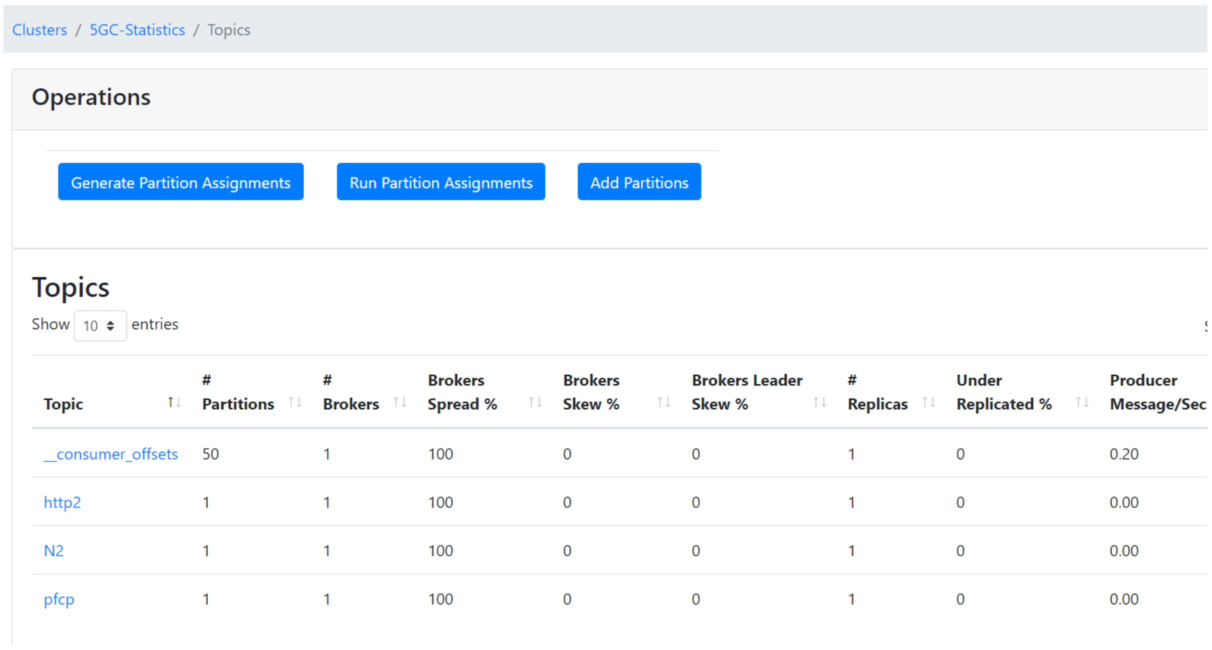
The Kafka topics are responsible for storing the data pushed by producers and it is possible to create assorted topics for various type of data being pushed from the producer. In previous section 7.1.2.1, the Kafka cluster is being setup with name “5G-Statistics”and the next steps is to add the Kafka topics to the Kafka cluster to enable the cluster to start storing data into the partitions. It is possible to add the Kafka topics via Kafka command line utility. In total, three Kafka topics are added to the Kafka cluster with the name “http2”, “N2” and “pfcp”. The topics are created with single partition and single replication factor as the resoruces are limited in the lab hence these topics are deployed with minimum configuration.

```
@kafka-client:$ kafka-topics.sh --create --replication-factor 1 --partitions 1 --topic N2 -
-bootstrap-server 10.0.71.29:9092
Created topic N2.

@kafka-client:$ kafka-topics.sh --create --replication-factor 1 --partitions 1 --topic pfcp
--bootstrap-server 10.0.71.29:9092
Created topic pfcp.

@kafka-client:$ kafka-topics.sh --create --replication-factor 1 --partitions 1 --topic
http2 --bootstrap-server 10.0.71.29:9092
Created topic http2.
```

The next step is to verify that topics are created successfully, and it is visible inside Kafka cluster. The verification can be performed by listing topics in the Kafka cluster via "list" command as below:

```
@kafka-client: $ kafka-topics.sh --list --bootstrap-server 10.0.71.29:9092
http2
N2
Pfcp
```

Similarly, it is possible to verify the topics from Kafka manager GUI, GUI will show the created topics under "5G-statistics" cluster as shown in Figure 7-4.


Clusters / 5GC-Statistics / Topics
Operations
Generate Partition Assignments
Run Partition Assignments
Add Partitions
Topics
Show 10 entries

Topic	# Partitions	# Brokers	Brokers Spread %	Brokers Skew %	Brokers Leader Skew %	# Replicas	Under Replicated %	Producer Message/Sec
__consumer_offsets	50	1	100	0	0	1	0	0.20
http2	1	1	100	0	0	1	0	0.00
N2	1	1	100	0	0	1	0	0.00
pfcp	1	1	100	0	0	1	0	0.00

Figure 7-4: Kafka Topics

7.1.2.3 KAFKA producer setup

The Kafka producers are responsible for pushing the data towards the specific topic within the Kafka cluster. The setup requires three producers to push the data towards the three Kafka topic independently. The Kafka producer is created in python by importing the Kafka Producer library along with the JSON. There are three independent python scripts to push data into Kafka three Kafka topic. Each python script opens the clean file created in section 7.1.1 ("pyhon.json" file is opened for PFCP messages) and then program starts reading the file line by line. Each line is JSON record of the actual messages processed by network function and it is pushed to the Kafka topic. In case of failure, an error message is printed, and it is possible to store the message into the text for verification. The complete code of python program is as follows for the pfcp messages:

```
from kafka import KafkaProducer
import json

def on_send_success(record_metadata):
    print(record_metadata.topic)
    print(record_metadata.partition)
    print(record_metadata.offset)

def on_send_error(excp):
    log.error('I am an errback', exc_info=excp)

def json_serializer(data):
    return json.dumps(data).encode("utf-8")

producer = KafkaProducer(bootstrap_servers=['10.0.71.29:9092'],
value_serializer=json_serializer)

if __name__ == "__main__":

    pfcp_file = open('pfcp.json', 'r')
    Lines = pfcp_file.readlines()
    for line in Lines:
        push_json= line.strip()
        record = json.loads(push_json)
        producer.send("pfcp",
record).add_callback(on_send_success).add_errback(on_send_error)
    producer.flush()
    pfcp_file.close()
```

The python code to push N2, N4, and N11 messages in to the Kafka topics can be viewed from Appendix - 1N, 1O & 1P.

7.1.2.4 *Data Loading into KAFKA*

The setup had three Kafka producers to push the JSON records to three Kafka topics – N2, http2 and pfcp. The high-level flow of message loading into Kafka topics is shown below in Figure 7-5.

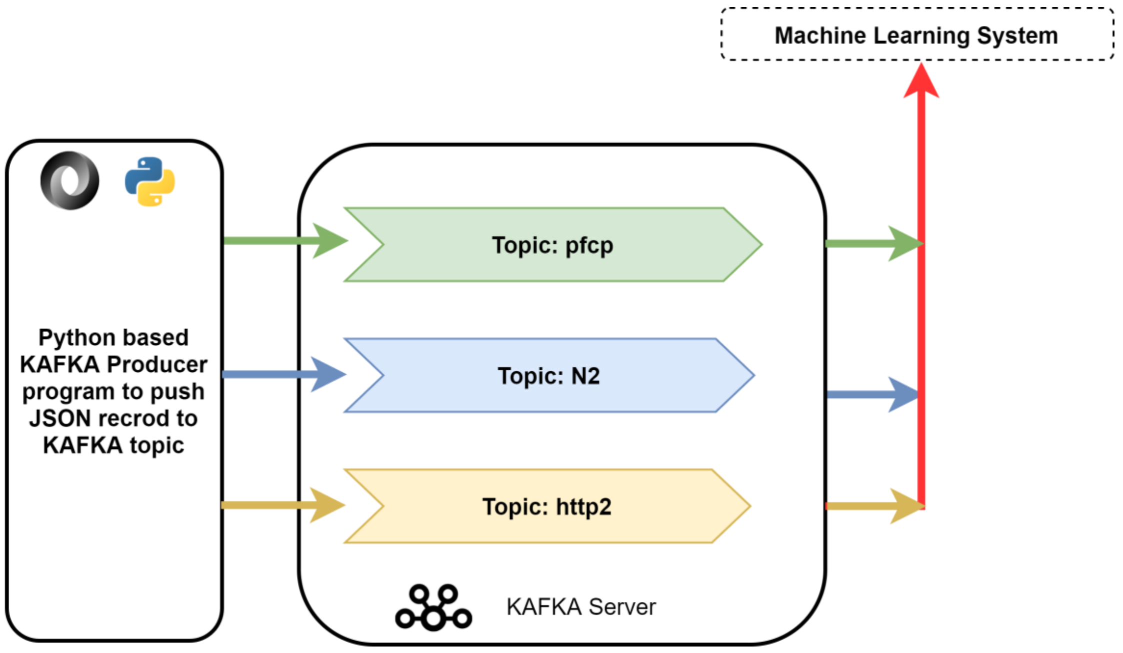

Machine Learning System
Python based KAFKA Producer program to push JSON recrod to KAFKA topic
Topic: pfcp
Topic: N2
Topic: http2
KAFKA Server

Figure 7-5: Data loading into Kafka

As presented in Figure 7-5, PFCP messages are pushed to the pfcp topic, N2 reference messages are pushed to N2 topic of Kafka and N11 messages are published into the http2 topic. These messages from each producer are published into the partition associated with the topics. The messages are published in real time without any latency by python program and separation of the messages is achieved by having separate Kafka topics.

The messages pushed to the Kafka broker are ready to be consumed by the Kafka consumer. The machine learning system should implement the Kafka consumer and it should inform the Kafka broker indicating the partitions it wants to consume. The consumer specifies the offset in the request towards the Kafka broker and accordingly, it will receive a log chunk from the offset position. Machine learning system has control over the offset hence it can request for the same data multiple times if required. Kafka broker will not delete the data after the consumption from consumer, data will be removed by Kafka broker after it reaches the configured retention period.

7.1.3 Loading Data into Elastic Stack

The data available in the Kafka broker allows any Kafka consumer to ingest the data from any partition. The Machine Learning (ML) System will ingest the data from the Kafka topics, but an additional setup has been created in the University lab to run the Elastic stack (ELASTIC, 2023). An elastic stack is a group of open-source products from Elastic, and it allows the processing, filtering, storing, and visualization of data in real-time from various sources. The stack comprises the three components such as Elastic search, Logstash, and Kibana, and this is also known as ELK. All the products used in ELK are developed, managed, and maintained by Elastic. In ELK, Elasticsearch provides a full-text search and analysis engine. Logstash is a flexible log parsing engine that takes the data from an input source and applies different transformations and enhancements before shipping it to various destinations. Finally, Kibana in ELK provides the visualization layer on top of Elasticsearch, and this gives the users ability to analyze and visualize the data in real time.

Figure 7-6 shows the high-level data loading flow into the Elastic stack along with the machine learning system.

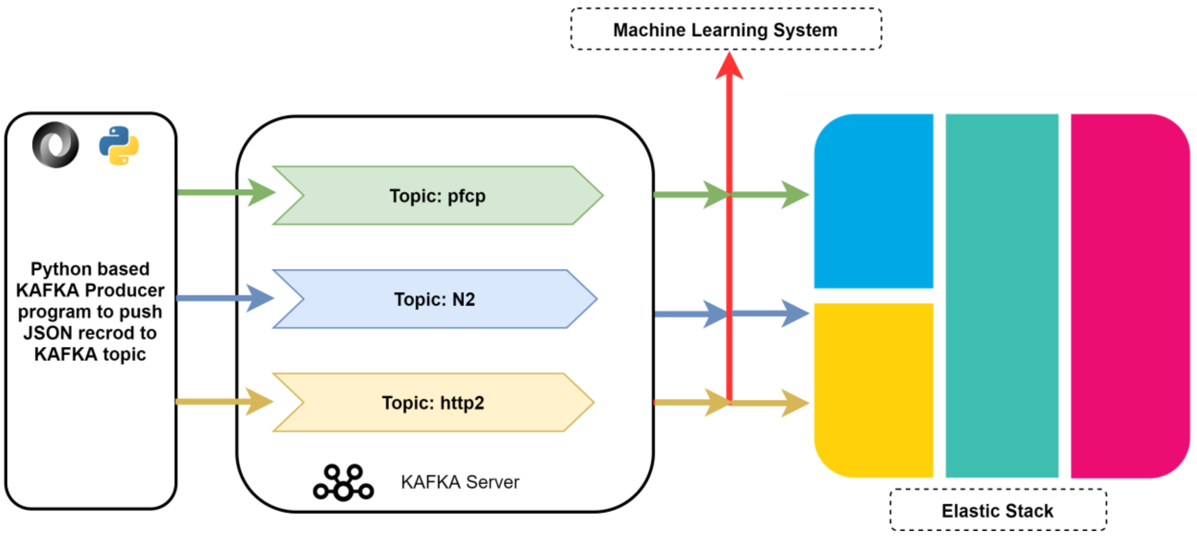


Figure 7-6: Data Loading into ELK

The purpose is to create an environment in parallel to the ML system so that data can be searched and analyzed. The Logstash has Kafka consumer plugin to ingest the data from each Kafka topic. The ingested data will be processed and loaded to the elastic search after implementing the required transformations and enhancements. As the data reaches elastic search then it is indexed so that user can login to the Kibana for searching and viewing the processed data. It should be possible to create the dashboard on Kibana for further data visualization.

7.1.3.1 ELK Setup

The ELK setup includes setting up Elastic Search, Logstash, and Kibana. A dedicated virtual machine in the university lab environment has been created with the Ubuntu operating system deployed along with the docker. This machine will host the elastic search, Kibana, and Logstash installation. Elastic search version “7.4.0” is installed, and it is configured to listen on port “9200”. The complete compose file of the elastic search installation can be viewed in Appendix - 1M. When the docker-compose is executed, the elastic search container is running and listening on port 9200, as shown below.Elastic Search & Kibana – Docker Compose file

```
root@worker2:~# docker ps | grep elasticsearch
4c698cb64c46   docker.elastic.co/elasticsearch/elasticsearch:7.4.0   "/usr/local/bin/dock…"
9 days ago   Up 9 days   0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9300/tcp,
:::9300->9300/tcp   elasticsearch
```

Similarly, Kibana is deployed as a container via docker-compose on the same worker node. Kibana version “7.4.0” is deployed as part of this installation, and it is configured to listen on the default port “5601”. The compose file for Kibana can be viewed in Appendix - 1M. The docker-compose also deploys Kibana as part of the installation, and the installation can be verified via the command line as the Kibana container should be running and listening on port “5601”.

```
root@worker2:~# docker ps | grep -e kibana
41c672bad325   docker.elastic.co/kibana/kibana:7.4.0                 "/usr/local/bin/dumb…"
9 days ago   Up 9 days   0.0.0.0:5601->5601/tcp, :::5601->5601/tcp
kibana
```

Elastic search and Kibana are running, and it is possible to access Kibana Graphical User Interface (GUI) via worker IP and port 5601 as shown in Figure 7-7.

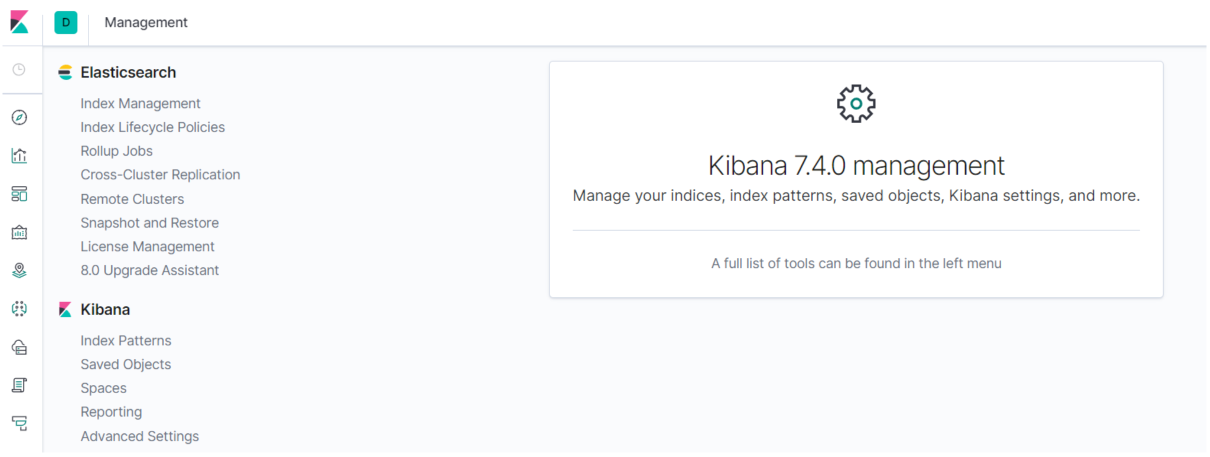


Figure 7-7: Kibana GUI

The last step of the setup is to deploy the Logstash on the worker node. Logstash will be deployed with “7.17.9” on port “9600”. The Logstash installation can be verified below.

```
root@worker2:~# curl -XGET 'localhost:9600/?pretty'
{
  "host" : "worker2",
  "version" : "7.17.9",
  "http_address" : "127.0.0.1:9600",
  "id" : "22005e8e-ea2d-447e-ac2f-312bd056d809",
  "name" : "worker2",
  "ephemeral_id" : "b89458a0-5cf3-4bc7-8681-075bab1ab666",
  "status" : "green",
  "snapshot" : false,
  "pipeline" : {
    "workers" : 8,
    "batch_size" : 125,
    "batch_delay" : 50
  },
  "build_date" : "2023-01-24T22:02:08+00:00",
  "build_sha" : "1f111758bcb47564ae897fc4fbb8b7c7b50883b7",
  "build_snapshot" : false
}root@worker2:~#
```

7.1.3.2 Logstash Configuration

The ELK component of Logstash acts as an integration point between the Kafka server and elastic search. The Logstash will subscribe to the Kafka topic, and it pulls the data from the Kafka topic as soon as new data is added by the Python producer. Figure 7-8 the pulling of the data from the Kafka server by Logstash, and then it is loaded to elastic search. Kibana is used to visualize the loaded data in the elastic search.

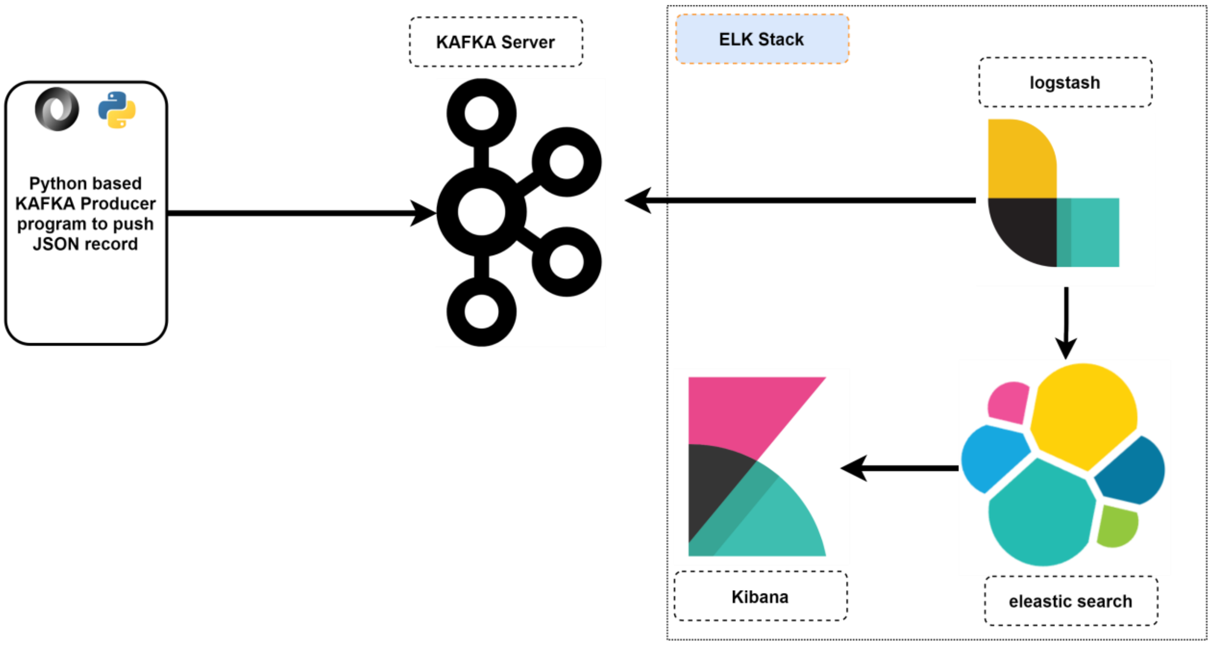


Figure 7-8: Data Ingestion via Logstash

The Logstash application of the ELK stack subscribes to Kafka topics, and in the lab setup, Logstash subscribes to the “pfcp”, “N2”, and “http2” Kafka topics. Logstash creates three different data pipelines to ingest the data from these Kafka topics, and the pipeline will monitor the Kafka topics for the new data and ingest it as soon as it is available. The ingested data from the Kafka broker is in JSON format, but it is possible to apply a filter on the data to transform it further before pushing the data toward the elastic search.

Logstash pipeline configured for the lab environment connects to the Kafka broker IP at port 9092 as input to ingest the data from the Kafka topics. The sample configuration to ingest data from "pfcp" Kafka topics is below.

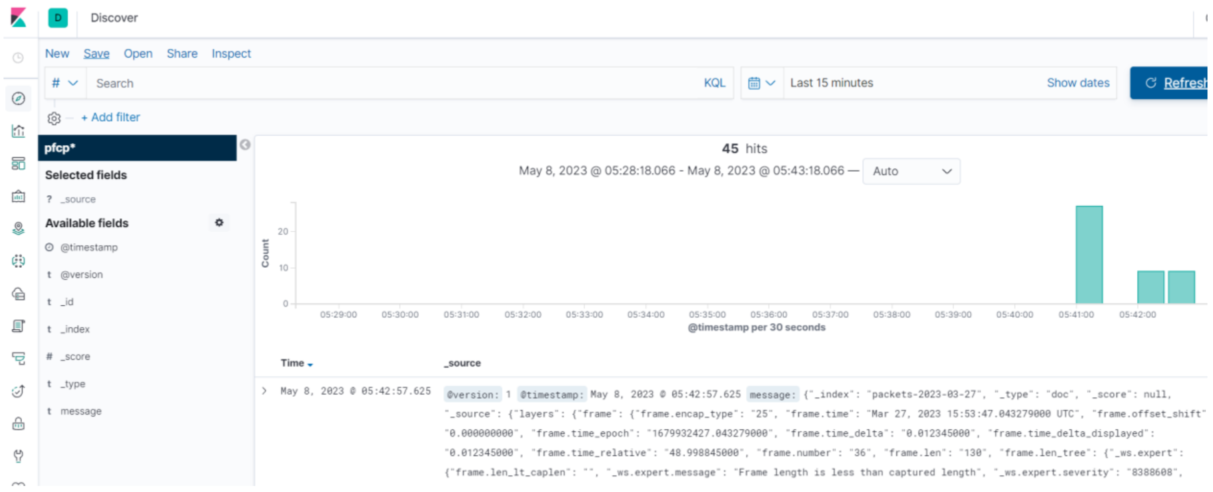
```
input {
    kafka {
            bootstrap_servers => "10.0.71.29:9092"
            topics => ["pfcp"]
    }
}
```

Logstash parses the received data and transforms it into the elastic search supported format. It has a rich filtering option for data transformation, and the most popular plugin used is called Grok filter plugin (GROK, 2023). Grok enables the parsing of unstructured data into structured and searchable format. The last step in the Logstash configuration is to write the data into the elastic search. Logstash will connect to the IP and port 9200 of the elastic search application and specify the index to push the data into elastic search. The sample configuration to push data into elastic search with index "pfcp" is below.

```
output {
   elasticsearch {
      hosts => ["10.0.71.29:9200"]
      index => "pfcp"
      workers => 1
    }
}
```

7.1.3.3 Data in Kibana

The data published from Logstash to the elastic search is seen in the Index Management of the elastic search application. The next step is to make the data available to Kibana, and this is achieved by creating an index inside Index Patterns in the Kibana setting. Figure 7-9 shows the "pfcp" received data in the Kibana GUI after the creation of the index, and this data can be searched and explored further based on the needs and requirements. It is possible to create the monitoring dashboard in Kibana based on the received data. The dashboard helps to understand the network traffic per protocol and monitoring of the network functions in real-time, and it enables the operator to protect the network against any cybersecurity attack.



```
May 8, 2023 @ 05:42:57.625   @version: 1 @timestamp: May 8, 2023 @ 05:42:57.625 message: {"_index": "packets-2023-03-27", "_type": "doc", "_score": null,
"_source": {"layers": {"frame": {"frame.encap_type": "25", "frame.time": "Mar 27, 2023 15:53:47.043279000 UTC", "frame.offset_shift"
"0.000000000", "frame.time_epoch": "1679932427.043279000", "frame.time_delta": "0.012345000", "frame.time_delta_displayed":
"0.012345000", "frame.time_relative": "48.998845000", "frame.number": "36", "frame.len": "130", "frame.len_tree": {"_ws.expert":
{"frame.len_lt_caplen": "", "_ws.expert.message": "Frame length is less than captured length", "_ws.expert.severity": "8388608",
```

Figure 7-9: "pfcp" data in Kibana

8 DISCUSSION

Communication is the backbone of the modern era, and data traffic becomes increasingly heterogeneous. The dependency of humans and even machines on communication networks are rapidly increasing, and this exponential increase requires a robust, reliable, resilient, and flexible network to be deployed on the modern infrastructure to meet the traffic growth. Subsequently, the infrastructure scaling demands minimal downtime, and it is mandatory to deliver the desired network performance to all consumers continuously. Networks have evolved in the past decades according to the traffic and service needs. The latest generation network, 5G, addresses the future needs of the society to establish advanced connectivity. The 5G network architecture is based on robust, adaptable, and dependable philosophy. It solves the design complexity of its predecessors and introduces a straightforward communication strategy using strict standards without any customization. 5G promises to deliver improved throughput and latency, a new ecosystem with new services, and new opportunities for the operators and industries.

Unambiguously, the design proposed by 3GPP for the 5G network is based on an innovative approach. The 5G Next-Generation Radio Access Network (NG-RAN) flexible architecture allows the network to operate in different modes, such as non-standalone and standalone. Hence, this allows the operator to enable the NR on the existing network infrastructure and achieve improved quality of service. The 5G NG-RAN also brings the concept of disintegrated RAN, where RAN is separated into distributed and centralized units based on different options to reduce the operational cost and deliver better performance. The 5G core network architecture is based on the Service Based Architecture (SBA) and this framework enables operators to create tailored slices for the mission-critical application with more resiliency and robustness without creating any managing additional network infrastructure. 3GPP has defined all the architecture and security options in its specification (ETSI.133.501, 2018) and some decision are being left open for the world operators. The operators can adapt to the network architecture depending on the needs and requirements based on operating country and its geopolitical situation.

As we have shown, the real-time monitoring of the 5G network based on network metrics is not a novel approach. However, the tools, metrics, and processes required to monitor the new architectural networks on the cloud-native infrastructure require advancement to the existing ecosystem. The new architecture demands implementation and extensive experimentation on the established 5G network in the control environment. In the implementation and experimentation phase of the thesis, the OpenAirInterace5G core network was established on the cloud-native environment. The lab settings, however, had limited resources, and environmental stability was also posing a challenge as cloud resources are being shared between tenants. The automation to keep such an environment stable becomes a mandatory requirement. Hence, MLN, Puppet, and Foreman are used to automate the creation of cloud-native infrastructure. The cloud-native infrastructure with Docker and Kubernetes is deployed to onboard the OAI 5G core network. The module used from Puppet encountered issues with the docker and Kubernetes versions and repositories during the creation process of the Kubernetes cluster, which been addressed by modifying the module with appropriate versions, fingerprints, and repositories of Docker and Kubernetes. The resource limitation was addressed by expanding the cloud resource quota for the 5G project, selecting the appropriate VM flavor, and assigning the minimal resources to the 5G network function pods in the helm charts.

The cloud-native environment enhances operational excellence with cost reduction, but it requires novel methods to mirror the traffic to the probing system. The top of the rack method used in the legacy architectures is ineffective due to the nature of cloud environments. One solution is to capture the traffic in a cloud via sidecar containers associated with the network functions. The sidecar containers introduce operational overhead and require compute resources, resulting in more cost to the solutions. We have demonstrated a cost-efficient and operational solution, where we capture the traffic on the worker interfaces and mirror it to the probing system. Another suggested efficient solution is to build the vtap interface within the network functions and allow the network function to perform the mirroring process directly from the running pod to the probing system.

The data generated in the 5G network from all the network functions is immense, and storing such captured data becomes expensive for the operators as it requires more storage. Therefore, a robust data management system is required where older data can be discarded while retaining more fresh versions. Consequently, a comprehensive backup solution is needed as the metrics can be critical for applying various machine learning techniques. Furthermore, it is a challenge to maintain data confidentiality and integrity as well. The appropriate metrics collection from the network functions without any duplication is necessary. Hence, this data metrics are a tradeoff between the quality vs quantity of the data. The suitable selection of interfaces and protocols from the various network functions becomes relevant. For instance, messages processed via the N4 interface can be captured from SMF and UPF simultaneously, hence such duplication is overcome by capturing the metrics at the single network functions or filtering the duplicate traffic before forwarding the traffic to the probing system. The metrics applicable to external interfaces such as N2, N6, N3, and N4 (in case of edge deployment scenario) are more relevant, involving essential data to monitor the 5G network. In contrast to the metrics related to HTTP/2 communications happening within the network functions of the same cluster, the metrics from external interfaces are essential for real-time network monitoring and facilitating protection of the network against cyberattacks. However, the metrics from the interfaces within the same cluster can be also utilized to protect the network from traffic loads by real-time scaling of the network functions and enabling the option to expand and scale the infrastructure. Additional interfaces related to protocol metrics can be added to the data store based on the needs and requirements of the operator.

Another advantage is the possibility to use Machine Learning (ML) system based on the monitoring data collected. ML systems can be designed to protect the network by triggering the logic to implement a specific security decision in the system. The real-time available data for the machine learning system is vital. A real-time data ingestion by a ML system opens the possibility for designing completely autonomous networks. Data availability in real-time to ML is only achieved with a cloud-native streaming tools that are highly scalable, offer low latency, and high throughput. The tools should also be exceptionally dependable and durable, in order to guarantee data availability to the ML system in case of failures. As indicated in our work, Kafka a tool with such features that allows for data streaming options for multiple systems. The Kafka cluster comprises multiple brokers and stores the data in various partitions to achieve data resiliency. The received data from the 5G network functions need separation at the protocol and interface levels for better processing and achieving better results in the ML system. This is accomplished by introducing the Kafka topics, where each protocol can be mapped to a specific Kafka topic. This was demonstrated by creating Kafka topics for three interfaces and allowing the ML system to ingest the data from those topics separately. The same data can be consumed by multiple ML systems and data can be made available to multiple monitoring systems.

According to data collection and processing results in the experimentation phase, the data is ingested into the ELK stack from Kafka to enable visualization and dashboard creation in Kibana. The dashboard creation provides the flexibility to visualize the network in a single place, and customized dashboards can be created on-demand if data is available in the Elasticsearch database. The amount of data ingested in the lab was minimal due to resource restrictions and the environment's limitations, but this is not relevant as we do not focus on providing quality data for testing machine learning systems. The data ingested is only for the three interfaces associated with SMF, AMF, and UPF; without numerous performed efforts to convert this data into the Kibana dashboard. However, the end goal was achieved by capturing, processing, and then loading the data into Kafka, as well as making the data available for a future machine learning system that can be implemented. Consequently, the data was also successfully loaded into the ELK stack for visualization.

9 CONCLUSION

In this work, we have studied the possibility to implement a monitoring and data collection capability within a 5G system. The cloud-native environments pose a challenge to the existing data capturing frameworks and require adaptations to dynamic architectures. It is vital to note that various data collection and capturing techniques can be applied to ensure and guarantee privacy, data integrity and anonymization while attaining the required monitoring capabilities for the evolving mobile networks. With the introduction of the Internet of Things and smart infrastructure, the number of connected devices is rising considerably. As a result, the amount of data that flows through the 5G networks is becoming immense and this is a challenge from different standpoints. However, the biggest challenge the 5G must address is the possibility to enable dynamic scaling and reduce operational costs because of the constant increase of connected entities. To do so, the control plane of 5G should be robust and provide a stable foundation for building the necessary applications that shall support different industry verticals. The data which flows through the control plane is crucial for the management of the 5G networks and therefore, it is important to understand how to collect and store the relevant information for further processing and facilitating network autonomy. This includes crafting reliable cybersecurity systems that can employ more sophisticated principles, such as artificial intelligence, to provide the necessary confidentiality, integrity and protection in the ever-growing threats and attack vectors on modern systems. Conclusively, the demonstrated data collection framework provides an insight into the possibilities that can arise from applying the correct philosophy of monitoring, to design future systems that shall ensure more secure society and mitigate various threats.

We have successfully implemented and demonstrated the framework in the virtualized controlled environment of the Oslo Metropolitan University lab. The proposed dynamic architecture of framework based on Kafka engine makes it extremely scalable, reliable, and durable. Hence, supporting of scaling and reconfiguration at ease or on demand without downtime. The experimentation phase of the thesis successfully demonstrated the implementation of a data capturing technique to capture real-time data from the 5G NFs. The data are collected from the control plane and user plane separately at interface level. It provides elasticity to an operator on a selection of collected data to be incorporated for different use cases. The data presentation and multi-tenancy concept has also been executed and proven in thesis by fulfilling the separation techniques on the collected data from the 5G network functions and visualization of the data via open-sources tools.

The proposed framework fulfills the data integrity and privacy requirement by demonstrating and implementing data separation techniques. The separation concept applied in the thesis can be scaled out to different network slices. Hence, the framework guarantees the adaptability to future heterogenous traffic growth. Similarly, the protection of confidentiality and integrity of data exchanged over the NAS has been demonstrated in thesis with study of encrypted traffic between a mobile station and eNodeB/AMF. Data anonymization requires masking of specific fields in the capture data to hide the user-specific information and it needs implementation on stored data as well. In the thesis, the data integrity and data anonymization techniques are discussed briefly. Consequently, the deeper experimentation is required to explore the data integrity techniques from all angles. The work on data integrity and anonymization are partial in this thesis and it needs further study while keeping in mind the security aspects of the 5G networks.

9.1 Future Work

Data can be monetized, and it is referred to as "gold" in some instances, therefore, the raw data needs to be ingested and structured appropriately to create a useful product. The legacy method to probe the network for capturing the data is insufficient for cloud based implementations of 5G. Hence, new methods and frameworks need to be developed to overcome the shortcoming of the legacy monitoring tools and adapt to emerging technologies.

The data-capturing techniques demonstrated in this thesis were performed on specific interfaces of 5G core network functions. In tandem, NG-RAN disaggregated architecture allows the data capture at DU and CU interfaces as well. Hence, future research can utilize the proposed framework in this thesis to be applied for the NG-RAN. It is possible to include capturing traffic on the DU and CU interfaces, along with the data integrity and privacy techniques applicable to the NG-RAN. The inclusion of various kinds of handover information in future data processing will enhance the usability of the new framework.

As an example, the Unstructured Data Storage Function (UDSF) is defined as an optional network function for storing and retrieving unstructured data from 5G network functions. This thesis study includes the UDSF in the 5G architecture, but the usage of UDSF to store and retrieve various kinds of statistical data has not been performed. The future researcher can explore the option to utilize the UDSF to store the captured from different interfaces, separation of data in UDSF to ensure confidentiality and methods to expose the data to the machine learning systems via NEF. This study will be attractive for most operators as this will utilize the existing network functions (NEF) to expose the statistical data to multiple different machine learning systems.

Network Data Analytics Function (NWDAF) is a concept that follows the accomplishment of this thesis work, and it is still getting towards its maturity. NWDAF is a network function proposed by 3GPP to collect the statistics from the 5G core network function directly from the event exposure interface. NWDAF excludes all the inefficient data collection methods and provides the standard approach for data collection from all the 5G NFs. NWDAF aims to provide statistical information on past events or predictive information based on the collected data. Future works can include the study of NWDAF analytics, serving, and training models based on NF load statistics. The exploration of new algorithms aims to understand the network data pattern to generate predictions. Further study is required to explore the supervising methods for monitoring the serving model predictions, continuously and according to the prediction precision model. The training model needs to be ready to serve the production environments, so manual and dynamic processes need to be defined in future research to replace the old serving model with a newly trained model.

The innovative technologies bring security challenges, and it opens the path for adversaries and security threats. The data collection allows the operator to collect substantial amount of raw data from a 5G network, including the customer information and network traffic load evidence. Hence, future studies can include vital topics such as the anonymization of the data completely before storing it in data storage systems, data integrity from source to destination by applying verification techniques, and security techniques to encrypt the data while transmitting or during storage and archiving.

A REFRENCES

3GPP-TS.23.501 (2022) ‘System architecture for the 5G System 23501-h40’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144.

3GPP-TS.23.502 (2022) ‘Procedures for the 5G System 23502-h40’. Available at: https://www.3gpp.org/ftp/Specs/archive/23_series/23.502/.

3GPP-TS.24.302 (2022) ‘Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks’, (Release 8). Available at: https://www.3gpp.org/ftp/Specs/archive/24_series/24.302/.

3GPP-TS.29.503 (2022) ‘UDM 29503-h60’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3342.

3GPP-TS.29.504 (2022) ‘UDR 29504-h60’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3405.

3GPP-TS.29.508 (2022) ‘SMF (AMF) 29508-h60’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3351.

3GPP-TS.29.509 (2022) ‘AUSF 29509-h50’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3343.

3GPP-TS.29.510 (2022) ‘NRF29510-h50’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3345.

3GPP-TS.29.512 (2022) ‘PCF (SMF)29512-h60’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3352.

3GPP-TS.29.514 (2022) ‘PCF (AF+NEF) 29514-h40’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3353.

3GPP-TS.29.518 (2022) ‘AMF 29518-h50’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3339.

3GPP-TS.29.522 (2022) ‘NEF - AF 29522-h50’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3437.

3GPP-TS.29.531 (2022) ‘NSSF 29531-h40’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3346.

3GPP-TS.29.542 (2022) ‘SMF (NEF)29542-h30’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3716.

3GPP-TS.29.551 (2022) ‘NEF SMF 29551-h60’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3391.

3GPP-TS.29.564 (2022) ‘UPF 29564-h00’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3938.

3GPP-TS.29.591 (2022) ‘NEF - NWDAF 29591-h50’. Available at: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3681.

3GPP-TS.33.501 (2022) ‘Security architecture and procedures for 5G system 33.510’. Available at: https://www.3gpp.org/ftp/Specs/archive/33_series/33.501/.

3GPP-TS.38.801 (2017) ‘Study on new radio access technology: Radio access architecture and interfaces’, 0(Release 14). Available at: https://www.3gpp.org/ftp/Specs/archive/38_series/38.801/.

3GPP (2022) *3GPP*. Available at: https://www.3gpp.org/ (Accessed: 11 October 2022).

3GPP (2023) *Split Decision Option-2*. Available at: https://www.3gpp.org/news-events/3gpp-news/open-ran (Accessed: 3 March 2023).

Afneh, M. (2023) *Protocol Stack*. Available at: https://novelbits.io/protocol-stacks-layered-architecture/ (Accessed: 18 April 2023).

Alani, M. M. (2014) *Guide to OSI and TCP/IP Models*, *Springer*. Springer. Available at: http://www.springer.com/series/10028.

Begnum, K. (2009) *MLN*. Available at: https://mln.sourceforge.net/ (Accessed: 10 November 2022).

Bertenyi, B. *et al.* (2018) ‘NG Radio Access Network (NG-RAN)’, *Journal of ICT Standardization*, 6(1), pp. 59–76. doi: 10.13052/jicts2245-800X.614.

Cisco (2023a) ‘End Marker Packets’. Available at: https://www.cisco.com/c/en/us/td/docs/wireless/upc/21-27/cups-cp-admin/21-27-upc-cups-cp-admin-guide/m_end-marker-packets.pdf.

Cisco (2023b) *Mirroring*. Available at: https://www.cisco.com/c/en/us/td/docs/routers/ncs6000/software/ncs6k-7-6/b-interfaces-hardware-component-cg-ncs6000-76x/configuring-traffic-mirroring.html (Accessed: 25 March 2023).

Cox, C. (2012) *AN INTRODUCTION TO LTE: LTE, LTE-Advance, SEA, VoLTE and 4G Mobile Communications*. second edi. WILEY. Available at: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118818046.

Denenberg, R. (1985) ‘Open Systems Interconnection’, *Library Hi Tech*, 3(1), pp. 15–26. doi:

10.1108/eb047578.

DOCKER.INC (2022) *Home - Docker*. Available at: https://www.docker.com/ (Accessed: 11 September 2022).

Docker (2023) *Bridge*. Available at: https://docs.docker.com/network/bridge/ (Accessed: 13 March 2023).

ELASTIC (2023) *Elastic Stack*. Available at: https://www.elastic.co/elastic-stack/ (Accessed: 25 March 2023).

ERICSSON (2023) *Traffic Growth*. Available at: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast (Accessed: 11 January 2023).

ETCD (2023) *ETCD*. Available at: https://etcd.io/ (Accessed: 11 April 2023).

ETSI-TS.138.410 (2020) 'TS 138 410 - V16.3.0 - 5G; NG-RAN; NG general aspects and principles (3GPP TS 38.410 version 16.3.0 Release 16)', 0, pp. 0–18. Available at: https://www.etsi.org/deliver/etsi_ts/138400_138499/138410/16.03.00_60/ts_138410v160300p.pdf.

ETSI-TS.138.413 (2020) 'TS 138 413 - V16.2.0 - 5G; NG-RAN; NG Application Protocol (NGAP) (3GPP TS 38.413 version 16.2.0 Release 16)', 0. Available at: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx.

ETSI.123.288 (2020) 'Architecture enhancements for 5G System (5GS) to support network data analytics services (3GPP TS 23.288 version 16.4.0 Release 16)', 0, pp. 0–58. Available at: https://www.etsi.org/deliver/etsi_ts/123200_123299/123288/16.04.00_60/ts_123288v160400p.pdf.

ETSI.123.501-R17 (2022) 'TS 123 501 - V16.12.0 - 5G; System architecture for the 5G System (5GS) (3GPP TS 23.501 version 16.12.0 Release 16)', 0. Available at: https://portal.etsi.org/webapp/https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/17.05.00_60/ts_123501v170500p.pdf.

ETSI.129.244 (2020) 'Interface between the Control Plane and the User Plane nodes (3GPP TS 29.244 version 16.5.0 Release 16)', 0. Available at: https://www.etsi.org/deliver/etsi_ts/129200_129299/129244/16.06.00_60/ts_129244v160600p.pdf.

ETSI.129.500 (2018) 'Technical Realization of Service Based Architecture', 0, pp. 0–28. Available at: https://www.etsi.org/deliver/etsi_ts/129500_129599/129500/16.07.00_60/ts_129500v160700p.pdf.

ETSI.129.598 (2020) 'TS 129 598 - V16.1.0 - 5G; Unstructured data storage services (3GPP TS 29.598 version 16.1.0 Release 16)', 0. Available at: https://www.etsi.org/deliver/etsi_ts/129500_129599/129598/16.01.00_60/ts_129598v160100p.pdf.

ETSI.133.501 (2018) 'TS 133 501 - V15.2.0 - 5G; Security architecture and procedures for 5G System (3GPP TS 33.501 version 15.2.0 Release 15)', 0. Available at: https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.04.00_60/ts_133501v150

400p.pdf.

ETSI.138.300 (2021) ‘TS 138 300 - V16.4.0 - 5G; NR; NR and NG-RAN Overall description; Stage-2 (3GPP TS 38.300 version 16.4.0 Release 16)’, 0. Available at: https://www.etsi.org/deliver/etsi_ts/138300_138399/138300/16.02.00_60/ts_138300v160200p.pdf.

ETSI.138.401 (2018) ‘TS 138 401 - V15.2.0 - 5G; NG-RAN; Architecture description (3GPP TS 38.401 version 15.2.0 Release 15)’, 0, pp. 0–39. Available at: https://www.etsi.org/deliver/etsi_ts/138400_138499/138401/15.05.00_60/ts_138401v150500p.pdf.

ETSI.138.420 (2020) ‘TS 138 420 - V16.0.0 - 5G; NG-RAN; Xn general aspects and principles (3GPP TS 38.420 version 16.0.0 Release 16)’, 0, pp. 0–17. Available at: https://www.etsi.org/deliver/etsi_ts/138400_138499/138420/16.00.00_60/ts_138420v160000p.pdf.

ETSI.138.460 (2020) ‘TS 138 460 - V16.1.0 - 5G; NG-RAN; E1 general aspects and principles (3GPP TS 38.460 version 16.1.0 Release 16)’, 0, pp. 0–12. Available at: https://www.etsi.org/deliver/etsi_ts/138400_138499/138460/16.01.00_60/ts_138460v160100p.pdf.

ETSI.138.470 (2020) ‘TS 138 470 - V16.2.0 - 5G; NG-RAN; F1 general aspects and principles (3GPP TS 38.470 version 16.2.0 Release 16)’, 0, pp. 0–17. Available at: https://www.etsi.org/deliver/etsi_ts/138400_138499/138470/16.02.00_60/ts_138470v160200p.pdf.

ETSI (2015) *5G - Spider Web, 2015*. Available at: https://www.etsi.org/technologies/mobile/5g (Accessed: 12 November 2022).

ETSI 123 501- V15.5.0 -5G (2019) ‘System architecture for the 5G System’, *Etsi*, 0. Available at: https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx.

EURECOM (2022a) *Accueil | EURECOM*. Available at: https://www.eurecom.fr/en (Accessed: 11 September 2022).

EURECOM (2022b) *EURECOM - OAI 5G*. Available at: https://gitlab.eurecom.fr/oai/cn5g (Accessed: 11 September 2022).

FOREMAN (2022) *FOREMAN*. Available at: https://theforeman.org/ (Accessed: 23 February 2023).

FREE5GC (2022) *free5GC*. Available at: https://www.free5gc.org/ (Accessed: 11 September 2022).

GROK (2023) *GROK*. Available at: https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html (Accessed: 21 April 2023).

HELM (2022) *HELM*. Available at: https://helm.sh/ (Accessed: 10 February 2023).

HEXA-X (2022) *Hexa-X – Simu5G – the first real-time open source 5G simulator – will support Federated XAI within Hexa-X project*. Available at: https://hexa-x.eu/simu5g-the-first-real-time-open-source-5g-simulator-will-support-federated-xai-within-hexa-x-project/ (Accessed:

11 September 2022).

IEEE (2022) *IEEE SA - Registration Authority*. Available at: https://standards.ieee.org/products-programs/regauth/ (Accessed: 17 May 2022).

ITU-T (2014) 'Architecture framework for the development of signalling and OA&M protocols using OSI concepts', *Paper Knowledge . Toward a Media History of Documents*, 4. Available at: https://www.itu.int/rec/T-REC-Q.1400-199303-I/en.

JQUERY (2023) *jQuery*. Available at: https://jquery.com/ (Accessed: 12 May 2023).

JSON (2023) *JSON*. Available at: https://www.json.org/json-en.html (Accessed: 11 March 2023).

KAFKA (2023) *KAFKA*. Available at: https://kafka.apache.org/ (Accessed: 12 April 2023).

Kozina, D., Soós, G. and Varga, P. (2016) 'Supporting LTE network and service management through session data record analysis', *Infocommunications Journal*, 8(2), pp. 11–16. Available at: https://dl.ifip.org/db/conf/im/im2017-ws1-annet/154.pdf.

KUBERNETES (2023) *Kubernetes*. Available at: https://kubernetes.io/ (Accessed: 12 February 2023).

Kuechler, B. and Vaishnavi, V. (2008) 'Theory development in design science research: Anatomy of a research project', *Proceedings of the 3rd International Conference on Design Science Research in Information Systems and Technology, DESRIST 2008*, (January), pp. 1–15. doi: 10.1201/b18448-8.

Lei, W. *et al.* (2020) *5G System Design: An End to End Perspective*. Second Edi. Springer. Available at: https://link.springer.com/book/10.1007/978-3-030-22236-9.

Lin, X. and Lee, N. (2021) *5G and Beyond: Fundamentals and Standards*. 1st ed. 20. Springer. Available at: https://www.amazon.com/Beyond-Fundamentals-Standards-Xingqin-Lin/dp/3030581969.

M. Stewart, J., Chapple, M. and Gibson, D. (2012) 'Certified Information Systems Security Professional Study Guide', in *Certified Information Systems Security Professional*. 6th edn. Sybex, pp. 88–145. Available at: https://www.amazon.com/CISSP-Certified-Information-Security-Professional/dp/1118314174.

MOZILLA (2023) *HTTP Methods*. Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (Accessed: 20 March 2023).

MULTUS (2023) *MULTUS*. Available at: https://github.com/k8snetworkplumbingwg/multus-cni (Accessed: 23 January 2023).

OPEN5GS (2023) *Open 5GS*. Available at: https://open5gs.org/ (Accessed: 25 January 2023).

OPENSTACK (2023) *Openstack*. Available at: https://www.openstack.org/ (Accessed: 20 January 2023).

Palat, S. and Godin, P. (2011) *LTE – The UMTS Long Term Evolution: From Theory to Practice*. 2nd Editio. WILEY. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470978504.ch2.

PROJECT-PRO (2023) *KAFKA Advantages*. Available at: https://www.projectpro.io/article/apache-kafka-architecture-/442 (Accessed: 15 March 2023).

PUPPET (2022) *PUPPET*. Available at: https://puppet.com/ (Accessed: 22 February 2023).

PUPPET (2023a) *Kubetool*. Available at: https://github.com/puppetlabs/puppetlabs-kubernetes/blob/main/tooling/kube_tool.rb (Accessed: 18 February 2023).

PUPPET (2023b) *Puppet Modules*. Available at: https://www.puppet.com/blog/puppet-kubernetes (Accessed: 1 April 2023).

PYTHON (2023) 'python'. Available at: https://www.python.org/.

RED HAT (2022) *Administration and Deployment Guide*. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html-single/virtualization_deployment_and_administration_guide/index (Accessed: 20 February 2023).

REDHAT (2022a) *Redhat interface mirroring*. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/assembly_port-mirroring_configuring-and-managing-networking (Accessed: 23 March 2023).

REDHAT (2022b) *Virtual Switch*. Available at: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_virtualization/configuring-virtual-machine-network-connections_configuring-and-managing-virtualization (Accessed: 24 January 2023).

RF-Wireless (2023) *RF Wireless - gNB stack*. Available at: https://www.rfwireless-world.com/Terminology/5G-Protocol-Stack-Layer-1-Layer-2-and-Layer-3.html (Accessed: 11 February 2023).

RFC5246 (2023) 'Transport Handshake'. Available at: https://www.ietf.org/rfc/rfc5246.txt.

RFC6749 (2012) *OAuth*. Available at: https://datatracker.ietf.org/doc/html/rfc6749 (Accessed: 20 April 2023).

RFC7540 (2015) *HTTP2*. Available at: https://httpwg.org/specs/rfc7540.html (Accessed: 23 March 2023).

RFC791 (1981) *RFC 791*. Available at: https://www.rfc-editor.org/rfc/rfc791 (Accessed: 30 October 2022).

Russell, A. L. (2013) *OSI: The Internet That Wasn't*. Available at: https://spectrum.ieee.org/osi-the-internet-that-wasnt#toggle-gdpr (Accessed: 26 October 2022).

Schmitt, J. (2023) *HELM Package*. Available at: https://circleci.com/blog/what-is-helm/ (Accessed: 14 March 2023).

See, A. von (2021) *M2M (machine-to-machine) - Statistics & Facts*. Available at: https://www.statista.com/topics/1843/m2m-machine-to-machine/#topicHeader__wrapper (Accessed: 21 April 2023).

SIMU5G (2022) *Simu5G - 5G New Radio User Plane Simulator for OMNeT++ and INET*. Available at: http://simu5g.org/ (Accessed: 11 September 2022).

Technologies Inc, J. (2007) *Networks Protocol Handbook*. 4th Editio. Javvin Press. Available at: https://www.amazon.com/Network-Protocol-Handbook-4th-www-Javvin-com/dp/1602670021.

Thales (2023) *HELM SPRAY*. Available at: https://github.com/ThalesGroup/helm-spray (Accessed: 29 January 2023).

TRIANZ (2022) *Virtualization & Containerization*. Available at: https://www.trianz.com/insights/containerization-vs-virtualization#:~:text=Containerization is a form of,isolate processes from one another. (Accessed: 11 April 2023).

TSHARK (2023) *Tshark*. Available at: https://linux.die.net/man/1/tshark#:~:text=TShark is a network protocol,the packets to a file. (Accessed: 22 January 2023).

Ubuntu (2023) *Containerization*. Available at: https://ubuntu.com/blog/containerization-vs-virtualization (Accessed: 21 April 2023).

UTEL (2023) *Monitroing - UTEL*. Available at: https://utel.tech/resource/4-reasons-why-you-need-real-time-telecoms-network-monitoring/ (Accessed: 12 February 2023).

Venable, J. R., Pries-Heje, J. and Baskerville, R. (2017) 'Choosing a Design science research methodology', *Proceedings of the 28th Australasian Conference on Information Systems, ACIS 2017*. Available at: https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1111&context=acis2017.

B APPENDIX

A. KubeTOOL (PUPPET, 2023a)

```
#!/usr/bin/env ruby

require 'slop'
require_relative 'kube_tool/pre_checks.rb'
require_relative 'kube_tool/create_certs.rb'
require_relative 'kube_tool/clean_up.rb'
require_relative 'kube_tool/other_params.rb'

class Kube_tool
  def self.parse_args
    begin
      opts = Slop.parse do |o|
        o.string '-o', '--os', 'The OS that Kubernetes will run on', default: ENV['OS']
        o.string '-v', '--version', 'The Kubernetes version to install', default: ENV['VERSION']
        o.string '-r', '--container_runtime', 'The container runtime to use. This can only be "docker" or
"cri_containerd"', default: ENV['CONTAINER_RUNTIME']
        o.string '-c', '--cni_provider', 'The networking provider to use, flannel, weave, calico, calico-
tigera or cilium are supported', default: ENV['CNI_PROVIDER']
        o.string '-p', '--cni_provider_version', 'The networking provider version to use, calico and cilium
will use this to reference the correct deployment download link', default: ENV['CNI_PROVIDER_VERSION']
        o.string '-t', '--etcd_ip', 'The IP address etcd will listen on', default: ENV['ETCD_IP']
        o.string '-i', '--etcd_initial_cluster', 'The list of servers in the etcd cluster', default:
ENV['ETCD_INITIAL_CLUSTER']
        o.string '-a', '--api_address', 'The IP address (or fact) that kube api will listen on', default:
ENV['KUBE_API_ADVERTISE_ADDRESS']
        o.int '-b', '--key_size', 'Specifies the number of bits in the key to create', default:
ENV['KEY_SIZE'].to_i
        o.int '--ca_algo', 'Algorithm to generate CA certificates, default: ecdsa', default: ENV['CA_ALGO']
        o.int '--sa_size', 'Service account key size', default: ENV['SA_SIZE'].to_i
        o.bool '-d', '--install_dashboard', 'Whether install the kube dashboard', default:
ENV['INSTALL_DASHBOARD']
        o.on '-h','--help', 'print the help' do
          puts o
          exit
        end
      end

      options = opts.to_hash
      options[:key_size] = 256 if options[:key_size] < 1
      options[:sa_size] = 2048 if options[:sa_size] < 1
      options[:ca_algo] ||= 'ecdsa'
      options[:container_runtime] ||= 'cri_containerd'
      options[:version] ||= '1.25.4'
      options[:os] ||= 'Debian'
      if options[:etcd_initial_cluster].nil?
        abort('Please provide IP addresses for etcd initial cluster -i/--etcd_initial_cluster (ENV
ETCD_INITIAL_CLUSTER)')
      end
      puts options
      return options

    rescue Slop::Error => e
      puts "ERROR: #{e.message}"
      exit 1
    end
  end

  def self.build_hiera(opts)
    OtherParams.create(opts)
    PreChecks.checks
    certs = CreateCerts.new(opts)
    certs.etcd_ca
    certs.etcd_clients
    certs.etcd_certificates
    certs.kube_ca
    certs.kube_front_proxy_ca
    certs.sa
    CleanUp.remove_files
    CleanUp.clean_yaml(opts[:os])
  end
```

```
end
Kube_tool.build_hiera(Kube_tool.parse_args)
```

B. Kubernetes – DEBIAN YAML FILE

```
kubernetes::kubernetes_version: 1.20.1
kubernetes::kubernetes_package_version: 1.20.1-00
kubernetes::container_runtime: docker
kubernetes::cni_network_provider:
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
kubernetes::cni_pod_cidr: 10.244.0.0/16
kubernetes::cni_provider: flannel
kubernetes::etcd_initial_cluster: control=https://10.0.71.26:2380
kubernetes::etcd_peers:
- 10.0.71.26
kubernetes::etcd_ip: 10.0.71.26
kubernetes::kube_api_advertise_address: 10.0.71.26
kubernetes::api_server_count: 1
kubernetes::install_dashboard: false
kubernetes::controller_address: 10.0.71.26:6443
kubernetes::token: 86198a.5a0f86f75947a194

kubernetes::etcd_ca_crt: |
  -----BEGIN CERTIFICATE-----
  MIIC7jCCAdagAwIBAgIUVlGb0CzM/0ZECYXb3tA22yj7r48wDQYJKoZIhvcNAQEL
  BQAwDzENMAsGA1UEAxMEZXRjZDAeFw0yMzA0MDMwODA5MDBaFw0yODA0MDEwODA5
  MDBaMA8xDTALBgNVBAMTBGV0Y2QwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
  AoIBAQDJ6LXINnUVSmnaVvZZOC1w0CZ+5x0/8Qkc+mcThlANvQkiIjAApDHQa6Lp
  9ZxT3gNYJg4H3W7YxI6b6WWO1UssP3xq6LfJoMLN8lIQPkbfhsiGu/DmiaNx+pvO
  6HVHgLvp0sZJF/vnjNvTyyBe0q0o6WCMDx60YDezuM8dMhS41xE/+vnqiOV4YCxN
  2nLqnxsEMH7R+vIhWawzoNokXjJVApBDlr+lCaIHkRWbCbRzHYh/r6ONShDxUzoL
  X2kR4t4P4ZEUzDVa2renzJHMXpXPTg8xpKCR5GJs1CjPpBJcMKgq8eCo7HGyBMdw
  EDUfbOB+sHgx7dFFlyaSIFxrPM2jAgMBAAGjQjBAMA4GA1UdDwEB/wQEAwIBBjAP
  BgNVHRMBAf8EBTADAQH/MB0GA1UdDgQWBBR/8/w6+MSOcH9dbHvDty5QqXkNRzAN
  BgkqhkiG9w0BAQsFAAOCAQEAlINr30Bq9O/ielVkwWnLVIijUsfzG1X032zEs2AB
  y24oEsmchWAdQiG/Cx77x+FA9nX61+rZ/8lYYv4hhTMF0MW6aIThyMNitVD4p3Dx
  odtD9MmDozPPHTaZ5TlVvooG+q10ULQ60MstydY+P+L4Unycr/lfd8xcTwoEePes
  Rc714aV41TyjlM2B44DkEfBq/GrOK+pUOM2WA512A6QVSrKOf389ejt2PLkuyBx4
  WL7NGP3TygEscjwuWh6V2TydZT2BfaogrUY/YG5zJpb+xJJZygKVR/7+zDrbFGtA
  9uUV/lcPV7byOPdeV/2irLFrrjLjkZWhNwmG0RJhdyX1hg==
  -----END CERTIFICATE-----
kubernetes::etcd_ca_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpQIBAAKCAQEAyei1yDZ1FUpp2lb2WTgtcNAmfucdP/EJHPpnE4ZQDb0JIiIw
  AKQx0Gui6fWcU94DWCYOB91u2MSOm+lljtVLLD98aui3yaDCzfJSED5G34bIhrvw
  5omjcfqbzuh1R4C76dLGSRf754zb08sgXtKtKOlgjA8etGA3s7jPHTIUuNcRP/r5
  6ojleGAsTdpy6p8bBDB+0fryIVmsM6DaJF4yVQKQQ5a/pQmiB5EVmwm0cx2If6+j
  jUoQ8VM6C19pEeLeD+GRFMw1Wtq3p8yRzF6Vz04PMaSgkeRibNQoz6QSXDCoKvHg
  qOxxsgTHcBA1H2zgfrB4Me3RRZcmkiBcazzNowIDAQABAoIBAD5CvTNk34vWK3gB
  kuuGMDT2arh9Kf7ao9XEvV4+75ac8SEOa8D24MmFN3Jvo2oVLvDSo0b3mEPNsLAR
  M5js2sZlOUd6RZouMYPyMi4CFja9SD6L28grLzRpc/xEoE2RO+Dpdu15MNuBCxRv
  Macrzeuss3HdA7nZZbPMzEkTdWJdmqm4Oetjmh9gzb/cwLgJMle350e2I7q0Sm9A
  t+pcX4p8WIeYdsBdK91iZThC3/snsVB0QLXi59ZAHtnSqG/YZrvEjFbff2oenltp
  h4GGUK92sylHlbHADRCBRNUOGhSYKFVX+WKP9oryIfswqh+EMt+KpQYN1Wad/mis
  SIKuzxECgYEA9u1xGZSszqLR0C7wkhVC1WEUqCl3LXgs10iRT5JeA9Vmil5SV307
  SH84brwC2BXAayhe/fQcEC2sEDb80vm/bqhMWgw+2xmyaKhhfwrb7ibIffqoblWN
  5UrR0GJtyT9DbECSMk3O7IvPC3nclsNitWPwE0OIt6z40TFCdkPR2asCgYEA0VPV
  AzbXwBlnoiBTdJ3aG343rpUw94+zRX0NmdvwHs4ZeGYikTakDLWHRhHPYt6AadBL
  2z51g+XEdlr/W4aGsPuWevop7vOxzAOc2Gu6PN9+jduuHQNf8PIli2JlDYDFF1zy
  0adeuBMCiFmIb+E12qSekIY7Gd6AMmhEwgBCE+kCgYEA5Sikvt3Iphha6X62iowq
  s0ZeWlXpIYyW6NCS8qrej2Y7vIwelM1G+FgA8luPtCQzp/8WcU5bwPPx7DZr2gwn
  ibWs7iDRMJsfhJtHqBRW2SSrCqdWKtdBUZtnLqI6SvtoQg2G4CJvt/1kD71vZ4c6
  kyaFRgt11Sg03kd9662zEfMCgYEAtfRf+b5HRhUPLy9YO8smlrVdI2VNvjNsCmAM
  XYx/qddO7oghTgaoDOASKQc8NN3h05ibB4XAUpl778FjrKRAGkFm6ZjgsHOmPgd5
  4+moZleDQz4Ml+2iDOf/WKx72IPxTkCeVT1eKTVlAy6IWW/zBw4HQutbHj55Qte9
  xMDCr1kCgYEAvF2pS7GajLPxUKGOncgJ3WFVCzQ23SYSHmHe4RKYXOqsHlLOV29p
  utVJv4z/y8P/cEM6hU+wf+ZjrsfBZ7UNr75f0bT+p9VeWpw0K03RbVvc2VZK6xJc
  fJcazQq8gKGrAhtmyieMGQrAEl4+n7YCD4DMM0Z256ulnJnNLS4tyXI=
  -----END RSA PRIVATE KEY-----

kubernetes::etcdclient_crt: |
  -----BEGIN CERTIFICATE-----
  MIIDMjCCAhqgAwIBAgIUZvve1nl6gZxprM46rW4wEz8YPhwwDQYJKoZIhvcNAQEL
  BQAwDzENMAsGA1UEAxMEZXRjZDAeFw0yMzA0MDMwODA5MDBaFw0yODA0MDEwODA5
  MDBaMBExDzANBgNVBAMTBmNsaWVudDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCC
  AQoCggEBALpWsLZslvRj0+qWK+vtzHeck7YS0B7XSYN5vUvt1PwPe3Z/zY0KNQcF
  cG4pm2SMi5AoyLbXCE/m7HFjoEW9VZ79jd8Jqb4/ER4LKNGgaR0mfDj9kzZ+hVbN
  KHiU/cAsjtcc0iZkHu382TSHmC/KnKRqlKD4kYJPdBnU7acvEPyQwQeLigpmswRM
  qOK6riAyHeplhGsRDbNRe3v1rNKxPmWgxRu5rlySBGTIUFekIZHgFc01EkiZ9hu3
  acyV1DTtpohF8u6ZD98hLVnL0/W4oS731X2hDfu8d3odYRcFpI3oa7WzJbostkXs
  DPZ5Cu2yE7syj4lSUMDr5MAMcwNT7GUCAwEAAaOBgzCBgDAOBgNVHQ8BAf8EBAMC
  BaAwEwYDVR0lBAwwCgYIKwYBBQUHAwIwDAYDVR0TAQH/BAIwADAdBgNVHQ4EFgQU
  xDDn53c0I6uYZp9pZM61qlUoI9QwHwYDVR0jBBgwFoAUf/P8OvjEjnB/XWx7w7cu
  UKl5DUcwCwYDVR0RBAQwAoIAMA0GCSqGSIb3DQEBCwUAA4IBAQCxpkOnDVf1Bs2Z
  JE+wmZE3G2LqiVTWp6dL7Ajzfv5ZyBkOGy2PREC3iNr+Kojbf8KaNoabrK1GT8Hg
  jf3UVyoVb3B7zfnZSa2AN1NhIO+t6T0dCmtCghqokBZ89TpgvJhcNxvT4eCKngCw
  Et/yEbbUW+Jdibjm4+mCHfc+nXHcQ1ULK8JciYZtoQcs818fK8/tzqFb7nMbRSkC
  n+LaXp30mrLGP61uXcxsInND6CeKqn/jTKTS35oDx6atuqk49nnP8Oki5jPUPlxL
  vo/ezakEgWgj//4r76W7zDkUWqtugDLa3OST12kOUzOAH/zrDYQvrdyyYg+yym3B
  9d9c1vf3
  -----END CERTIFICATE-----
kubernetes::etcdclient_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEogIBAAKCAQEAulawtmyW9GPT6pYr6+3Md5yTthLQHtdJg3m9S+3U/A97dn/N
  jQo1BwVwbimbZIyLkCjIttcIT+bscWOgRb1Vnv2N3wmpvj8RHgso0aBpHSZ8OP2T
  Nn6FVs0oeJT9wCyO1xzSJmQe7fzZNIeYL8qcpGqUoPiRgk90GdTtpy8Q/JDBB4uK
  CmazBEyo4rquIDId6mWEaxENs1F7e/Ws0rE+ZaDFG7muXJIEZMhQV6QhkeAVzTUS
  SJn2G7dpzJXUNO2miEXy7pkP3yEtWcvT9bihLvfVfaEN+7x3eh1hFwWkjehrtbMl
  uiy2RewM9nkK7bITuzKPiVJQwOvkwAxzA1PsZQIDAQABAoIBACr9dSehRfJ8VZf5
  rYhxBItUBIAtNvbmlH9QxuuNo2XD7KgaNEIl6LYF+zpvPvqjsk9AGY6VbcWBcWt3
  oLJ2TBqyRNkAUbkis8BKstAqQhYHmwFK/3vDFOBB1OR8+04q+YmHjLRuxPQDMdl8
  rw/XnuwP6tp4fpmh2xdamLnJkhGyrki0ah+YqO9AuaZBZN6gI5rd4nCw7WwO3oCD
  awlTaagtat/1V4CkR21zNQiARURQAm/tcdI9q5rlbFplIUXS4a2YgPAaRXipTldw
  b+i+o/k3eSRkS/AHgsaRCm5RVqromWgCB0YBBGpGlGoTMOevahuzj8KueVwJrwgB
  QuKE1gECgYEAwPybeHBVkRGSbnGAqnEAAreZI+0Ajj5Fg/yH0h5dPwjNWwrb2bmf
  pguFiSnjpUsiaCr5C9DxoNnBggJq2Fp5IBZlcEvTuRdZZpHZqLuBmaG6maSExBZg
  IeUqt2dm1/eoM7oGswEDeTMzidvbcSheGZVKDn9d/w6GJJ28ujYjyZ0CgYEA9y5h
  QcEHgbZsma5DTrkiWzcbhnFGcFZ/dkoHh5fo8RRgOvlLayfCSGBlI+YlBuY2HT7Q
  /Yh2QMuPYffswBHrUrHoZCY8+3AHlxcRvVuPZfZs30U+JUBTE4PRDe3/sl+KpJp/
  fbG02KYswAuWXxKHauzJssdXzmdOiWI9Hchzt2kCgYBRzQENXPg4BG3AGKZEGJ+7
  hx0HaFca8/Q9TNY5TxuRM2bwFzs3H4I3PJz+ld3jW2SbKNPlmUxCNOrb87BcQoUi
  /7tBjTKSOv5vBpVu+wOlHjNdGqoX/7ABzgR8Nv0Vv+jw/AII7/4L6pMG9UUfws+Y
  InqAiKZMDVj4vk+X0oj61QKBgAEZ78z/My1cxrcYk6wyHvkREcpTjuDJQeAhDE+K
  WbtZP+SDX9amYrM32ruCNwAE3pDaysuWZBB917G0DRX9/nJr4IExumvUX/RIYgfZ
  9JWbt3h3MoOv+a7Ik5HgUZdV3aRGG/NPa2Lbuq9QXecmifvr+ioVwUpmI29xRJxx
  XDKhAoGAYaqSiubgge3otBuOeFJ6n1kSKBZTPkIjd0FOJ9TUxvBKaysvv9ipaMWB
  p192MzaeRX8tEYJYfL9qWJWyI58xCzARrNzVe3RxG3tbzKZCX1auDplDmrgW13+K
  WeWheuJs0nnoGMf7l9o+/6hXnF10qp66NgydKoJXj28O62+YEu0=
  -----END RSA PRIVATE KEY-----

kubernetes::kubernetes_ca_crt: |
  -----BEGIN CERTIFICATE-----
  MIIC+jCCAeKgAwIBAgIUOjJoKSotS5AZ87wpwRJzIqpBtmYwDQYJKoZIhvcNAQEL
  BQAwFTETMBEGA1UEAxMKa3ViZXJuZXRlczAeFw0yMzA0MDMwODA5MDBaFw0yODA0
  MDEwODA5MDBaMBUxEzARBgNVBAMTCmt1YmVybmV0ZXMwggEiMA0GCSqGSIb3DQEB
  AQUAA4IBDwAwggEKAoIBAQDEocUf7ZoSiuyj4clRmoMpCdLO7tWte+jMqbz/s9FD
  9iEw51vsuj5/HZTLPgBDZm6rqc+2Lku77PPe+iMfbXOkEQesWJWNGuYN/DY9Ka2u
  t8Qs1oU8H24PQaoVOMDnAotezEyda1rXSVRnOwJ5meNex168oP//IN/aSxXewbrg
  EwD+TFBpu5OgfvJCZkpCqjn6RNApgIe/EOVD2CG0SsNwSu1KpWtni68fvLcryF15
  2TiB3rPehOoqfApfU8tqqvZGWeisrTvyZq+dCy1tMeeiOGmWEcSIEj1WUg388NGP
  uGsagQsdIO8/2zdAl6mrB4mxKnIJoP8FZH0GJ5Q9LScrAgMBAAGjQjBAMA4GA1Ud
  DwEB/wQEAwIBBjAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQWBBTFuouEm6qjHcit
  pNB5uEk+dNAt7zANBgkqhkiG9w0BAQsFAAOCAQEAVeUmfsN2o550Wo9pF6zVob4V
  LvWW5urf20EJe3SZOcvUoSbmGFwHAqvTXU+0uhZG21Bksc5/XrbE41VwejePbla9
  I1XbnjYAdgpSx361i35kbeYAxfk/Y4YuzkmMCyYDdNghm3qU83LZuK8+qqqwYhH1
  BwGQwJDBNeQgZcwvreDNMeAWRvS1vSBd1uxdSo2DORLDvvRt5x2rhgT+zDr+POVM
  mCn0MJRzgncrcTHgmalG/zUfVG0mdzYES+v9tfLEd61IiS3Ct4kpTJRDF2CkBCZb
  N9wrdd8wZP8JRZqZY/qsjVPqmXYCZqQdRZbcFmjdUK9xjvq3xhI2yNqBI7Nz8Q==
  -----END CERTIFICATE-----
kubernetes::kubernetes_ca_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpgIBAAKCAQEAxKHFH+2aEorso+HJUZqDKQnSzu7VrXvozKm8/7PRQ/YhMOdb
  7Lo+fx2Uyz4AQ2Zuq6nPti5Lu+zz3vojH21zpBEHrFiVjRrmDfw2PSmtrrfELNaF
  PB9uD0GqFTjA5wKLXsxMnWta10lUZzsCeZnjXsdevKD//yDf2ksV3sG64BMA/kxQ
  abuToH7yQmZKQqo5+kTQKYCHvxDlQ9ghtErDcErtSqVrZ4uvH7y3K8hdedk4gd6z
  3oTqKnwKX1PLaqr2RlnorK078mavnQstbTHnojhplhHEiBI9VlIN/PDRj7hrGoEL
  HSDvP9s3QJepqweJsSpyCaD/BWR9BieUPS0nKwIDAQABAoIBAQCD1L3YGTIaSoA7
  o+6YwpI7WnW2/ZPPW2sKhKbNfR7JhORd6E/OWFP8X7XZyfjdN3jtqM01JLbsCQK8
  NVKFCJnmnvBZEY866pThPX6T5TSoFlb1hOIRHDVrDyhcUA+tEhyk8y5OwKJIAscO
  6xZOtuklK2AN8ZeQ0taigYLpNVP4VDhWr4I3njCnXMvbw6aD8kmK9qoJNoq/itgP
  NFa5Xswy4G5f+XVQpsLI2AUmgGtudRWvy/oGCz9cmdyvgAYwOZxn31Ic7QssUoq/
  TBfrs3B6dVIDttIy4O26dNfnjcaJQJKP3UbYAUPyIcbfg+QQoGRqa+EnG4+2si8s
  duhiRtTpAoGBAOmoSi43O5STYms58a+pEXJ2xvs558J+L2a4Qftbgd5f2l5Z6tvf
  SwZcvJewXpxquAXYhuW5B2q41YI1f2v+i2dqf1DlXP2mot0zcamXxxTQPowppHfh
  TWMTWDqRsfbl1R7reItp6rVSFh1/iMIpmnjDXx+eR+vsAvDbkrz5eoAlAoGBANdv
  Ic5B8ufw6ONXTJ3dc1fAvGfi9MvoEtPb4DmrG1PD2Y7H2knuXPdrJEHpiCrnzKb3
  SeSos6sDewCSlOLEoZK6/tCYRXVXHzlPnZAd2mI49842Arna0exVGpczxQ5BRtoa
  X80bKFA/MwfKQl6/pFnr3TAP3CAtdvouB7NcqYEPAoGBAJDnSBG/ORjcCiYbSjI4
  9FP7K01zzLHGVZkrXegHCNQit6bZtpfis4Ffk6myvAne+P7PGGFwzRO3Stm0WbBc
  rJFW71v40iGP7OKnW67kLJdQyelcjd22gKqXvZic8DQAtpm1SA2VgvQt0v86L1LS
  QDTyXJScMVgzBmPel/LP+NNtAoGBAL5efcj9jeBXifQa2KFHZv1MWzCR5S+qS7ja
  uzE6elNy5XS+Na7O62fXTZrQ+nqvirCgJLPiP6IG2VKQNPCOQRigFTvO0tH1wqmq
  KNmJB8lz/cZDA/Wn34ivS2gBX0hRdoKUe8OLE/yVXEqt+CgxwNGVE0e9PA9THafF
  tRhR8ZcRAoGBANckrlOrN7PL6v/S7yJFcKoi1Az+d+PiRwOft/B9JUl2uAAEIuIP
  LivKo53re/ZdsBqgBoF+HWdVotyc03dCCvbOSqnWfFzVWD3xzkYrXcGjIu6gG1Ub
  Eb6MzOK5TqijELvJeMDD7k09yhcAbGGAurtLjNXJ7Ll63+4nf2SzFrGv
  -----END RSA PRIVATE KEY-----
kubernetes::discovery_token_hash: 2020f1577c030a0ab03b2305ba3daabc440c8363c8dd25ab5af7841836188f21



kubernetes::kubernetes_front_proxy_ca_crt: |
  -----BEGIN CERTIFICATE-----
  MIIDAjCCAeqgAwIBAgIUJ4qesFnxGi/+wrt2uV2eWbe9f/YwDQYJKoZIhvcNAQEL
  BQAwGTEXMBUGA1UEAxMOZnJvbnQtcHJveHktY2EwHhcNMjMwNDAzMDgwOTAwWhcN
  MjgwNDAxMDgwOTAwWjAZMRcwFQYDVQQDEw5mcm9udC1wcm94eS1jYTCCASIwDQYJ
  KoZIhvcNAQEBBQADggEPADCCAQoCggEBAMFtTvpyFYuwaOPuwqxkTWHsjcrgiBWs
  gLsQ8NjKjZuY8fNoHxE9k8A97M0LY7KaekRJ3IPPz4qssC9QpRFoJd25N96BTSdK
  +Jk1YAn/ljBIdcuQ7YIPsNxUd1z57rd9i4xN5UVKMYsRqaLJxwTzHuuXmbuNt/+p
  dE/KvZo0ZCtrduxMVwY1VtpgzjlRWUcAzl8Mlc4xY1EKuX4lP+JvlGX9zfp1t16S
  99mIOwPm87iZn5Xay7unkR0nEfPWaKQUF7S7NYqqcMVAwXNc33doWI9t+M692w9Z
  3CGvenLRNqwAu5cI0p4Ep7dI9RIUomarPNB8Kpd2iFAGAAziWtCI1ocCAwEAAaNC
  MEAwDgYDVR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFL5K
  YIdiTiPC63kjZYIxXeA7ARp8MA0GCSqGSIb3DQEBCwUAA4IBAQAiQXFUbVH4Cm4v
  xZq4J6W1saVXs8Jx7AjijiOuLE1R3H4whkVwMQp7jAlwfQDzxKhCOQvjHBLZtgak
  F1sM+LoMGgU7/dTeWKaBUh/gxOGph2Nwze+hcvd1LDo1ZVB3pRK5VuWZHBQRb+4z
  s8zzVPLmGRckHYAjVfblqQcWKtu7TR4BPTZxbLTrbyODARRSSIBlf2IJsunjN59Y
  XNcKByk6sTUTRFkQUPX+0K6lzZv1KUEmIA/17QZGfpO8yUKat4sFB4t3C4PUtGef
  FACixT/tGhURWROIQ0wuC8GMh4MU57ZF9nnYcol0CiL8AziXYJeFIAKsx+xjLC2t
  Ezs0Mo1j
  -----END CERTIFICATE-----
kubernetes::kubernetes_front_proxy_ca_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpAIBAAKCAQEAwW1O+nIVi7Bo4+7CrGRNYeyNyuCIFayAuxDw2MqNm5jx82gf
  ET2TwD3szQtjspp6REncg8/PiqywL1ClEWgl3bk33oFNJ0r4mTVgCf+WMEh1y5Dt
  gg+w3FR3XPnut32LjE3lRUoxixGposnHBPMe65eZu423/6l0T8q9mjRkK2t27ExX
  BjVW2mDOOVFZRwDOXwyVzjFjUQq5fiU/4m+UZf3N+nW3XpL32Yg7A+bzuJmfldrL
  u6eRHScR89ZopBQXtLs1iqpwxUDBc1zfd2hYj234zr3bD1ncIa96ctE2rAC7lwjS
  ngSnt0j1EhSiZqs80Hwql3aIUAYADOJa0IjWhwIDAQABAoIBAHoltxw88xI1oudd
  nZ4T8TIHmjsls4pMVzbKqe2da/N4kEIv3H6crjNWZ7XMnXbqSo/ZPOIYV22xQ8l9
  WeA2LsNn9boTWU+Y7oiBxlZKCuQOviSBLxtVIVDiHkaepntFUtyVTFWHVuYFj2R3
  m7CqfzozkCd6hVlz/zlsRHOe5irsJgUsNByQId68uP++9FwLBwSfl0P0QNTsdEex
  LeLxar+dP5o4LhB2WnyiZl2MGRLd/BVTFqbsfm/GaakdHLAv4fU+QEtj4X7C1g2n
  ipRtMaNsibF8NP565fmoiTTVcHjepOJFeQfvCy5GlAZ0NW7CXr1JRNrPc2yfurwM
  pX5wiSECgYEAyy5k2y5MYSZxMGr6A29hkc5ltZY2k3Jo4An1241XmzE9C+0DOnik
  9x7g9PpxGnJj4DQr3E0Lbe73u93mrHEs9VvL2fC6PlCPEIY5vnoNORPExXu/1OPI
  lmomOhOkvFHfSfU9a6K0PQ7O6MKJZHop1yKq3EFVpmc3bk8S8YAQEVkCgYEA87XG
  XsDvVziA6DEjj4YMmt+2mbrI6d3udSiWVh3yUSjtDV9b8hAmhYd+EiO4yUKBHsf8
  rUhQRJsoom6fLJvulLBM0FCQFCeLXz9r5UYAdyelaG8mad3ToRznV6u/kKSvjg5A
  dtIXpmp184lz4rEP+trJj4hqgLhBMZSXvkL2St8CgYBDa9K11Igt2KbIrUbnKueS
  jY28kurrRJS+ey+lOiKu8cjTE0P1/CCpjAqT4CL15q1zsXw4byxyv1Bfe8PWxvtn
  M53SZQo4MV0324J/zVpj7UZnTagbSba/CtxCe1A32Wch0IogFG6AsFBCdoNG3y7R
  H8TYr6lvPWLsbLrZfmhWiQKBgQDwVG1Ou4xKwtm8H9kli9eaUsrLRrsihUzGFERM
  oDN5fZZm/Ya6atrSCw9z6+4p6n9ST30JlKozPZ8Qr24gm5Tm5ASb2RZ8CZMTVoOc
  ETAcUOh42ENVWv97fflVJ3U9umm0+LDxe7T+0zWF+CJjhqBvjJoABVKhpRIFRrdt
  VLN0mQKBgQCiLBncc6UcKZiET48d2tsL70HGF/jk2UAH9IAUyo1wMAN1bKudtjYc
  9sbxFA/5VHGRK/PmouxHYmLIWmLDqPu0gddmwFBiwxBi3AOUodB1z1kEmhaHbKcK
  MVUBmjk7O70PL6lUy604BUg/JbDlckEjYw6GmSICjnNPAPy8qSgmXw==
  -----END RSA PRIVATE KEY-----

kubernetes::sa_pub: |
  -----BEGIN PUBLIC KEY-----
  MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxvGjKX5VvlsBJInT+Z3D
  XDRfi11WlRHKtVqVBmpQheNyrVe2Kcq2SWHQ7jw4iaVqGrgfkdIE31etYpZJerun
  o0qBTKJ9TCypP2Lzm37xERBWQw60Pktf/vDu79jASgZcp/jhpSFxc4+IZ/aUHzUd
  gPhBu44/9wvinWMw/0mScr2KpDQhJ/B5vRejOHFrQLfVMMNKpmovfysq7f4LmIae
  FR84s9cN1MX3K3ZNa2yL06lE9DAmymvzOu/94JPKGAQXC/CI5B8ytX4dOzgtgYz8
  5BNei6H3zEiotWSCgvPXh1H5JAXXR/LsWpFvkIQSsW6crJwKlVsSNJ9uar3E90Ko
  rwIDAQAB
  -----END PUBLIC KEY-----
kubernetes::sa_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpAIBAAKCAQEAxvGjKX5VvlsBJInT+Z3DXDRfi11WlRHKtVqVBmpQheNyrVe2
  Kcq2SWHQ7jw4iaVqGrgfkdIE31etYpZJeruno0qBTKJ9TCypP2Lzm37xERBWQw60
  Pktf/vDu79jASgZcp/jhpSFxc4+IZ/aUHzUdgPhBu44/9wvinWMw/0mScr2KpDQh
  J/B5vRejOHFrQLfVMMNKpmovfysq7f4LmIaeFR84s9cN1MX3K3ZNa2yL06lE9DAm
  ymvzOu/94JPKGAQXC/CI5B8ytX4dOzgtgYz85BNei6H3zEiotWSCgvPXh1H5JAXX
  R/LsWpFvkIQSsW6crJwKlVsSNJ9uar3E90KorwIDAQABAoIBAEFACcrXM9o+UO6i
  AhNx6hx41QmJsJcuAbVfo1c6UXTId9lg1k94vDbvTS6GM+j5hwaCc2BJaO2z/RtD
  vY/ysnX9PtFazgvI7i3rD4llpj15hUFzNizkXfTpIwUpq0/cLFGvcJrPkD1dy1iF
  6Gkf67BcPHr+mf42OLp0SSLCB7V33xv9LFAfYv1jQJzBrRffSXyx0nCgTS2Sl6Py
  yLPsPLoWOdYpS7T8zCicdWIYtrKaIWWfl7TUGq8/ELDtrtE0rA7mMkWUwgNhosJ9
  SpjpoK03a5pyfO1FzHFZmj2K55jPBsxUTV8H+T4oxsERR4ADXsTuGC+DL0AkEKey
  tTXG7FkCgYEA7hoo0+brfJocJdaHH01QxVyy0C15fSA43F0AgtZFlQzshVgdRAnm
  I/MPwVnIm+9ij0SbkAuWjly4iz8c3TR9YySNN+Xjhni2kLCAS+MKmmXw0+j7wcXI
  0IR5tZaB4hC1OdGHzWDpAAdCa2NgkqZiSnwL/gijKUZUPdVHW9QbSnMCgYEA1eXy
  w5wJj5HV3ew7SOU2qxJ3vwgQDrVp/EN6B+wzbTaD/1mLb+TzXgL6XvjUyxY6TL13
  oeBEOFJHsYOmZo9gxF5EnGuQcHXzUM0UKz4oIzfUnkOh2P1CvS2R7ESx1EMqqyeg
  wlhCt6u9APyCwmyym4a+RPV76VHqPCapbuJKrdUCgYAVfk8JMeM4EEFSORdhf/02
  k7OgjzpUTUBUxCBzrxwA5HXFY7rglXQs8pgNgClXyjg/bpYMXr9sgno2N7BHV6Gp
  /XayARKU5Pi1/vjV4NzoKyOePX3cdBo4mHzUI4399jwlyABWTdU6p20bfJEPK+3b
  g28hs23nH+4i1snmRDNieQKBgQCkiJlQ9no0Qc3tQ4Xm8Mo4laAfBiiz58B7F7yK
  05pLbB/lEBA+JjM+IbRN0cJ95b86wJOBpW1UlZYoakhZo/iflJPabrNQjqp3gQj+
  qnVdgdi/VO7uZwIrWx5gTAcua+wqc/UpCZNojWZ5l8NNoydQkzlQ+jQyqZwwYNz8
  On2PcQKBgQDOrI7cfW7T1pcG/vjKdmVYeJJssW6oKpZYZp7m0D1nD+FmEG8jNqwu
  Vs7Z2RVcVtv/mryKApfSfu0ojoEmwPZGKGZ2li08pZ78fc0Jreo5okaN+NWsm4e7
  s9AbQqv1G2sWehboT9LL3t8rCnnFdw5ki9LjAk+ggpKN7JIPJPf+Lg==
  -----END RSA PRIVATE KEY-----
```

C. Kubernetes – CONTROL YAML FILE

```
---
kubernetes::etcdserver_crt: |
  -----BEGIN CERTIFICATE-----
  MIIDTzCCAjegAwIBAgIUFGWRROlhxh+Thy5GQ13fe417z0UwDQYJKoZIhvcNAQEL
  BQAwDzENMAsGA1UEAxMEZXRjZDAeFw0yMzA0MDMwODA5MDBaFw0yODA0MDEwODA5
  MDBaMBcxFTATBgNVBAMTDGV0Y2QtY29udHJvbDCCASIwDQYJKoZIhvcNAQEBBQAD
  ggEPADCCAQoCggEBAO65+42cm3gX6O5dlE5Td+mqw2n3w/M2gvNyg7x/mV8KCQ+t
  BPEX43TG06YR1CsSbB17YdJXkGwG1jWdZvGeQIDhJogkVytm+4RA/RrRvTBQlcT0
  SeeyoU4tHTbdsFTLeulyz8nKdV/SDz8STXtUImHKe17XRnk6n59CfCtzAjTetNSE
  m4tLZF1KwSgjVlkrO8aonxQKB97WNzR3dNdfxvHdw4Okn3oge7xR/5tWDwtGXb0L
  J4z9vgdYAR0i1Yl35FccIIfv66oq4niZjgtO7cX9zckgQXMfbWmbrm2RJSz/CAie
  3inYNuJdATKxizLtfqrm6z4+p82eAsMchOJJL6MCAwEAAaOBmjCBlzAOBgNVHQ8B
  Af8EBAMCBaAwHQYDVR0lBBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMAwGA1UdEwEB
  /wQCMAAwHQYDVR0OBBYEFPRUq1aRptYQhG4vTts9Sh+l92PwMB8GA1UdIwQYMBaA
  FH/z/Dr4xI5wf11se8O3LlCpeQ1HMBgGA1UdEQQRMA+CB2NvbnRyb2yHBAoARxow
  DQYJKoZIhvcNAQELBQADggEBABRuQ6TrCH8Zjzjooqjy69rx2/Xpor4WBSK1pYmN
  lDLgRJ1j25I8cw8gU/e09uc5MZTpnFKzxd+hu3vwuGB8uIENbLR/WqlgMxALV5Iw
  rWp0oMB3/d7dLMbhKm0tMLQIIT7tME6bOO1e8pbrnf7SCQ1Rsf2FtOjgM/D2xWbB
  bT6FQk5KLrO9407QJcTEF0KThQIlxe8uXBZgeJFp/R9NVIL4Ic9xbc7IGfgiLSJB
  it6DHuBfI7yrjMjcCaGyZTJ45PPaM6nrix6/EtceQW1Z+yjtaYdiE8E7GFf621fQ
  hcaDgXBUvUfchPDM0jRIyAxP4kn1EEFTyxa95HLGVTBWZfE=
  -----END CERTIFICATE-----
kubernetes::etcdserver_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpAIBAAKCAQEA7rn7jZybeBfo7l2UTlN36arDaffD8zaC83KDvH+ZXwoJD60E
  8RfjdMbTphHUKxJsHXth0leQbAbWNZ1m8Z5AgOEmiCRXK2b7hED9GtG9MFCVxPRJ
  57KhTi0dNt2wVMt66XLPycp1X9IPPxJNe1QiYcp7XtdGeTqfn0J8K3MCNN601ISb
  i0tkXUrBKCNWWSs7xqifFAoH3tY3NHd011/G8d3Dg6SfeiB7vFH/m1YPC0ZdvQsn
  jP2+B1gBHSLViXfkVxwgh+/rqirieJmOC07txf3NySBBcx9taZuubZElLP8ICJ7e
  Kdg24l0BMrGLMu1+qubrPj6nzZ4CwxyE4kkvowIDAQABAoIBAA+GXMua0al/tJA7
  2bc2SmgRyN+NU1rvwphebB7IFYtOtcQlNlsPeKVRgNYd9rROYGWSbowlUakU0L1v
  3Q/9gKhg/AaChoC2E8lICK3D0g/gTb0fPecpgL/6wxmcVSlOilHAO68d0YBHwuIR
  KKTKlenSJy/FxDmCDHto8XZJ5mzVprYARvwQBFnWl067uNMHqNMhQznDuR8tEenR
  fYs9KQ1aZp5qVFy+eohj0Ig2/pJ0YALAt8+UeTXUDhtpsHjsh/lVqIWWgIPTfcqf
  1s+MQaPXO0cHFkKh5qVj/3LQWbSTnBxHBk+0lnfECTVBN7+3ZZOl0btvOuG+0S50
  DqPJZHECgYEA/oMz1qnJISItnwBD0d0552AJMrU25Ho2KtE2JtuHHqpGVgJnUTpE
  MDHZZS+5s193MSHnccZrYkq4bwwlZ5XyEIRqRvQl4FJuyzlda0Q8RnEkusZYmHLx
  U+rWzLFmcieoFDhP1JKu+2rAYajjZCXKdTr7FeXh8pHyr3MNmHpbIM8CgYEA8B8p
  TofzZsxqjKH8IqbrsTcLo+324lkUwCvKSu/Kdt/zczjdCKD/DnZEpqwHlpGQUUxo
  3zP+jJy2ZkAX/EUm3qSWG7Hhdrhwq9e0LljZKNPGRD/I53xi4W+2xqgd7yaoPXYF
  mhCPVWT2EAYdnoakGbZawHy0Q8U83nf81o0rMO0CgYBXFV1vXbjL9X3WoaiS7jcZ
  y+pZx1d45bgS/nKg6QxambhnRXtEd8NNGkFgNew8S7Xkwc0HSPUFNLvWgzlpMA0b
  SbNzg6ZSEuKF9qLYSQi2sY/3uFYoE2wvYnMMpsZ+2MLR6FKoUWg97lDVGYx0f9m+
  MugmnfmpnfaDwcItIfB8UQKBgQDgmBT3yjuFFIA4qC6AbuaXbGgr3aGFl1LMaGkf
  bGhQhipcxHwh3QyUB1UxHElAsUhucmzJEQXvmYkV3K9Sm4++we1RXDEJ7Xwdj3WN
  wDbmdCbHoW2V74dBW093Qro0/VBxpFmbABBBDF0HIeFEbKVDE4iQ8FNf7DWB1HMe
  bw5OBQKBgQD6AZla2Wvmk+O2bPsqnnDHE2wE1VeVWRN4dbZjoQTQGPSERLv9o8ps
  tmXEQLEJ1VY6vDgAV1ba766zNxyykp2ODZdEM02vKxPeYEo7Su/a/j+SozkJEtm6
  NHPbgxVoe8mYTfToWIvwIUrv3YLNqqbXz8xI57AnB285w6e/D/pIBg==
  -----END RSA PRIVATE KEY-----
kubernetes::etcdpeer_crt: |
  -----BEGIN CERTIFICATE-----
  MIIDTzCCAjegAwIBAgIUb2KG9QhQ2mQG4k3sq7c6GzOLNnYwDQYJKoZIhvcNAQEL
  BQAwDzENMAsGA1UEAxMEZXRjZDAeFw0yMzA0MDMwODA5MDBaFw0yODA0MDEwODA5
  MDBaMBcxFTATBgNVBAMTDGV0Y2QtY29udHJvbDCCASIwDQYJKoZIhvcNAQEBBQAD
  ggEPADCCAQoCggEBALrN2wuc5/6vz+KYYH/cRkwhoRUjYrqmFiITXYbfwR7JK5Dv
  EfPLyOup5Mq+IR+G24w7fSvsZA14yReV3S9dvt8yRBU2ySoaFA1fQgNHvkKPZ05z
  MhaOVF2rmxQZd20PXHBQQYkiM7/iwczgXaFie9jYiRzz+AQBY7MexPxZwOBiXrWm
  A/E4c5qvmEo8f6SuOa9oPG2JwclTN45s9MPutybbclVWCNblR9mGfHGDTnToN8tA
  1CZprUTiDwBiBz1B09eUJLFg31tJtnP2qr/AoU5Qz0h3j0asZ4lyESpQ2SWdJUMy
  2rUiwVAjoP5plNROBuiethPp/aqm4WJdDL0WFr8CAwEAAaOBmjCBlzAOBgNVHQ8B
  Af8EBAMCBaAwHQYDVR0lBBYwFAYIKwYBBQUHAwEGCCsGAQUFBwMCMAwGA1UdEwEB
  /wQCMAAwHQYDVR0OBBYEFEuiALWBehiH7u0btbyuOHR06hfQMB8GA1UdIwQYMBaA
  FH/z/Dr4xI5wf11se8O3LlCpeQ1HMBgGA1UdEQQRMA+CB2NvbnRyb2yHBAoARxow
  DQYJKoZIhvcNAQELBQADggEBAG1MTZuo2/isxlGbt5N7zyej64EyVC5GT4q1WBpy
  g7sPY1qPNa2LKisZHyUPwoSOsq9JZzZy/4hKE8VJV2AhvKfor24Le5qlbZIAPiu/
  itya7lk5zSHPcwdleiCu6I/NgsqV0iFJ5EehhJMPc93QSSax9edCZD8bnPLuDbEO
  nzRoAyoqGeyF1w3l31AXz4FXPUEx7mORQE579qRN3LB8Y0beC2iOmuTWFFHZcyL8
  5N2MSB873OGiItpF6c2Q/b1q3O5rcpInBEhXba/bI5KUpUjfNAkcMyvYKj+wA11o
  VKcf6rlhC0pvXyua2YtE9DHDG5F1KyRKcgql5swV5VIVoeY=
  -----END CERTIFICATE-----
kubernetes::etcdpeer_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpAIBAAKCAQEAus3bC5zn/q/P4phgf9xGTCGhFSNiuqYWIhNdht/BHskrkO8R
  88vI66nkyr4hH4bbjDt9K+xkDXjJF5XdL12+3zJEFTbJKhoUDV9CA0e+Qo9nTnMy
  Fo5UXaubFBl3bQ9ccFBBiSIzv+LBzOBdoWJ72NiJHPP4BAFjsx7E/FnA4GJetaYD
  8Thzmq+YSjx/pK45r2g8bYnByVM3jmz0w+63JttyVVYI1uVH2YZ8cYNOdOg3y0DU
  JmmtROIPAGIHPUHT15QksWDfW0m2c/aqv8ChTlDPSHePRqxniXIRKlDZJZ0lQzLa
  tSLBUCOg/mmU1E4G6J62E+n9qqbhYl0MvRYWvwIDAQABAoIBADBIQsSGoqP+dyHf
  Npn1qinFS2g2RC9v/CqEjNjufnDhmCQW1rIHyv+2Ys9QQylt5tMhESJIMDgXLwqs
  joXOmiYATP6asXXzVZl7NIm6gl+bxxzMw3Z3BTFMyIFyb569qKvS4o16TJvdFTBK
  EYlD9jS/CvlzyXfnkAhc/Jco76RO9yhKrb4vP2MhU7Xxs4eUmil1ufshPTt+4PLU
  TIZ/4QA5VWyS02jdnKskG31RllBaWknegYfS0xuSyPzb9q+XDDjEWr+ca09T3bHe
  SBS3sqCkLDPwHFFkw0nHbKm6oCm0BKFsYXftwVHjnV5BnjCvLYyfIhzkdyV1vAPL
  T+Of+XECgYEAyG3/QGqMb5dNKp/v6tj8P+dWdGCT45C97ssdLTzQ/2PY7GKYLBu/
  hWPgqQj1u/cJJPXkOi+qlRgnaclZpEuHl+kQjlkrI59LRlZ4OJsOystEhePt80Y5
  q5jhqZfChsXPAGS/Xb2Hzn0dRZuyjPNFB1NLAc5eBSBpHzV8RCiKw/kCgYEA7pjA
  I1hKRVs50m9aHYjJqWjMZ+dPe6SA9y/lKqRqB/r0DN0kLYezzJLs1R/2v+CI0E1s
  RIJ59AiM0GnX9L8iDGtHVr/WaqBdVRy5OKc+nF3rEg/qMWq40XPXJGCl9dEiotRe
  3EfSbL8m87JoA1NQgoK+ImKOGeknLCPZwr9gbncCgYEApLP3gJjOJNlhmThq8EM1
  LtJM8j926BdGbRfONyF14ZjMZpwyznRS8yt7L4zkWn0Lr6q8zK/k7rY1lTHYDzHe
  EBB/AWr3D0PUmM1v55xMEIvvGFOQc0xvEbsFHddPDG86xukDdOXN1vprdosWs8sp
  G3bv+WD7VB0pJXNPVbc3KQkCgYEAtCsEa6wv6U+Jyi3rjoHMRfCs6YjxrBv8hqvU
  EO82YfBOeahDuV3Pl5sa25TXbuzOYv0T1GfyTyRlsLs8J8RsoZQgupECCGIwGCUo
  eG9SCCGjcHA0GDzOcttWilGxAH57+RkEyLIzY4q0jzEY+enf0a2Ihc6GH2q/+cTY
  mwaIeuECgYAr2Py3e/alxxG+pU22opMnP+zRxOtp+yj60uV6+QLcL9CKlXeJ3+4K
  8cstCk8vjxWcPvItgwMMeDxQoZMW9lsygwdrrGV7/SMvKmP1U87LijNHY8sU9D9T
  FrEb70svSgL4K7dU0OUPgGnw7TihpWg9AWya3d0EAiWUROgLxGLJkA==
  -----END RSA PRIVATE KEY-----
kubernetes::etcdclient_crt: |
  -----BEGIN CERTIFICATE-----
  MIIDRTCCAi2gAwIBAgIUZJQeLsOM35b2Eno0OviOI6K8DUAwDQYJKoZIhvcNAQEL
  BQAwDzENMAsGA1UEAxMEZXRjZDAeFw0yMzA0MDMwODA5MDBaFw0yODA0MDEwODA5
  MDBaMBcxFTATBgNVBAMTDGV0Y2QtY29udHJvbDCCASIwDQYJKoZIhvcNAQEBBQAD
  ggEPADCCAQoCggEBAKnAuJozoVKvDj9x4lLs7gJ5Br1CR7Hozj/tT7UnxsMkt7YS
  DGEGO6EQpp8lHyqmdNuGd9iXHF4OjdAHALenE25WzGrLMONJLRq9EfLSQUl1gMbM
  lqnEOoGQViZWFJQY2kKq68jrDbUeGCFiKGYMVWIT7ph9IOQDDKuzAlu1WsB9S4gk
  f4rNgItUap04luh/J0hR6vv/UrSsXTEum/MUJKBYdDSd2WJB45P5AI0J2b9IcnPf
  2UFmJ6snnUrLDlFVZ+7AU29l2nosMGZtMCl+u8O9JixPewujFt7JrWeq0m7vnG98
  krkNJq4dMlUX5mrMI//tbEPM2uHRdZQBurtjQ70CAwEAAaOBkDCBjTAOBgNVHQ8B
  Af8EBAMCBaAwEwYDVR0lBAwwCgYIKwYBBQUHAwIwDAYDVR0TAQH/BAIwADAdBgNV
  HQ4EFgQUl8ORnTr+5mQsEQlTseTfrhASbQEwHwYDVR0jBBgwFoAUf/P8OvjEjnB/
  XWx7w7cuUKl5DUcwGAYDVR0RBBEwD4IHY29udHJvbIcECgBHGjANBgkqhkiG9w0B
  AQsFAAOCAQEAKShrVLjy2k/AkX5f7OfwY10RcL50qtqF1RKL+/uxeWxl49IHg2HO
  dGGB9qDMiUok27EYIiqPscGDSUU5y690qAxJo5MQDqtNVMqlXEgEcbE1CI40I1Tv
  NoTyFzlfpZuLuKgkC/NA554dm1/7bdn47X987Hvs9CjxdM2qejyI4WaqsMtrm0Ys
  uPP1hP0Si9F/c5nlg16nKlSqQ3G8jN3/F98Pc5fbpCXbcT5OfeNnjlKLw0B88A1x
  5Bh+8SX4RAKKb6rtewO3KWu442Bq7RKZLQtbBq8/6Qqp2beQefmVQy+TRLb4Alo9
  k76DortwUYFcOlE6lhYei5SjFqDx8RH2Ww==
  -----END CERTIFICATE-----
```

```
kubernetes::etcdclient_key: |
  -----BEGIN RSA PRIVATE KEY-----
  MIIEpAIBAAKCAQEAqcC4mjOhUq8OP3HiUuzuAnkGvUJHsejOP+1PtSfGwyS3thIM
  YQY7oRCmnyUfKqZ024Z32JccXg6N0AcAt6cTblbMassw40ktGr0R8tJBSXWAxsyW
  qcQ6gZBWJlYUlBjaQqrryOsNtR4YIWIoZgxVYhPumH0g5AMMq7MCW7VawH1LiCR/
  is2Ai1RqnTiW6H8nSFHq+/9StKxdMS6b8xQkoFh0NJ3ZYkHjk/kAjQnZv0hyc9/Z
  QWYnqyedSssOUVVn7sBTb2XaeiwwZm0wKX67w70mLE97C6MW3smtZ6rSbu+cb3yS
  uQ0mrh0yVRfmaswj/+1sQ8za4dF1lAG6u2NDvQIDAQABAoIBAE7aWiYvM5IABUNN
  eI+CxbDaXVoO7tTiyuQBLZR+DKNbbAckbSGozowr+upUSRz7w66x0PwIfcHnStLG
  XfvxnWIU6dH13xjFRjF0gGDmcDNaZN5z7MyhfXfv/0TBmb+Np91uHibvoSdv+k+P
  lCqLMiJkFeI5xcyE6h7cAY1uOeDN0VrrxMkWx6qY2/QFNQzqEqm6YRVVi+mqCX9P
  LaBiPQe+wHsw5BJQBn3Y1frarp2GoEvfnQkMD6op+31ZB0Kw9zC5FCcntYw2PQp9
  zMCavCRDNlAGEF4JyTuB3axxtcW/wX1r9YzTmx6LcowSDFI9b6jN06Qk+/EOyBGv
  L4L8ltUCgYEA0VFEtbPQe52qGn1Id28fr4O/8cyqwutWjwfbJ/upPVaNjw07cRrM
  5WGd5XEVYSbI59un4V4/5tMwfU906ip/OZr2wXBwq/kdOEIqzMmF2Seak6DRjP7c
  4s+HErrWfVF0Y73tqJ72Sv9Ys/e8jZWmQOVzJfW6okFL5kUL3h3Ns/8CgYEAz5yO
  ggsplNkTQ1gGlU1d4/Y1iHiPdRPD2sfZiZ2GWxuT4WmJPz2gOeCofQlU3qXB5kir
  MyQeJjfkPanQUjgsp0DsYqALc5x7h4dJcyfNp+WttsUB5DomFRHD6QmWbkv2dKY+
  gZ6zTXUTckjml0j2AZfP4nAjy615YdyGxElS2EMCgYEAr75NM0YrSG7jMGR6IJUz
  hFjZIvqVcRzIy0RsVFCvEuoF5tLUUq2O1RS1zru4mJ5I1qsNdGJ+wp4Uu1GoNyfB
  NCuQ/G8cmreJOgAMEW8uK1peJY9EAd7GtMRLg4VNq2BlmY0Q6p+06Nklr8zUCghY
  02oS2Q9NSQgL4uWcBjz9xh8CgYEAxFfvfpVH8fmJ+jqZn0HX9eu3nSVHpxXWOGah
  GUDc+/UM2jWREzoY+iO5tObKWPaydjNrlaYmzUY2mQqsddWihVslM/DgY6ouOXJ8
  yGdwW8UfkKaor0s/ENYITbA/kHhO0OsYWkXYlPyQM5k2WwUpg0Ar3p7ne4zIKwEA
  g0pFCzkCgYB6FxQzVbqkBgIDSN51N9tjCGT+N5Xvg3bTXkc9uee3IruMcpCxmnuA
  muBHSmyuGTgF3Vyf92uxILA+X6jxZsUa9AiV9iw5+fVHdvnveAgsiy3WkTtrjVaa
  AbU66kfwvgtcDUxI6WknqdzaLs4j1903FAXaEKF7mGhCOSOF6RDNig==
  -----END RSA PRIVATE KEY-----
```

D. Puppet – agent.pp

```
$master = "master.openstacklocal"
$master_ip = "10.0.48.123"
$interval = "600"

host { 'master.openstacklocal' :
     name => $master,
     ensure => present,host_aliases => "master",
     ip => $master_ip
}
augeas { "puppet_default" :
       context => "/files/etc/default/puppet",
       changes => ["set START yes"],
       notify => Service['puppet'],
       }
augeas { "puppet.conf" :
       context => "/files/etc//puppetlabs/puppet/puppet.conf",
       changes => [
                   "set agent/server $master",
           "set agent/runinterval 600",
       ],
       notify => Service['puppet'],
}
service { "puppet" :
         ensure => running,
    hasrestart => true,
    hasstatus => true,
    require => Augeas['puppet.conf','puppet_default'],
}
```

E. MLN configuration file

```
global {
    project 5GC
}

superclass agent {
    openstack {
      image Ubuntu-20.04-LTS
      flavor m1.large
      keypair baston
      user_data {
         apt-get update
         sleep 30
         wget https://apt.puppet.com/puppet6-release-focal.deb
         sudo apt install ./puppet6-release-focal.deb
```

```
          sudo apt-get update
          apt-get -y install puppet-agent  puppet-common augeas-tools facter
          echo "10.0.71.13 baston.openstacklocal baston" >> /etc/hosts
          sudo ufw disable && swapoff -a
          echo "net/bridge/bridge-nf-call-ip6tables = 1" >> /etc/ufw/sysctl.conf
          echo "net/bridge/bridge-nf-call-iptables = 1" >> /etc/ufw/sysctl.conf
          echo "net/bridge/bridge-nf-call-arptables = 1" >> /etc/ufw/sysctl.conf
          sudo sysctl -a
          wget https://git.cs.oslomet.no/s361748/acit4430-project1-files/-/raw/master/agent.pp --no-check-
certificate
          sed -i -e 's/10.0.48.123/10.0.71.13/g' agent.pp
          sed -i -e 's/master.openstacklocal/baston.openstacklocal/g' agent.pp
          /opt/puppetlabs/puppet/bin/puppet apply agent.pp
       }

    }
    network eth0 {
        net netsys_net
    }

}

host control {
    superclass agent
}

host worker0 {
    superclass agent
}

host worker1 {
    superclass agent
}

host worker2 {
    superclass agent
}
```

F. Kubernetes deploy script

```
#/bin/bash
source /root/.openstack
ERR="ERROR"
RUN="ACTIVE"
SCH="scheduling"
:>/root/thesis/k8-stack/deploy-p2.log
LOG="/root/thesis/k8-stack/deploy-p2.log"
echo "******************** BUILDING PROJECT ********************************" > $LOG
mln build -f kub.mln >> $LOG
sleep 5
vm=(control worker0 worker1 worker2)

for i in ${!vm[@]}; do
    echo "*******************Starting ${vm[$i]} ********************************" >> $LOG
    mln start -p Project2 -h ${vm[$i]} >> $LOG
    sleep 20
    Status=`nova list | grep ${vm[$i]} | awk -F'|' '{print $4}'| sed 's/ //g'`
    Status1=`nova list | grep ${vm[$i]} | awk -F'|' '{print $5}'| sed 's/ //g'`
    while [ "$ERR" == "$Status" -o "$SCH" == "$Status1" ]
    do
        echo "*******************Deleting ${vm[$i]} ERROR STATE in NOVA ******************" >> $LOG
        nova delete ${vm[$i]}.Project2 >> $LOG
        sleep 10
        echo "*******************Starting again ${vm[$i]} ********************************" >> $LOG
        mln start -p Project2 -h ${vm[$i]}  >> $LOG
        sleep 80
        Status=`nova list | grep ${vm[$i]} | awk -F'|' '{print $4}'| sed 's/ //g'`
        Status1=`nova list | grep ${vm[$i]} | awk -F'|' '{print $5}'| sed 's/ //g'`
    done

    if [[ $RUN == $Status ]]
    then
        echo ${vm[$i]} "VM is up and running"
    else
```

```
        echo ${vm[$i]} "VM is either coming up or in weird state - please check the Openstack HOrizon and
if required undelopy everthing"
    fi
#         Get IP of control node
    if [[ ${vm[$i]} == control ]]
        then
                IP_control=`nova list | grep ${vm[$i]} | awk -F'|' '{print $7}'| sed 's/ //g' | awk -F'='
'{print $2}'`
        fi

done

# Generate configuration files for Kubernetes installation
cd /root/kubernetes/
:>/root/kubernetes/env_final
sed       -r       's/(\b[0-9]{1,3}\.){3}[0-9]{1,3}\b/'$IP_control'/g'       /root/kubernetes/env       >>
/root/kubernetes/env_final
cp /root/kubernetes/env_final /root/kubernetes/env
docker run --rm -v $(pwd):/mnt --env-file /root/kubernetes/env puppet/kubetool:6.0.0
cp                        /root/kubernetes/control.yaml                       /root/kubernetes/Debian.yaml
/etc/puppetlabs/code/environments/production/data/

# Register the nodes into puppet
count=`puppetserver ca list | wc -l`

while [ $count -ne 5 ]
do
        echo 'Waiting for node to come up'
        sleep 30
    count=`puppetserver ca list | wc -l`
done

for i in ${!vm[@]}; do
    puppetserver ca sign --certname  ${vm[$i]}
    sleep 60
done

echo "COMPLETED - Wait for the Foreman GUI to show all the nodes"
```

G. Kubernetes undeploy script

```
#/bin/bash
source /home/ubuntu/.openstack
hammer auth login basic -u admin -p oslomet@2022

echo "******************* STOPPING SERVERS ********************************"
echo " BE PATIENT "

vm=(control worker0 worker1 worker2)
for i in ${!vm[@]}; do
    echo "******************* STOPPING ${vm[$i]} ********************************"
    mln stop -p Project2 -h ${vm[$i]}
    Status=`nova list | grep ${vm[$i]} | awk -F'|' '{print $4}'| sed 's/ //g'`
    sleep 10
    echo "******************* Deleting ${vm[$i]} from puppet master ********************************"
    find /etc/puppetlabs/puppet/ssl -name ${vm[$i]}".pem" -delete
    echo "******************* Deleting ${vm[$i]} from foreman ********************************"
    hammer host delete --name ${vm[$i]}
done
sleep 10
echo "******************* Deleting MLN Project ********************************"
mln remove -p Project2
sleep 300
echo " Operation Completed"
echo "Make sure all instances associated with project-2 are removed from Horizon"
```

H. OAI 5G – CHART.YAML

```
apiVersion: v2
name: oai-5g-basic
type: application
icon: http://www.openairinterface.org/wp-content/uploads/2015/06/cropped-oai_final_logo.png
version: v1.5.0
description: OAI 5G Release 16 Core Network
```

```
appVersion: master-v1.5.0

keywords:
  - 5GCN
  - AMF
  - SMF
  - NRF
  - UPF
  - UDM
  - UDR
  - AUSF

maintainers:
  - name:  OPENAIRINTERFACE
    email: contact@openairinterface.org

dependencies:
- condition: mysql.enabled
  name: mysql
  repository: "file://../mysql"
  version: 8.0.31
- condition: oai-nrf.enabled
  name: oai-nrf
  repository: "file://../oai-nrf"
  version: v1.5.0
- condition: oai-udr.enabled
  name: oai-udr
  repository: "file://../oai-udr"
  version: v1.5.0
- condition: oai-udm.enabled
  name: oai-udm
  repository: "file://../oai-udm"
  version: v1.5.0
- condition: oai-ausf.enabled
  name: oai-ausf
  repository: "file://../oai-ausf"
  version: v1.5.0
- condition: oai-amf.enabled
  name: oai-amf
  repository: "file://../oai-amf"
  version: v1.5.0
- condition: oai-smf.enabled
  name: oai-smf
  repository: "file://../oai-smf"
  version: v1.5.0
- condition: oai-spgwu-tiny.enabled
  name: oai-spgwu-tiny
  repository: "file://../oai-spgwu-tiny"
  version: v1.5.0
```

I. OAI 5G – VALUES.YAML

```
mysql:
  enabled: true
  weight: 0
  imagePullPolicy: IfNotPresent
  oai5gdatabase: basic
  imagePullSecrets:
    - name: "regcred"
  persistence:
    enabled: false
oai-nrf:
  enabled: true
  weight: 0
  kubernetesType: Vanilla #Openshift/Vanilla Vanilla for Upstream Kubernetes
  nfimage:  # image name either locally present or in a public/private repository
    repository: docker.io/oaisoftwarealliance/oai-nrf          ## The image will be pulled from dockerhub
    version: v1.5.0                                    ## The branch to be used to pull from dockerhub
    # pullPolicy: IfNotPresent or Never or Always
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: "regcred"
  nodeSelector:
   cnf: load
oai-udr:
  enabled: true
  weight: 1
  kubernetesType: Vanilla #Openshift/Vanilla Vanilla for Upstream Kubernetes
  nfimage:  # image name either locally present or in a public/private repository
    repository: docker.io/oaisoftwarealliance/oai-udr          ## The image will be pulled from dockerhub
    version: v1.5.0                                    ## The branch to be used to pull from dockerhub
    # pullPolicy: IfNotPresent or Never or Always
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: "regcred"
  nodeSelector:
   cnf: load
oai-udm:
  enabled: true
  weight: 2
  kubernetesType: Vanilla #Openshift/Vanilla Vanilla for Upstream Kubernetes
  nfimage:  # image name either locally present or in a public/private repository
    repository: docker.io/oaisoftwarealliance/oai-udm          ## The image will be pulled from dockerhub
    version: v1.5.0                                    ## The branch to be used to pull from dockerhub
    # pullPolicy: IfNotPresent or Never or Always
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: "regcred"
  nodeSelector:
   cnf: load
oai-ausf:
  enabled: true
  weight: 3
  kubernetesType: Vanilla #Openshift/Vanilla Vanilla for Upstream Kubernetes
  nfimage:  # image name either locally present or in a public/private repository
    repository: docker.io/oaisoftwarealliance/oai-ausf          ## The image will be pulled from dockerhub
    version: v1.5.0                                    ## The branch to be used to pull from dockerhub
    # pullPolicy: IfNotPresent or Never or Always
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: "regcred"
  nodeSelector:
   cnf: load
oai-amf:
  enabled: true
  weight: 4
  kubernetesType: Vanilla #Openshift/Vanilla Vanilla for Upstream Kubernetes
  nfimage:  # image name either locally present or in a public/private repository
    repository: docker.io/oaisoftwarealliance/oai-amf          ## The image will be pulled from dockerhub
    version: v1.5.0                                    ## The branch to be used to pull from dockerhub
    # pullPolicy: IfNotPresent or Never or Always
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: "regcred"
  multus:
    create: true
    n1IPadd: "172.21.6.200"
    n1Netmask: "22"
    n1Gateway: "172.21.7.254"
    hostInterface: "bond0"      # Interface of the host machine on which this pod will be scheduled
  config:
    amfInterfaceNameForNGAP: "net1" # If multus creation is true then net1 else eth0
    amfInterfaceNameForN11: "eth0"  # Service based interface
    externalAusf: "yes"
    # Mandatory
    sst0: "1"
    sd0: "0xFFFFFF"
    # Optional upto 4 slices can be define using below template
    sst1: "1"
    sd1: "1"
    mcc: "001"
    mnc: "01"
    tac: "0x0001"
    nfRegistration: "yes"
    smfSelection: "no"          #Bug in SMF can not have SMF selection when using UDR to fetch user-
information
    useHttp2: "no"
    intAlgoList: '[ "NIA1" , "NIA1" , "NIA2" ]'
    ciphAlgoList: '[ "NEA1" , "NEA1" , "NEA2" ]'
  nodeSelector:
   cnf: load
oai-spgwu-tiny:
  enabled: true
  weight: 5
  kubernetesType: Vanilla #Openshift/Vanilla Vanilla for Upstream Kubernetes
  nfimage:  # image name either locally present or in a public/private repository
    repository: docker.io/oaisoftwarealliance/oai-spgwu-tiny    ## The image will be pulled from dockerhub
    version: v1.5.0                                    ## The branch to be used to pull from dockerhub
    # pullPolicy: IfNotPresent or Never or Always
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: "regcred"
  multus:
    create: true
    n3Ip: "172.21.6.201"
    n3Netmask: "22"
    n6Gw: "172.21.7.254"
    hostInterface: "bond0"
  config:
    n3If: "net1"  # net1 if gNB is outside the cluster network and multus creation is true else eth0
    n4If: "eth0" # use for SMF communication
    n6If: "net1"  # net1 if gNB is outside the cluster network and multus creation is true else eth0
(important because it sends the traffic towards internet)
    threadsN3Ul: "1"
    threadsN6Dl: "1"
    threadsN6Prio: 98
    threadsN3Prio: 88
    threadsN4Prio: 88
    netUeIp: "12.1.1.0/24"  # The range in which UE ip-address will be allocated should be configured the
same in SMF
    registerNRF: "yes"
    nrfFqdn: "oai-nrf-svc" # make sure this can be resolved by container dns
    #Mandatory to configure atlease one slice
    nssaiSst0: 1 # should match with SMF information
    nssaiSd0: "0xFFFFFF"  # should match with SMF information (Optional, if removed sd value will be
0xFFFFFF)
    dnn0: "oai" # should match with SMF information
    #Upto 4 slices can be added from here to add more you need to add manullay in config.yaml. Please
follow the same way of adding slices nssaiSST$,nssaiSd$,dnn$ ($=0,1,2,3)
    nssaiSst1: 1 # should match with SMF information (Optional, if removed slice will not be configured)
    nssaiSd1: "0xFFFFFF"  # should match with SMF information (Optional, if removed sd value will be
0xFFFFFF only if nssaiSst1 is configured)
    dnn1: "ims" # should match with SMF information
  nodeSelector:
   cnf: load
oai-smf:
  enabled: true
  weight: 6
  kubernetesType: Vanilla #Openshift/Vanilla Vanilla for Upstream Kubernetes
  nfimage:  # image name either locally present or in a public/private repository
    repository: docker.io/oaisoftwarealliance/oai-smf          ## The image will be pulled from dockerhub
    version: v1.5.0                                    ## The branch to be used to pull from dockerhub
    # pullPolicy: IfNotPresent or Never or Always
    pullPolicy: IfNotPresent
  imagePullSecrets:
    - name: "regcred"
  config:
    useLocalSubscriptionInfo: "no" #this means ip-address information will be fetched from UDR
    ueMtu: 1500
    dnsIpv4Address: "172.21.3.100" # configure the dns for UE don't use Kubernetes DNS
    dnsSecIpv4Address: "172.21.3.100" # configure the dns for UE don't use Kubernetes DNS
    defaultCSCFIpv4Address: "172.21.6.13" # For IMS server if needed
    discoverUpf: "yes"              # if NRF is used then it can be used for UPF discovery
    # You can define maximum 4 slices from here in case of more please change the configuration file.
    dnnNi0: "oai"
    pdusessiontype0: "IPv4"
    ipv4dnnRange0: "12.1.1.2 - 12.1.1.254"
    nssaiSst0: 1
    nssaiSd0: "0xFFFFFF"
    qosProfile5qi0: 2
    sessionAmbrUl0: "1000Mbps"
    sessionAmbrDl0: "1000Mbps"
    # Extra optional slice
    dnnNi1: "ims"
    pdusessiontype1: "IPv4v6"
    ipv4dnnRange1: "12.2.1.2 - 12.2.1.254"
    nssaiSst1: 1
    nssaiSd1: "0xFFFFFF"
    qosProfile5qi1: 1
```

```
     sessionAmbrUl1: "1000Mbps"
     sessionAmbrDl1: "1000Mbps"
   nodeSelector:
    cnf: load
```

J. MULTUS – DAEMONSET.YAML

```
---
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: network-attachment-definitions.k8s.cni.cncf.io
spec:
  group: k8s.cni.cncf.io
  scope: Namespaced
  names:
    plural: network-attachment-definitions
    singular: network-attachment-definition
    kind: NetworkAttachmentDefinition
    shortNames:
    - net-attach-def
  versions:
    - name: v1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          description: 'NetworkAttachmentDefinition is a CRD schema specified by the Network
Plumbing
            Working Group to express the intent for attaching pods to one or more logical or
physical
            networks.          More          information          available          at:
https://github.com/k8snetworkplumbingwg/multi-net-spec'
          type: object
          properties:
            apiVersion:
              description: 'APIVersion defines the versioned schema of this represen
                tation of an object. Servers should convert recognized schemas to the
                latest internal value, and may reject unrecognized values. More info:
                https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#resources'
              type: string
            kind:
              description: 'Kind is a string value representing the REST resource this
                object represents. Servers may infer this from the endpoint the client
                submits  requests  to.  Cannot  be  updated.  In  CamelCase.  More  info:
https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-
kinds'
              type: string
            metadata:
              type: object
            spec:
              description: 'NetworkAttachmentDefinition spec defines the desired state of a
network attachment'
              type: object
              properties:
                config:
                  description:  'NetworkAttachmentDefinition config is a JSON-formatted CNI
configuration'
                  type: string
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: multus
rules:
  - apiGroups: ["k8s.cni.cncf.io"]
    resources:
      - '*'
    verbs:
      - '*'
  - apiGroups:
      - ""
    resources:
      - pods
      - pods/status
    verbs:
      - get
      - update
  - apiGroups:
      - ""
      - events.k8s.io
    resources:
      - events
    verbs:
      - create
      - patch
      - update
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: multus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: multus
subjects:
- kind: ServiceAccount
  name: multus
  namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: multus
  namespace: kube-system
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: multus-cni-config
  namespace: kube-system
  labels:
    tier: node
    app: multus
data:
  # NOTE: If you'd prefer to manually apply a configuration file, you may create one here.
  # In the case you'd like to customize the Multus installation, you should change the
arguments to the Multus pod
  # change the "args" line below from
  # - "--multus-conf-file=auto"
  # to:
  # "--multus-conf-file=/tmp/multus-conf/70-multus.conf"
  # Additionally -- you should ensure that the name "70-multus.conf" is the alphabetically
first name in the
  # /etc/cni/net.d/ directory on each node, otherwise, it will not be used by the Kubelet.
  cni-conf.json: |
    {
      "name": "multus-cni-network",
      "type": "multus",
      "capabilities": {
        "portMappings": true
      },
      "delegates": [
        {
          "cniVersion": "0.3.1",
          "name": "default-cni-network",
          "plugins": [
            {
              "type": "flannel",
              "name": "flannel.1",
                "delegate": {
                  "isDefaultGateway": true,
                  "hairpinMode": true
                }
              },
              {
                "type": "portmap",
                "capabilities": {
                  "portMappings": true
```

```
                 }
               }
            ]
          }
        ],
        "kubeconfig": "/etc/cni/net.d/multus.d/multus.kubeconfig"
      }
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: kube-multus-ds
  namespace: kube-system
  labels:
    tier: node
    app: multus
    name: multus
spec:
  selector:
    matchLabels:
      name: multus
  updateStrategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        tier: node
        app: multus
        name: multus
    spec:
      hostNetwork: true
      tolerations:
      - operator: Exists
        effect: NoSchedule
      - operator: Exists
        effect: NoExecute
      serviceAccountName: multus
      containers:
      - name: kube-multus
        image: ghcr.io/k8snetworkplumbingwg/multus-cni:snapshot
        command: ["/thin_entrypoint"]
        args:
        - "--multus-conf-file=auto"
        - "--multus-autoconfig-dir=/host/etc/cni/net.d"
        - "--cni-conf-dir=/host/etc/cni/net.d"
        resources:
          requests:
            cpu: "100m"
            memory: "50Mi"
          limits:
            cpu: "100m"
            memory: "50Mi"
        securityContext:
          privileged: true
        volumeMounts:
        - name: cni
          mountPath: /host/etc/cni/net.d
        - name: cnibin
          mountPath: /host/opt/cni/bin
        - name: multus-cfg
          mountPath: /tmp/multus-conf
      initContainers:
        - name: install-multus-binary
          image: ghcr.io/k8snetworkplumbingwg/multus-cni:snapshot
          command: ["/install_multus"]
          args:
            - "--type"
            - "thin"
          resources:
            requests:
              cpu: "10m"
              memory: "15Mi"
          securityContext:
            privileged: true
          volumeMounts:
            - name: cnibin
```

```
              mountPath: /host/opt/cni/bin
              mountPropagation: Bidirectional
      terminationGracePeriodSeconds: 10
      volumes:
        - name: cni
          hostPath:
            path: /etc/cni/net.d
        - name: cnibin
          hostPath:
            path: /opt/cni/bin
        - name: multus-cfg
          configMap:
            name: multus-cni-config
            items:
            - key: cni-conf.json
              path: 70-multus.conf
```

K. Single Message on Multiline - JSON

```
{
  "_index": "packets-2023-03-27",
  "_type": "doc",
  "_score": null,
  "_source": {
    "layers": {
      "frame": {
        "frame.encap_type": "25",
        "frame.time": "Mar 27, 2023 15:52:58.044434000 UTC",
        "frame.offset_shift": "0.000000000",
        "frame.time_epoch": "1679932378.044434000",
        "frame.time_delta": "0.000000000",
        "frame.time_delta_displayed": "0.000000000",
        "frame.time_relative": "0.000000000",
        "frame.number": "1",
        "frame.len": "70",
        "frame.len_tree": {
          "_ws.expert": {
            "frame.len_lt_caplen": "",
            "_ws.expert.message": "Frame length is less than captured length",
            "_ws.expert.severity": "8388608",
            "_ws.expert.group": "117440512"
          },
          "_ws.malformed": "Malformed Packet"
        },
        "frame.cap_len": "86",
        "frame.marked": "0",
        "frame.ignored": "0",
        "frame.protocols": "sll:ethertype:vlan:ethertype:ip:udp:pfcp"
      },
      "sll": {
        "sll.pkttype": "3",
        "sll.hatype": "1",
        "sll.halen": "6",
        "sll.src.eth": "3a:2d:03:e6:1d:50",
        "sll.unused": "00:00",
        "sll.etype": "0x8100"
      },
      "vlan": {
        "vlan.priority": "0",
        "vlan.dei": "0",
        "vlan.id": "1004",
        "vlan.etype": "0x0800",
        "vlan.trailer": "00:00:98:00:00:00:da:bb:21:64:1f:03:a6:02:32:00",
        "vlan.trailer_tree": {
          "_ws.expert": {
            "eth.padding_bad": "",
            "_ws.expert.message": "Didn't find padding of zeros, and an undecoded trailer
exists. There may be padding of non-zeros.",
            "_ws.expert.severity": "4194304",
            "_ws.expert.group": "150994944"
          }
        }
      },
      "ip": {
        "ip.version": "4",
        "ip.hdr_len": "20",
        "ip.dsfield": "0x00",
          "ip.dsfield_tree": {
            "ip.dsfield.dscp": "0",
            "ip.dsfield.ecn": "0"
          },
          "ip.len": "50",
          "ip.id": "0xc56d",
          "ip.flags": "0x02",
          "ip.flags_tree": {
            "ip.flags.rb": "0",
            "ip.flags.df": "1",
            "ip.flags.mf": "0"
          },
          "ip.frag_offset": "0",
          "ip.ttl": "64",
          "ip.proto": "17",
          "ip.checksum": "0x89bc",
          "ip.checksum.status": "2",
          "ip.src": "10.189.235.13",
          "ip.addr": "10.189.235.13",
          "ip.src_host": "10.189.235.13",
          "ip.host": "10.189.235.13",
          "ip.dst": "10.189.235.9",
          "ip.addr": "10.189.235.9",
          "ip.dst_host": "10.189.235.9",
          "ip.host": "10.189.235.9"
        },
        "udp": {
          "udp.srcport": "8805",
          "udp.dstport": "8805",
          "udp.port": "8805",
          "udp.port": "8805",
          "udp.length": "30",
          "udp.checksum": "0x9a54",
          "udp.checksum.status": "2",
          "udp.stream": "0",
          "Timestamps": {
            "udp.time_relative": "0.000000000",
            "udp.time_delta": "0.000000000"
          },
          "udp.payload": "21:38:00:12:72:48:93:40:23:10:74:41:00:ad:f6:00:00:27:00:02:80:06"
        },
        "pfcp": {
          "pfcp.flags": "0x21",
          "pfcp.flags_tree": {
            "pfcp.version": "1",
            "pfcp.spare_b4": "0",
            "pfcp.spare_b3": "0",
            "pfcp.fo_flag": "0",
            "pfcp.mp_flag": "0",
            "pfcp.s": "1"
          },
          "pfcp.msg_type": "56",
          "pfcp.length": "18",
          "pfcp.seid": "0x7248934023107441",
          "pfcp.seqno": "44534",
          "pfcp.spare_oct": "0",
          "Report Type : ": {
            "pfcp.ie_type": "39",
            "pfcp.ie_len": "2",
            "pfcp.spare_b7": "1",
            "pfcp.report_type.uisr": "0",
            "pfcp.report_type.sesr": "0",
            "pfcp.report_type.tmir": "0",
            "pfcp.report_type.upir": "0",
            "pfcp.report_type.erir": "0",
            "pfcp.report_type.usar": "0",
            "pfcp.report_type.dldr": "0",
            "IE data not decoded by WS yet": {
              "_ws.expert": {
                "pfcp.ie_data_not_decoded": "",
                "_ws.expert.message": "IE data not decoded by WS yet",
                "_ws.expert.severity": "4194304",
                "_ws.expert.group": "83886080"
              }
            }
          }
        }
      }
    }
  }
```

L. KAFKA – Docker compose file

```
version: "3"
services:
  zookeeper:
    image: zookeeper
    restart: always
    container_name: zookeeper
    hostname: zookeeper
    ports:
      - 2181:2181
    environment :
      ZOO_MY_ID: 1

  kafka:
    image: wurstmeister/kafka
    container_name: kafka
    ports:
      - 9092:9092
    environment:
      KAFKA_ADVERTISED_HOST_NAME: 10.0.71.29
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181

  kafka_manager:
    image: hlebalbau/kafka-manager:stable
    container_name: kakfa-manager
    restart: always
    ports:
      - "9000:9000"
    depends_on:
      - zookeeper
      - kafka
    environment:
      ZK_HOSTS: "zookeeper:2181"
      APPLICATION_SECRET: "random-secret"
    command: -Dpidfile.path=/dev/null
```

M. Elastic Search & Kibana – Docker Compose file

```
version: '3.7'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.4.0
    container_name: elasticsearch
    restart: always
    environment:
      - xpack.security.enabled=false
      - discovery.type=single-node
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    cap_add:
      - IPC_LOCK
    volumes:
      - elasticsearch-data-volume:/usr/share/elasticsearch/data
    ports:
      - 9200:9200
      - 9300:9300

  kibana:
    container_name: kibana
    image: docker.elastic.co/kibana/kibana:7.4.0
    restart: always
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
    ports:
      - 5601:5601
    depends_on:
      - elasticsearch

volumes:
  elasticsearch-data-volume:
    driver: local
```

N. PYHON Code to Push the PFCP JSON file

```
from kafka import KafkaProducer
import json

def on_send_success(record_metadata):
    print(record_metadata.topic)
    print(record_metadata.partition)
    print(record_metadata.offset)

def on_send_error(excp):
    log.error('I am an errback', exc_info=excp)

def json_serializer(data):
    return json.dumps(data).encode("utf-8")

producer = KafkaProducer(bootstrap_servers=['10.0.71.29:9092'],
value_serializer=json_serializer)

if __name__ == "__main__":

    pfcp_file = open('pfcp.json', 'r')
    Lines = pfcp_file.readlines()
    for line in Lines:
        push_json= line.strip()
        record = json.loads(push_json)
        producer.send("pfcp",
record).add_callback(on_send_success).add_errback(on_send_error)
    producer.flush()
    pfcp_file.close()
```

O. PYHON Code to Push the N2 JSON file

```
from kafka import KafkaProducer
import json

def on_send_success(record_metadata):
    print(record_metadata.topic)
    print(record_metadata.partition)
    print(record_metadata.offset)

def on_send_error(excp):
    log.error('I am an errback', exc_info=excp)

def json_serializer(data):
    return json.dumps(data).encode("utf-8")

producer = KafkaProducer(bootstrap_servers=['10.0.71.29:9092'],
value_serializer=json_serializer)

if __name__ == "__main__":

    pfcp_file = open('n2.json', 'r')
    Lines = pfcp_file.readlines()
    for line in Lines:
        push_json= line.strip()
        record = json.loads(push_json)
        producer.send("n2",
record).add_callback(on_send_success).add_errback(on_send_error)
    producer.flush()
    pfcp_file.close()
```

P. PYHON Code to Push the HTTP2/N11 JSON file

```
from kafka import KafkaProducer
import json

def on_send_success(record_metadata):
    print(record_metadata.topic)
    print(record_metadata.partition)
    print(record_metadata.offset)

def on_send_error(excp):
    log.error('I am an errback', exc_info=excp)

def json_serializer(data):
    return json.dumps(data).encode("utf-8")

producer = KafkaProducer(bootstrap_servers=['10.0.71.29:9092'],
value_serializer=json_serializer)

if __name__ == "__main__":

    pfcp_file = open('http2.json', 'r')
    Lines = pfcp_file.readlines()
    for line in Lines:
        push_json= line.strip()
        record = json.loads(push_json)
        producer.send("http2",
record).add_callback(on_send_success).add_errback(on_send_error)
    producer.flush()
    pfcp_file.close()
```

Q. Logstash Configuration file for PFCP

```
input {
    kafka {
            bootstrap_servers => "10.0.71.29:9092"
            topics => ["pfcp"]
    }
}

output {
   elasticsearch {
      hosts => ["10.0.71.29:9200"]
      index => "pfcp"
      workers => 1
    }
}
```

R. Logstash Configuration file for HTTP2

```
input {
    kafka {
            bootstrap_servers => "10.0.71.29:9092"
            topics => ["http2"]
    }
}

output {
   elasticsearch {
      hosts => ["10.0.71.29:9200"]
      index => "http2"
      workers => 1
    }
}
```

S. Logstash Configuration file for N2

```
input {
    kafka {
            bootstrap_servers => "10.0.71.29:9092"
            topics => ["N2"]
    }
}

output {
   elasticsearch {
      hosts => ["10.0.71.29:9200"]
      index => "N2"
      workers => 1
    }
}
```

C APPENDIX – DRAFT PAPER

A Novel Unstructured Data Storage Function (UDSF) Architecture for 5G Core Networks Based on Kubernetes and Kafka

Furqan Ahmad[1], Bruno Dzogovic[4,1], Thanh Van Do[4,1], Boning Feng[1], Bernardo Flores[1], Van Thuan Do[3,1], Niels Jacot[3]

[1] Oslo Metropolitan Univeristy, Pilestredet 35, 0167 Oslo, Norway
[2] Mavenir Systems AB, Garvis Carlssons Gata 1, 0217 Oslo, Norway
[3] Wolffia AS, Haugerudvn. 40, 0673 Oslo, Norway
[4] Telenor ASA. Snarøyveien 30 1331 Fornebu, Norway
furqan.ahmad@mavenir.com
{bruno.dzogovic, thanh-van.do@telenor.com}
{boning.feng, bersan}@oslomet.no
{vt.do, n.jacot}@wolffia.net

1. Introduction

As the Internet of Things devices are dominating various industry verticals for facilitating smart infrastructure, robust connectivity has become a mandatory requirement. 5G established itself as the prime candidate for delivering seamless connectivity with higher quality of service to satisfy the needs of customers and industries. However, as the connectivity degree in networks increases, so does the network activity and thus data which traverses the network. With excessive amounts of data flows, system administrators need to consider the management and control systems in place, which are dedicated to the administration of the 5G cores. High traffic loads and traffic bursts can incur significant security deficiencies, cause degradation of quality of service and expose the users to various threats such as identity theft, denial of service or attacks that in general compromise the personal data and information. Consequently, this research paper focuses efforts to examine and experiment with the possibility of building a comprehensive solution for monitoring and data collection and suggest a seamless integration for a unified 5G Core Unstructured Data Storage Function, while fostering the prospects for instituting machine learning based systems for data processing in 5G.

2. Research Background

To understand how monitoring and data collection should be achieved within a 5G core network, it is necessary to describe and comprehend the 5G core functionality. Communication between the Radio Access Network (RAN) and the core is essential to establish the protocols and which data should be collected and parsed for the purpose of further applying different machine learning methods. However, the methodology of data capture and storage is critical for facilitating advanced intelligence of the 5G core, as different types of data can be associated with diverse traffic. In general terms, data can be divided into user-plane and control-plane data. The control-plane data refers to the information that the core network functions transmit between each other and with the RAN, to enable the functionality of the 5G system. User-plane data is the information that flows from the User Equipment (UE) to the core network. This data is also private and must be treated with care and confidentiality.

A. The 5G Core Network Architecture

The 5G Core architecture represents a mobile core network that is responsible for session management, authentication, service continuity, and security. Within the 5G Core Network architecture, since there are no pre-defined interfaces, one or more network functions (NFs) can be chained to create new services, which enables rapid creation and deployment of new services. According to the European Telecommunications Standards Institute (ETSI) and 3GPP, the 5G system architecture is represented by Service-Based Architecture (SBA) or reference point architecture (point-to-point-based architecture) [1].

Figure 1 depicts the 5GC service-based architecture, where every network service is disaggregated into its own independent function communicating over a common communication bus. The service-based architecture enables flexible and efficient network management. This architecture minimizes dependencies between the Access Network (AN) and Core Network (CN). In a SBA, services can register themselves and subscribe to other services. This enables flexible development of services and connections to other components without introducing new interfaces.

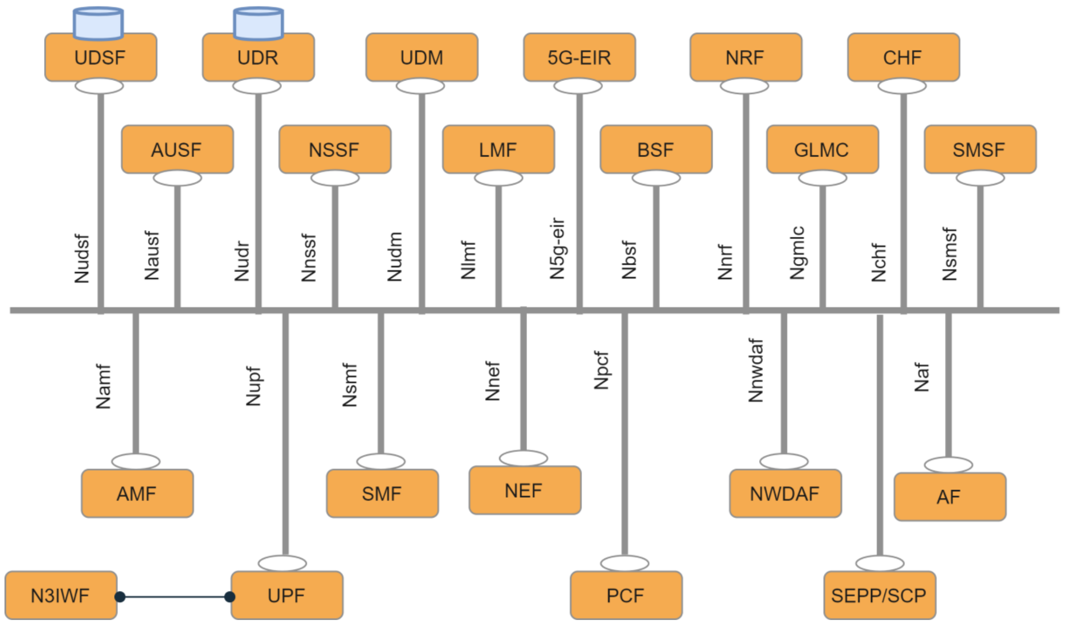


Figure 1. 5GC Service-Based Architecture

Communication within a 5G Core is enabled via a service framework, which involves service registration, authorization, and discovery. The Network Functions (NF) within the 5G Core will only use Services Based Interfaces (SBI). Every NF using SBI can potentially interact with any other NF directly by using SBI. All other NFs are not supporting SBI, or outside the trust domain communicate via the Network Exposure Function (NEF) or via Message Routing Forwarding Function (MRFF), which is a proxy-like function to support point-to-point (P2P) on the Northbound interface. In addition, NFs can support (if required) an extension/plugin framework to enable customization, extension, and enrichment. This enables the 5GC NFs to interwork with any custom APIs of multiple vendors or third-party applications [1].

Within the 5G Core architecture, new services can be created by combining and reusing existing network functions in a dynamic manner. This enables rapid creation and deployment of new services. A Service Based Architecture can be described as a toolbox of capabilities that can be orchestrated, optimized, and sliced to support a wide variety of services as well as fulfill specific operator's needs. Service Based Architectures (SBA) have become an important enabler for network simplification, automation, and operational efficiency [2].

B. The Next-Generation Radio Access Network in 5G

5G proposed the new architecture for the Radio Access Network (RAN) to accommodate the existing networks and support the implementation of the 5G use cases. Schematically, the 5G architecture resembles the 4G LTE. Figure 2 shows the high-level 5G architecture and its components. It comprises of User Equipment (UE), which includes a Mobile Station (MS) and a Universal Subscriber Identification Module (USIM), Next Generation RAN and 5G Core Network (CN). NG-RAN provides data access to the 3GPP compliant devices via 5G CN, whereas devices connected to the public network can access the 5G core network via Non-3GPP Interworking Function (N3IWF). N3IWF is equivalent to the ePDG in the LTE network and provides the control plane connectivity to the AMF via N2 reference point, as well as user plane connectivity towards the UPF via N3 reference point. The 5G architecture provides complete separation of control plane and user plane traffic for 3GPP and non-GPP access [1].

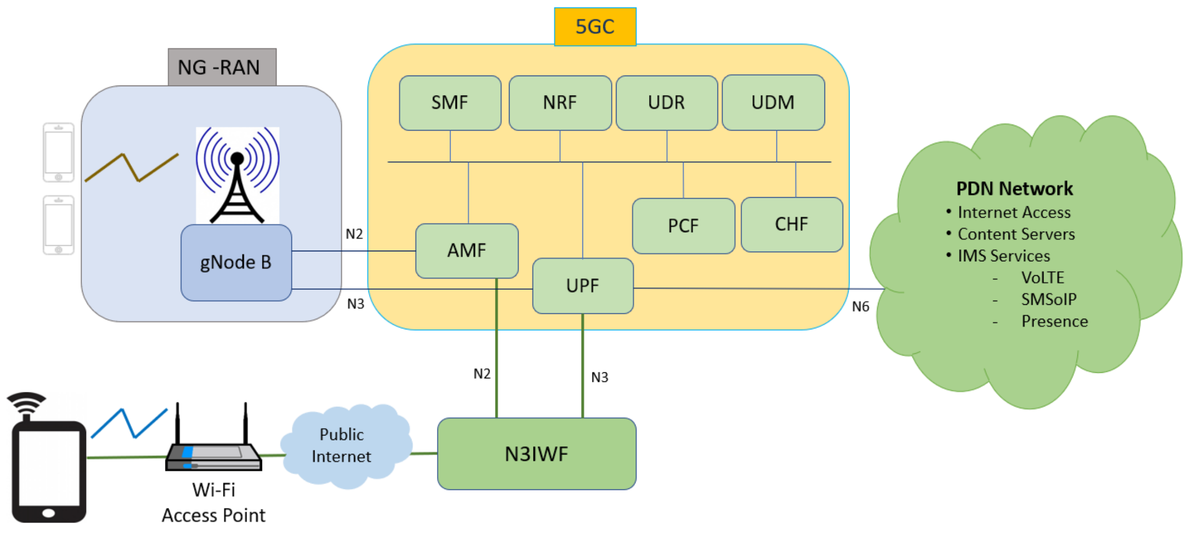


Figure 2. 5G Architecture

The NG-RAN architecture is a pillar of the 5G system. It has been designed to operate in two modes to support various combinations of Dual Connectivity (DC) with LTE and different core networks [3]:

- **Non-standalone (NSA):** In NSA mode, NG-RAN comprises of Next Generation Node-B (gNB) and Next-Generation Evovled Node-B (ng-eNBs). gNBs and ng-eNBs interoperate with one another to provide connectivity and they are connected to the same 5G core network, providing DC toward the same terminal.
- **Standalone (SA):** In SA mode, a NG-RAN comprises of only gNBs, which connect directly to the 5G core network.

A Next Generation (NG) reference point is introduced in the 5G network to provide the connectivity to the 5G core. NG-RAN includes gNB and gNB supports Frequency Division Duplex (FDD), Time Division Duplex (TDD) or dual mode operation. The interconnection between the gNB is achieved via Xn interface. In Release-15, 3GPP added split functionality to gNB so that more flexibility can be achieved at the RAN level. gNB is split into three logical nodes [4]:

- Central Unit (CU) or gNB-CU
- Distributed Unit (DU) or gNB-DU
- Radio Unit (RU)

Figure 4 39 depicts the high-level split architecture of gNB. It shows that multiple gNBs are interconnected through the Xn Interface, and the same gNBs are connected to the 5G CN through the NG interface. A gNB consists of a gNB-CU and one or more gNB-DU(s), and the interface between gNB-CU and gNB-DU is called F1. The F1 interface supports signaling exchange and data transmission between distributed and central units. It also separates the radio network layer and transport network layer and enables the exchange of UE-associated and non-UE-associated signaling. gNB-CU provides connectivity to the 5G CN via NG interface and connectivity to the gNB-CU via Xn interface. In 3GPP Release-15 [4], one gNB-DU connects to only one gNB-CU, but it allows implementations to connect multiple gNB-CUs to a single gNB-DU for fault tolerance and resiliency. One or more cells can be supported by a single gNB-DU. The internal split of gNB is not visible to the 5G core network and other RAN nodes, it is seen in the network as a single gNB [4].

Figure 3 outlines the details of the architecture split, as per the 3GPP specification TS 38.401 [4]. The CU is split further into control and user plane. The Central Unit Control Plane (CU-CP) is responsible for the control plane messages and the Central Unit User Plane (CU-UP) is responsible for the user plane messages. A gNB may consist of a single CUCP and multiple CUUPs and DUs. CUUPs and DUs can be connected to a single CUCP. The control interface between the CUCP and DU is called F1-C and user plane interface between CUUP and DU is called F1U. CUUP is connected to the CUCP on E1 interface.

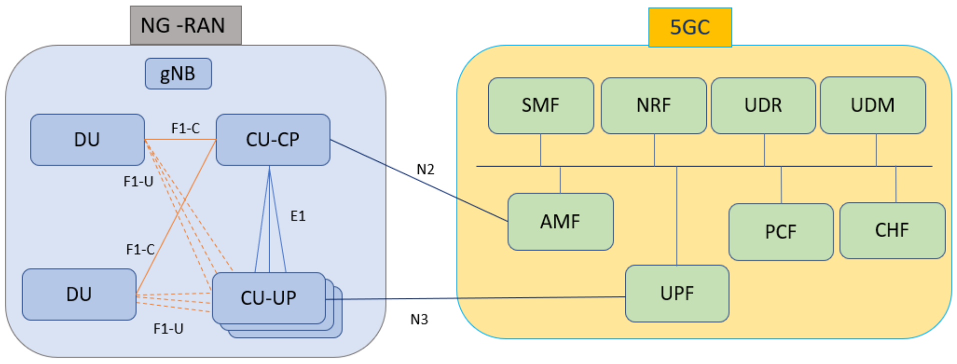


Figure 3. NG-RAN – Detailed Split Architecture

Protocol-oriented 5G control-plane communication

The protocol-centric communication of 5G fits into five main categories:

- Next-Generation Application Protocol (NGAP) over N2 reference point, with only NG signaling traffic (NG-C) between the AMF and gNB.
- Packet Forwarding Control Protocol (PFCP) over N4 reference point between the SMF and UPF.
- HTTP2 over various reference points, uses the default CNI between all the network functions.
- General Packet Radio Service Tunnelling Protocol User (GTP-U) over N3 reference point with only NG user plane (NG-U) between gNB and UPF.
- Traffic towards the Data Network (DN) from UPF over N6 reference point.

The AMF and SMF participate in the signaling messages and the UPF for the user plane traffic, hence we focus on the analysis of protocols relevant to AMF, SMF and UPF.

The 5G core network is designed on the principles of service-based architectures and adopts the HTTP/2 as the application layer protocol. HTTP/2 is used at the control plane for communication between the 5G core network functions. Consequently, the AMF then oversees the connection and mobility management tasks, while all messages related to session management are forwarded to the SMF over the N11 interface via HTTP/2 [2].

Furthermore, in a 5G core network, the PFCP (Packet Convergence Control Protocol) provides the control means for SMF to manage packet processing, as well as means for UPF to perform forwarding. PFCP was also used as a protocol in the Control and User Plane Separation (CUPS) architecture of the

Evolved Packet Core (EPC) between the Serving Gateway Control plane function (SGW-C) and Serving Gateway User plane function (SGW-U) as Sxa interface, as well as between the PDN Gateway Control plane function (PGW-C) and PDN Gateway User plane function (PGW-U) as Sxb interface. In a 5G core network, the Control Plane (CP) is managed by the SMF, while the User plane is managed by the UPF. The Packet Forwarding Control Protocol (PFCP) is a defined protocol by the 3GPP and resides between the control and user plane network function described in the TS 29.244 specification [5].

The UDSF function

UDSF is a 5G network function responsible for storing unstructured data. The 5G system supports stateless NFs where the computational resource is decoupled from the storage resource. When NFs are implemented to support stateless architecture, then NF can utilize the UDSF for storing/retrieving unstructured data into/from a storage. The unstructured data does not mean that data has no structure but rather it is pointing to the fact that 3GPP has not defined structure in specification and suppliers are allowed to define their own structure and then utilize UDSF to store the date into storage. The UDSF is deployed along with the other 5G NFs within the same Public Landline Mobile Network (PLMN) and with the possibility having the common UDSF for all NFs to store/retrieve data or dedicated UDSF for particular NF depending on operator needs [6].

However, it is unclear how the control-plane data shall be collected, processed, and stored.

C. Kubernetes and Kafka

Kubernetes is a container management system which enables orchestration and automation of deployments. Since the 5G core functions are service-oriented, deploying resilient services is exactly what Kubernetes excels in. By leveraging techniques like service discovery, replication control or stateful applications, Kubernetes provides a flexible and automated way for deploying and managing 5G core networks [7].

Apache Kafka is designed for distributed streaming, and it comprises of several components. Figure 4 shows the high-level architecture of Kafka along with its necessary components and based on this it can be divided into eight different components and a Kafka manager [8]:

- Brokers: The servers associated to the Kafka cluster are known as brokers and each cluster can have multiple brokers as shown in Figure 4, there are three servers and three brokers in the cluster. Connection with any one of the broker within the implies a connection to the whole cluster(PROJECT-PRO, 2023).
- Topics: A stream of messages is referred to as Kafka topics. Producers are storing the data into topics and consumers are reading it from the topic.
- Producers: Producer in Kafka is responsible for publishing the messages into the topics. The message from produced is received by broker and then it appends it to the partition. The producer also has flexibility to publish the message into its own choice of partition.
- Consumers: Consumers read the data from the specific topic of the Kafka cluster.
- Partitions: Topics in Kafka are divided into partitions to enable multiple consumers to read from the topic in parallel. The partitions are distributed across servers in the Kafka cluster as shown in Figure 4 of Topic A.

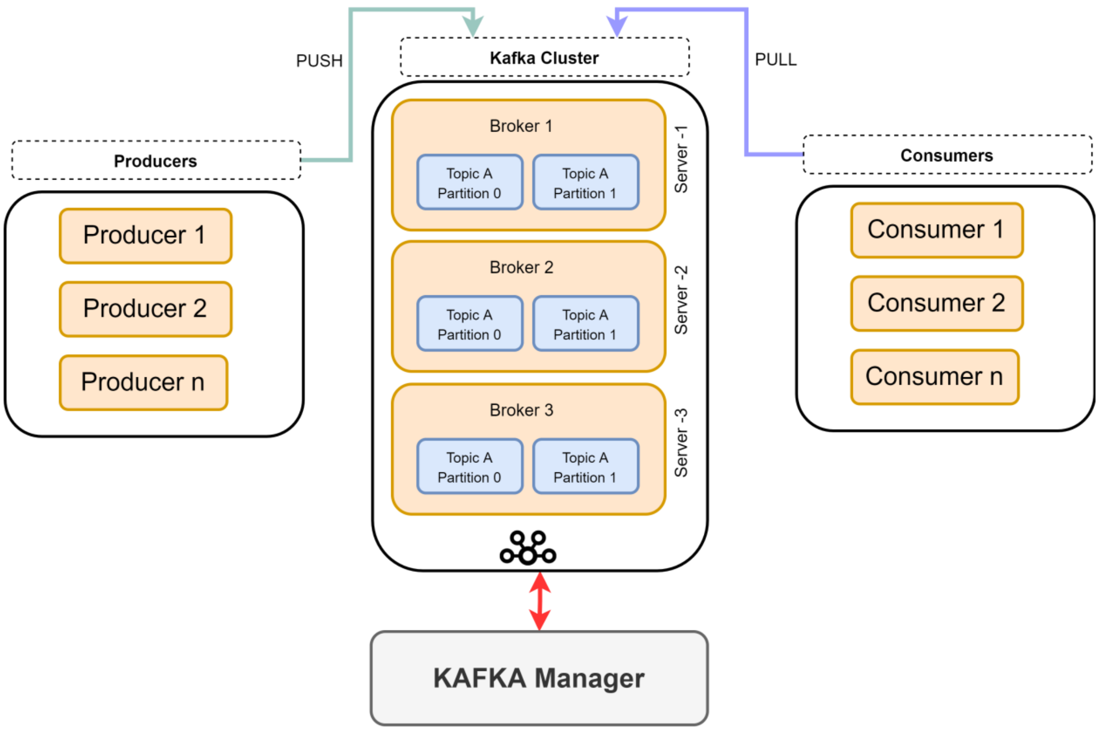


Figure 4. Kafka Architecture

- Partition Offset: Messages records within the partition are provided with the offset. The offset value associated to the message can uniquely identify the message(PROJECT-PRO, 2023).
- Replica: Partition of the topics are kept on the different brokers, and these are called replica.
- Leader and Follower: One server will always act as leader for the particular partition and the leader is responsible for performing all the read and write tasks.
- Kafka Manager (CMAK): Kafka manager provides the UI to manage the Kafka cluster and there are Kafka managers available freely on the internet. CMAK is a comprehensive tool to manage Apache Kafka cluster.

3. Methodology and Implementation

In this research, we craft a data collection and processing architecture that shall utilize the advantages of Kafka and Kubernetes to obtain relevant information from the control plane in a 5G core. The goal of the solution is to collect the data from the network function on specific interfaces/protocols by using Wireshark, process the data and then make the clean data available to machine learning system to be consumed in real time. The overall solution has been divided into the three main blocks for simplicity as shown in Figure 5.

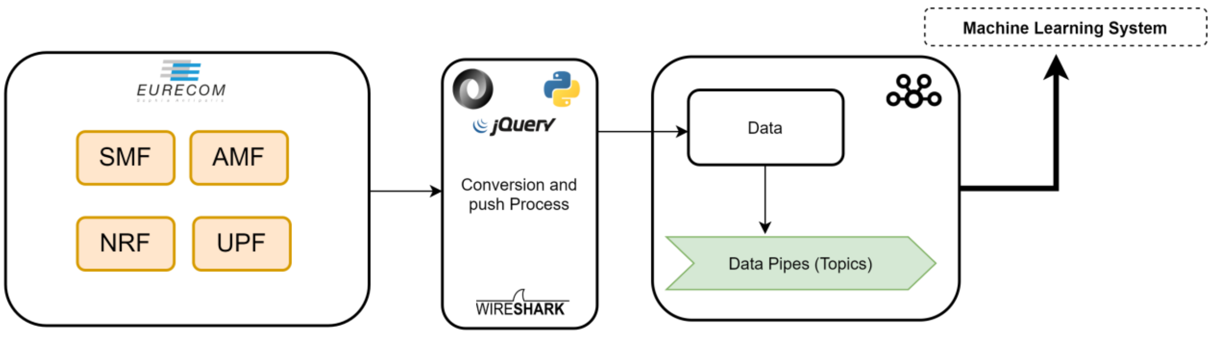


Figure 5. Solution Overview - Data Collection and Processing

The first block of the solution is to have functional 5G core network deployed in lab facility of Oslo Metropolitan University [9]. To accomplish the first block, OpenAirInterface5G (OAI) [10] 5G core has been deployed and made operational. The open-source OAI 5G core is deployed with other network functions such as UDM, UD and AUSF but only a few network functions are shown in Figure 5. The focus is mainly on the data collection and processing from SMF, AMF and UPF on specific reference points. The same process of data capturing, collection and processing can be applied to the other network function as well.

The second main block of the solution focuses on the data capturing, separating the data with respect to the 5G reference points such as N2, N4 and N11/HTTP2. The separation of the data with respect to the reference points ensures that machine learning systems can load the data instantly without implementing further parsing algorithm related to data separation. Furthermore, captured separated reference point data is converted into JSON format with tshark and then cleaned up with jQuery [11]. Finally, Kafka producer written in Python is used to push the data to Kafka server.

The final module of the solution comprises of the Kafka server setup in the lab facility of Oslo Metropolitan University. The purpose of the Kafka server is to make the collected and separated data available for the machine learning system to be consumed in real time. The data is loaded to the Kafka server by the Kafka producer written in the python for each reference point.

The network functions are residing on the worker nodes and all the traffic associated to the network functions passes through the Network Interface Card (NIC) of the worker nodes. The data captured on the worker node is raw as it captures all the packets passing through the worker NIC. The separation and cleaning process of the data has been divided into three sub processes P-1, P-2 and P-3 as shown in Figure 6. The purpose of each sub-process is to perform a certain action on the raw data and then forward the data to the next sub-process for further processing.

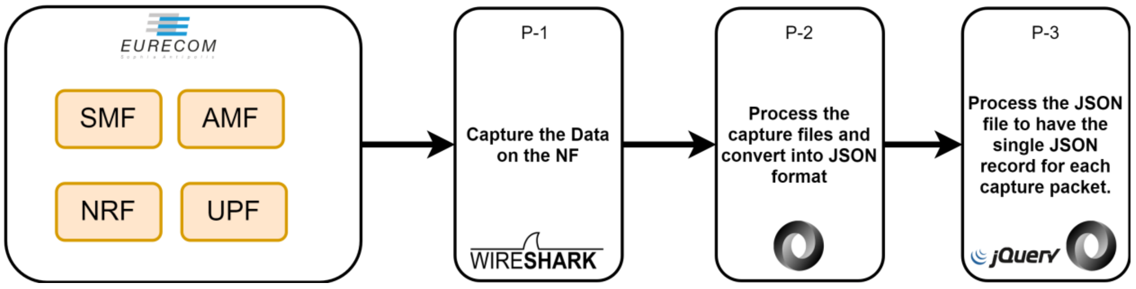

EURECOM
SMF
AMF
NRF
UPF
P-1
Capture the Data on the NF
WIRESHARK
P-2
Process the capture files and convert into JSON format
P-3
Process the JSON file to have the single JSON record for each capture packet.
jQuery

Figure 6. Raw Data Separation Process

The actions performed by each sub-process has been detailed as below:

- **P-1**: The data is captured on the worker nodes/network functions via tshark utility using specific filter related to the protocols. NGAP filter has been used to separate the packets relevant to the N2 reference point, PFCP filter has been deployed to capture packets related to the N4 reference point and HTTP2 filter is used for N11 packets between AMF and SMF (this filter might capture HTTP packets from other network functions, but this can be separated by applying the filters on the URI within the HTTP messages). The captured data is then stored in a separate file with timestamp included in the name of the file. The P-1 sub-process then forwards the data to P-2 for further processing.
- **P-2**: This sub-process will receive three captured files for each reference point such as N2 (NGAP), N4 (PFCP) and N11 (HTTP/2). The sub-process will convert these files to JSON format via tshark utility hence this will produce three JSON files and forward the files to the next sub-process for further processing.
- **P-3**: P-3 sub-process will receive three JSON files from the P-2 sub-process. Ideally, these files should be ready to be processed further but received files by P-2 sub-process does not represent the single message per line.

The P-3 sub process will convert the messages inside the received file from P-2 into single JSON record via jQuery and then file is ready to be used for further processing and loading into the Kafka. jQuery is deployed to perform the conversion.

The files produced from the previous processing modules contain a single JSON record for each message processed by the network functions in the 5G core network. The files are clean and ready to be processed for further loading onto the Kafka server. Data loading into Kafka is achieved by following steps:

- Set up Kafka broker and manager along with the zookeeper.
- Setup topics for each reference point.
- Setup Kafka producer to load data into Kafka topics.

The setup has three Kafka producers to push the JSON records to three Kafka topics – N2, HTTP2 and pfcp. The high-level flow of message loading into Kafka topics is shown below in Figure 7.

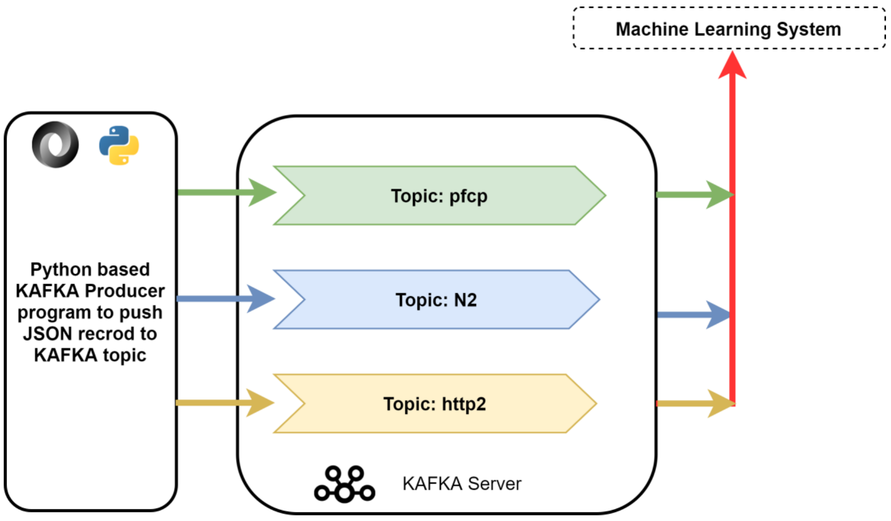

Machine Learning System
Python based KAFKA Producer program to push JSON recrod to KAFKA topic
Topic: pfcp
Topic: N2
Topic: http2
KAFKA Server

Figure 7. Data loading into Kafka

As presented in Figure 7, PFCP messages are pushed to the pfcp topic, N2 reference messages are pushed to N2 topic of Kafka and N11 messages are published into the http2 topic. These messages from each producer are published into the partition associated with the topics. The messages are published in real time without any latency by python program and separation of the messages is achieved by having separate Kafka topics.

The messages pushed to the Kafka broker are ready to be consumed by the Kafka consumer. The machine learning system should implement the Kafka consumer and it should inform the Kafka broker indicating the partitions it wants to consume. The consumer specifies the offset in the request towards the Kafka broker and accordingly, it will receive a log chunk from the offset position. Machine learning system has control over the offset hence it can request for the same data multiple times if required. Kafka broker will not delete the data after the consumption from consumer, data will be removed by Kafka broker after it reaches the configured retention period.

4. Evaluation

The data available in the Kafka broker allows any Kafka consumer to ingest the data from any partition. The Machine Learning (ML) System will ingest the data from the Kafka topics, but an additional setup has been created in the University lab to run the Elastic stack [12]. An Elastic stack is a group of open-source products from Elastic, and it allows the processing, filtering, storing, and visualization of data in real-time from various sources. The stack comprises the three components such as Elastic search, Logstash, and Kibana, and this is also known as ELK. All the products used in ELK are developed, managed, and maintained by Elastic. In ELK, Elasticsearch provides a full-text search and analysis engine. Logstash is a flexible log parsing engine that takes the data from an input source and applies different transformations and enhancements before shipping it to various destinations. Finally, Kibana in ELK provides the visualization layer on top of Elasticsearch, and this gives the users ability to analyze and visualize the data in real time.

Figure 8 shows the high-level data loading flow into the Elastic stack along with the machine learning system.

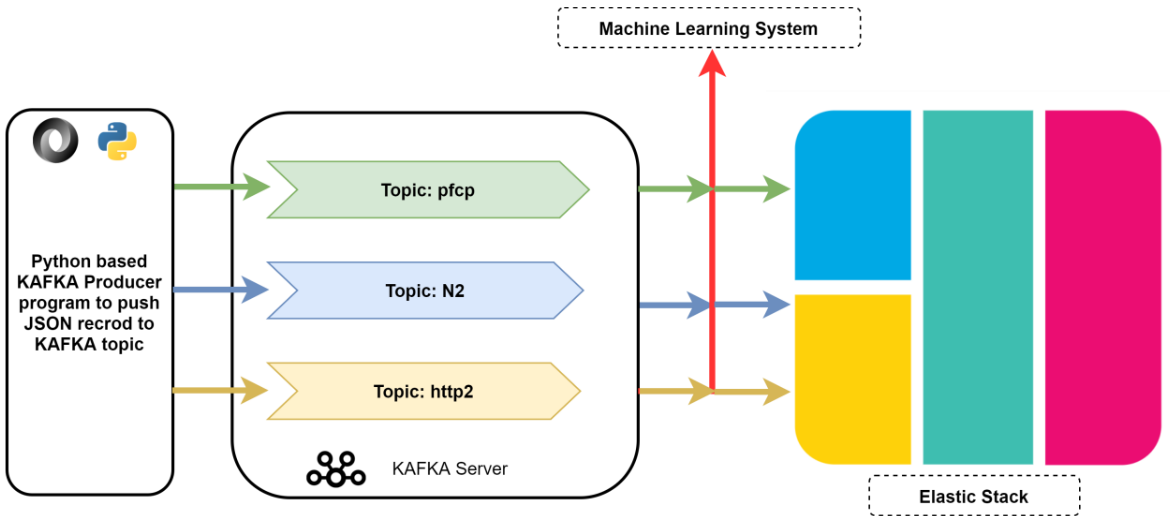
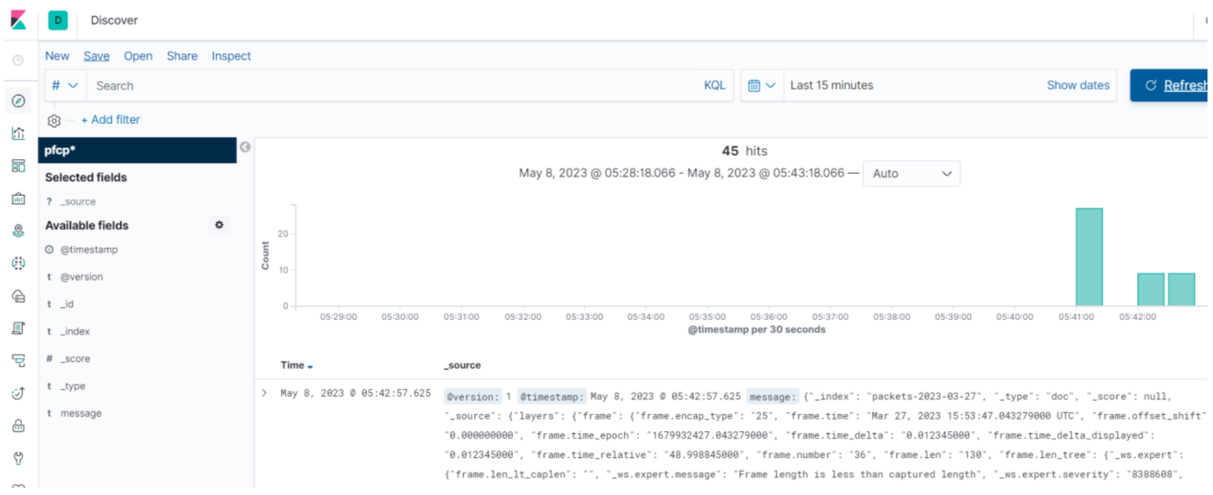


Figure 8. Data Loading into ELK

The purpose is to create an environment in parallel to the ML system so that data can be searched and analyzed. The Logstash has Kafka consumer plugin to ingest the data from each Kafka topic. The ingested data will be processed and loaded to the elastic search after implementing the required transformations and enhancements. As the data reaches elastic search then it is indexed so that user can login to the Kibana for searching and viewing the processed data. It should be possible to create the dashboard on Kibana for further data visualization. The data published from Logstash to the elastic search is seen in the Index Management of the elastic search application. The next step is to make the data available to Kibana, and this is achieved by creating an index inside Index Patterns in the Kibana setting. Figure 9 shows the “pfcp” received data in the Kibana GUI after the creation of the index, and this data can be searched and explored further based on the needs and requirements. It is possible to create the monitoring dashboard in Kibana based on the received data. The dashboard helps to understand the network traffic per protocol and monitoring of the network functions in real-time, and it enables the operator to protect the network against any cybersecurity attack.



```
Time      _source
> May 8, 2023 @ 05:42:57.625   @version: 1 @timestamp: May 8, 2023 @ 05:42:57.625 message: {"_index": "packets-2023-03-27", "_type": "doc", "_score": null,
"_source": {"layers": {"frame": {"frame.encap_type": "25", "frame.time": "Mar 27, 2023 15:53:47.043279000 UTC", "frame.offset_shift"
"0.000000000", "frame.time_epoch": "1679932427.043279000", "frame.time_delta": "0.012345000", "frame.time_delta_displayed":
"0.012345000", "frame.time_relative": "48.998845000", "frame.number": "36", "frame.len": "130", "frame.len_tree": {"_ws.expert":
{"frame.len_lt_caplen": "", "_ws.expert.message": "Frame length is less than captured length", "_ws.expert.severity": "8388608",
```

Figure 9. PFCP data analysis representation in Kibana

5. Discussion

Although we have not implemented a machine learning or artificial intelligence system to deliver intelligence in the 5G core networks, the design of a UDSF framework is the key precursor to further development and implementations of the NWDAF (Network Data Analytics Function). There can also be different ways to design a UDSF architecture and this is highly dependent on the demand of the vertical for which the 5G system is being utilized, the customer requirements as well as the toolset used for that purpose.

6. Conclusion

In this research, we have demonstrated a successful implementation of the UDSF 5G function, which aims to deliver data collection and processing for facilitating monitoring operations. Such data can be utilized in different ways, including the troubleshooting of the core networks, cybersecurity purposes or even providing an intelligent way of autonomous self-management based on the performance output and other metrics. It is important to note that this approach focuses on the control plane data that flows in the 5G core, avoiding the use of user plane information and retaining the confidentiality and data integrity of the end users.

References

[1] 3rd Generation Partnership Project (3GPP), "Technical Specification TS 23.501: 5G; System Architecture for the 5G System (5GS)", 2022 v17.5.0, (Online). URL: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/17.05.00_60/ts_123501v170500p.pdf

[2] European Telecommunications Standardization Institute (ETSI), "Technical Specification TS 129 500: 5G; 5G System; Technical Realization of Service Based Architecture; Stage 3", 2022, v16.10.0. (Online) URL: https://www.etsi.org/deliver/etsi_ts/129500_129599/129500/16.10.00_60/ts_129500v161000p.pdf

[3] Lin, X. and Lee, N. "5G and Beyond: Fundamentals and Standards", Springer, 2021.

[4] 3rd Generation Partnership Project (3GPP), "Technical Specification TS 38.401: 5G; NG-RAN; Architecture Description", 2020. (Online) URL: https://www.etsi.org/deliver/etsi_ts/138400_138499/138401/16.03.00_60/ts_138401v160300p.pdf

[5] European Telecommunications Standardization Institute (ETSI), "Technical Specification TS 129 244: LTE; 5G; Interface between the control plane and the user plane nodes", 2020, v16.5.0. (Online) URL: https://www.etsi.org/deliver/etsi_ts/129200_129299/129244/16.05.00_60/ts_129244v160500p.pdf

[6] European Telecommunications Standardization Institute (ETSI), “Technical Specification TS 129 598: 5G; Unstructured Data Storage Services”, 2020. v16.1.0, Release-16. (Online) URL: https://www.etsi.org/deliver/etsi_ts/129500_129599/129598/16.01.00_60/ts_129598v160100p.pdf

[7] The Linux Foundation, “Kubernetes”. (Online) URL: https://kubernetes.io/

[8] Apache Software Foundation, “Apache Kafka”, (Online) URL: https://kafka.apache.org/

[9] Dzogovic, B., Santos, B., Feng, B., Do, V.T., Jacot, N., Van Do, T. (2021). Optimizing 5G VPN+ Transport Networks with Vector Packet Processing and FPGA Cryptographic Offloading. In: Bentahar, J., Awan, I., Younas, M., Grønli, TM. (eds) Mobile Web and Intelligent Information Systems. MobiWIS 2021. Lecture Notes in Computer Science(), vol 12814. Springer, Cham. https://doi.org/10.1007/978-3-030-83164-6_7

[10] EURECOM, “OpenAirInterface5G – 5G Software Alliance for Democratising Wireless Innovation”. (Online). URL: https://openairinterface.org/

[11] The OpenJS Foundation, “jQuery”. (Online) URL: https://jquery.com/

[12] Elasticsearch B.V, “Elastic Stack”. (Online) URL: https://www.elastic.co/