

ALMA MATER STUDIORUM — UNIVERSITA DI BOLOGNA

SCHOOL OF ENGINEERING AND ARCHITECTURE

MASTER'S DEGREE

IN

TELECOMMUNICATIONS ENGINEERING

REDUNDANCY TECHNIQUES FOR
5G ULTRA RELIABLE LOW LATENCY
COMMUNICATIONS

Master Thesis

in

Laboratory of Networking M

Supervisor

Prof. WALTER CERRONI

Candidate

ONUR ÖZENIR

Co-supervisor

DAVIDE BORSATTI

SESSION III

ACADEMIC YEAR 2020/2021

Contents

<i>Abstract</i>	<i>vi</i>
1 Introduction	1
2 5G System Architecture and Session Management	6
2.1. 5GS Architecture Models and NFs	6
2.1.1. Architecture Models	6
2.1.2. Control / User Plane Separation – CUPS in 5GS	9
2.1.3. NG – RAN	10
2.1.4. NFs and Interfaces	11
2.2. SMF and UPF	14
2.2.1. SMF—Session management function	14
2.2.2. SMF Services	15
2.2.3. UPF —User plane function	17
2.2.4. PFCP – Packet Forward Control Protocol	19
2.2.5. Node Related Procedures	21
2.2.6. UPF Selection and Discovery	23
2.3. Session Management Procedure	24
2.3.1. PDU Session Concept and PDU Session Types	24
2.3.2. N4 Session Management Procedures	27
2.4. User Plane Management	30
2.4.1. GTP – GPRS Tunneling Protocol	30
2.4.2. Packet Forwarding Model	35
2.5. System Procedures	39
2.5.1. Registration Procedure	39
2.5.2. Service Request Procedure	40
2.5.3. Session Management Procedures	42
3 Ultra-Reliable Low Latency Communication – URLLC	47
3.1. Support for Ultra-Reliable Low Latency Communication	48
3.2. Dual Connectivity based end to end Redundant User Plane Paths	48
3.3. Support of redundant transmission on N3/N9 interfaces	50
3.3.1. N3/N9 GTP – U Tunnel Setup	52

3.3.2.	Redundant Transmission support only on N3 Interface	53
3.3.3.	Redundant Transmission support on N3/N9 Interface	54
3.4.	Support for redundant transmission at the transport layer	56
4	<i>Open-source Software Tools and Testing Environment</i>	60
4.1.	5G AN Software Tool – UERANSIM	60
4.1.1.	Overview	60
4.1.2.	Configuration	61
4.2.	5G CN Software Tool – Open5GS	63
4.2.1.	Overview	63
4.3.	Testing Environment	65
4.3.1.	VMs and Network Configurations	65
4.3.2.	Setting to Work	71
4.3.3.	PDU Session and Data Connection	76
5	<i>Implementation and Proof of Concept – Redundant Transmission Support on N3 Interface Technique in 5G CN</i>	83
5.1.	Motivation	83
5.2.	Implementation	86
5.3.	Proof of Concept	91
5.4.	Testing and Measurements	95
6	<i>Conclusion</i>	100
	<i>Appendix A - Ryu Controllers Code Files</i>	102
A.1	Packet Duplication Block Controller	102
A.2	Destination Address Modification Block Controller	105
	<i>Bibliography</i>	108
	<i>List of Figures</i>	110

Abstract

The 5G Core Network architecture is modeled to include instruments that can establish networks built on the same physical infrastructure but serve different service categories for communication types with varying characteristics. Relying on virtualization and cloud technologies, these instruments make the 5G system different from previous mobile communication systems, change the user profile, and allow new business models to be included in the system.

The subject of this thesis includes the study of Ultra-reliable low latency communication, which is one of the fundamental service categories defined for the 5G system, and the analysis of the techniques presented in 3GPP's Release 16, which enhance the service parameters by modifying the core network. In the theoretical part, the 5G system and URLLC are introduced with a particular focus on the user plane on the core network. In the implementation part, redundant transmission support on the N3 interface, one of the techniques presented in the technical specification, is modeled using open source software tools (Open5GS and UERANSIM) and network virtualization instruments. As a result of the tests and measurements performed on the model, it was observed that the implemented technique enhanced the system's reliability.

1 Introduction

The previous generations of mobile communication systems were human oriented. With the development through generations, the service features offered to the users such as bandwidth, delay, reliability improved gradually. In the 5G System – 5GS, the target is a bandwidth of 10Gbps and a 1msec delay. However, the concepts and technologies introduced for the core network architecture distinguish the 5GS from previous generations of mobile communication systems. Control/User Plan Separation – CUPS and Network Slicing are crucial enablers among these concepts. In 3GPP's Release 15, CUPS is introduced as a complete separation between control plane functions and user plane functions for the 5GS. Hence, the opportunity to design and deploy more flexible and scalable networks for their services occurs for the telecom operators.

On the other hand, Network Slicing promises to establish different logical (virtual) networks, also known as "Network slices," on the same physical network infrastructure. Each network slice can be designed to serve different services or users. It means that the same core network can be utilized to satisfy various characteristics and requirements of different services. The implementation of CUPS and Network Slicing concepts in the 5GS relies on integrating Network Function Virtualization, Software Define Network, and Cloud-Native technologies into the core network design. These technologies can be summarized as follows.

Network Function Virtualization – NFV is a network architecture concept where functionalities of network elements are virtualized in building blocks with the aid of IT virtualization technologies. These building blocks, called Virtual NFs, are implemented in virtual machines or containers running on different software.

Software-Defined Networking – SDN is a network architecture approach that enables the network's intelligence to be programmable with the software applications; therefore, it enables the control mechanism to be centralized and re-configurable.

Cloud-Native architecture refers to the extension of benefits of Cloud Computing throughout mobile communication networks, particularly the 5GS. The 5G Core Network – 5G CN is defined as Service Based Architecture – SBA. The interaction of network elements that act as a web-scale application relies on the Server / Client (Request / Response) paradigm.

Utilizing these technologies aims to make the 5G core network more flexible, agile, and scalable. Three service categories have been introduced in the 5G System based on the network slice concept.

Enhanced Mobile Broadband – eMBB slice aims to serve the end-users that provide improved mobile communication characteristics where downlink – DL speed is 1Gbps indoor and 300 Mbps outdoor in densely populated metropolitan areas centers. The service is typically Internet access for web browsing, messaging applications, or video streaming.

Massive Machine Type Communication – mMTC slice, at which the user profile changes, aims to provide a convenient architecture for Machine to Machine–M2M or IoT applications. It is proposed to provide narrow band access such devices like sensors, actuators, etc.

Ultra-Reliable and Low Latency Communications - URLLC slice is dedicated to mission-critical applications for which both high reliability and low latency are essential. Verticals like factory automation, V2X Communication, or remote surgery could be examples of mission-critical applications.

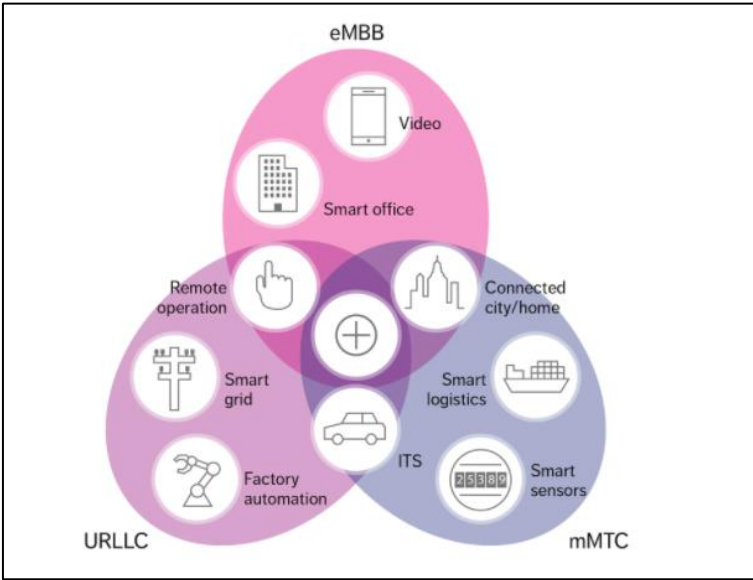


Figure 1-1 Service Categories in 5GS [1]

Defined network slices (or service categories) added features to the 5G system, such as adding 'things' to the user profile or satisfying the reliability requirement of mission-critical applications. These features allow new business models, called verticals, to emerge or existing business models for integrating into the 5G system [2].

URLLC has the most stringent design requirements due to the demanding service requirements among the three service categories. Ultra-High (> %99.999) reliability and very low latency(< 1msec) are the requirements that URLLC defines as design parameters, and they are individually challenging to fulfill [3]. Besides, the inherent trade-off between reliability and latency makes URLLC design more complicated [4].

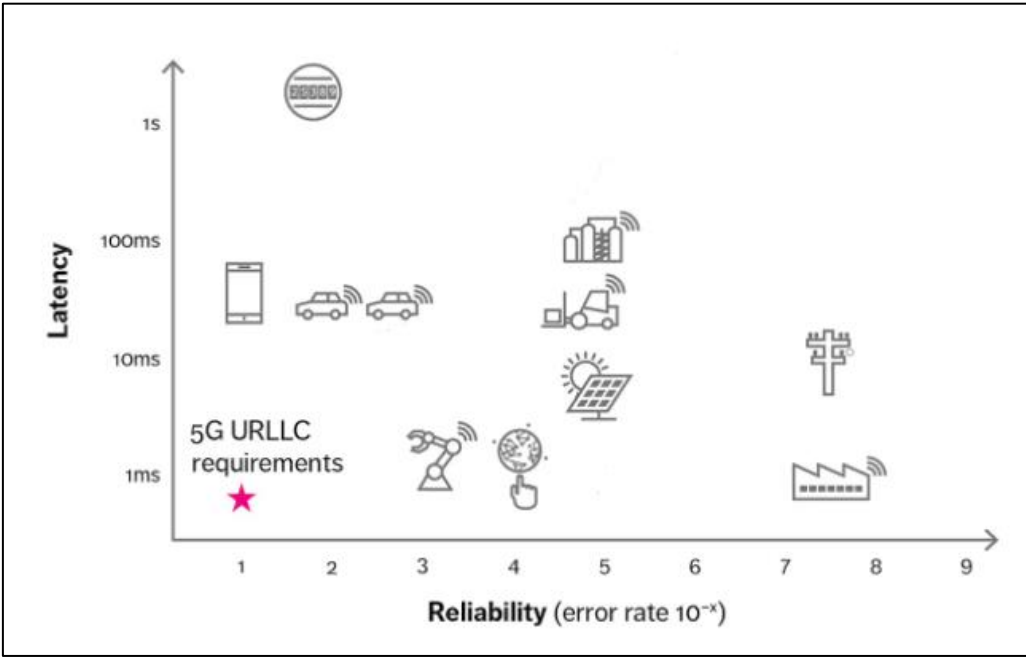


Figure 1-2 5G URLLC Requirements [1]

The scope of this thesis is the analysis of redundancy techniques for high-reliability communication introduced in 3GPP's Release 16 to enhance the 5G URLLC service. Proposed models aim to increase the reliability of the entire transport path (end to end) or a part of the transport path (core network). In this context, 5GS was simulated by using Open5GS for core network virtualization and UERANSIM for access network virtualization. Moreover, N3 support for redundant transmission technique, one of the design models specified in the 3GPP's Release 16, is implemented and integrated into the virtualized 5GS with the aid of SDN technologies.

This document is structured as follows; Chapter 2 is dedicated to giving technical background about the components of the 5GS that are within the scope of URLLC service. That is 5G CN and its elements. In addition, 5G Session Management procedure and PDU¹ Session concept are introduced, PDU Session Concept is detailed with a perspective of Control plane – CP and User Plane – UP protocols. Finally, the system procedures covered by URLLC are specified. In Chapter 3, URLLC use cases and requirements are specified. Redundant transmission support techniques in 3GPP's Release 16 are examined and evaluated. Chapter 4 includes a brief presentation about open-software tools used (UERANSIM and Open5GS) and the test environment for the implementation. Chapter 5 is dedicated to the implementation of the redundant transmission support on the N3 interface technique and test results.

¹ PDU stands for Protocol Data Unit. Session have the same meaning as PDU Session in 5GS. These terms are used interchangeably within the thesis.

2 5G System Architecture and Session Management

As indicated, the 5G CN utilizes the advent of new technologies in IT such as Network Virtualization or Software Defined Networking. These techniques are used to adapt the system architecture to fulfill the requirements of various services without modifying the physical infrastructure.

According to [5], the fundamental principles of the 5GS are summarized as follows.

- Provide Control Plane – User Plane Separation, enabling independent scalability, evolution, and flexible deployments.
- Modularize functionalities to enable flexible and efficient network slicing.
- Define procedures for the interactions between the network elements and the operations during the system services.
- Introduce NFs that interact with each other directly or indirectly on a server-client model paradigm.
- Lower dependencies as much as possible between the Access Network – AN and The Core Network – CN.
- Enable local and centralized services available simultaneously to support low latency services and access to local data networks, e.g., UP functions can be deployed close to the Access Network.

2.1. 5GS Architecture Models and NFs

5GS architecture models ease the definition of procedures for an operation or a function in the 5G System. NFs are the elements of the 5G CN. They consume the other services to perform the procedures such as Mobility Management or Session Management. Before going into details of the procedures, it is essential to introduce architecture models and NFs.

2.1.1. Architecture Models

5G architecture models are categorized in the Reference Point Architecture model and the Service – Based Architecture model. The first model is proposed for node-level procedures for NF interactions, while the second model is presented for service base interfaces and service-related interactions.

Reference Point Architecture: The representation of 5GS architecture involves a set of Network Elements (NE) and point-to-point interfaces, which are the interconnections between

Network Elements. The Control Plane interactions between the NEs, are defined as Signaling Procedures. They are specific for each point-to-point interface.

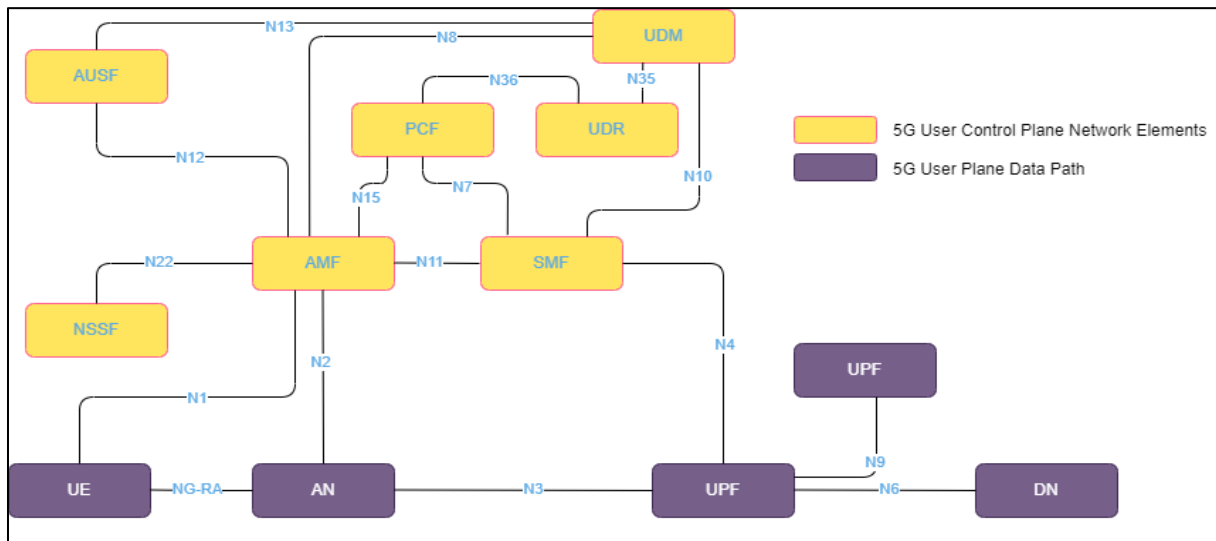


Figure 2-1 5G Reference Point System Architecture (Simplified Version) [5]

List of NFs presented in Figure 2.1.

- Authentication Server Function (AUSF),
- Access and Mobility Management Function (AMF),
- Data Network (DN), e.g., operator services, Internet access, or 3rd party services,
- Network Repository Function (NRF),
- Network Slice Selection Function (NSSF),
- Policy Control Function (PCF),
- Session Management Function (SMF),²
- Unified Data Management (UDM),
- Unified Data Repository (UDR).
- User Plane Function (UPF),
- User Equipment (UE),
- (Radio) Access Network ((R)AN).

List of reference points presented in Figure 2.1.

- **N1:** The Reference point between the UE and the AMF,
- **N2:** The Reference point between the (R)AN and the AMF,
- **N3:** The Reference point between the (R)AN and the UPF,
- **N4:** The Reference point between the SMF and the UPF,
- **N6:** The Reference point between the UPF and a Data Network,
- **N7:** The Reference point between the SMF and the PCF.
- **N8:** The Reference point between the UDM and the AMF
- **N9:** The Reference point between two UPFs,
- **N10:** The Reference point between the UDM and the SMF.
- **N11:** The Reference point between the AMF and the SMF.
- **N12:** The Reference point between AMF and AUSF.

² While here providing the list of NFs and reference points, the detailed explanations of the related NFs, Node – Level interactions, NF Service–based interfaces and NF Services will be provided in the following sections.

- **N15:** The Reference point between the PCF and the AMF
- **N22:** The Reference point between AMF and NSSF.
- **N35:** The Reference point between UDM and UDR.
- **N36:** The Reference point between PCF and UDR.

Service-Based Architecture: The representation of 5GS architecture uses NFs instead of NEs and replaces the point-to-point interfaces with a common bus, enabling the one NF to utilize the service(s) provided by other NF(s).

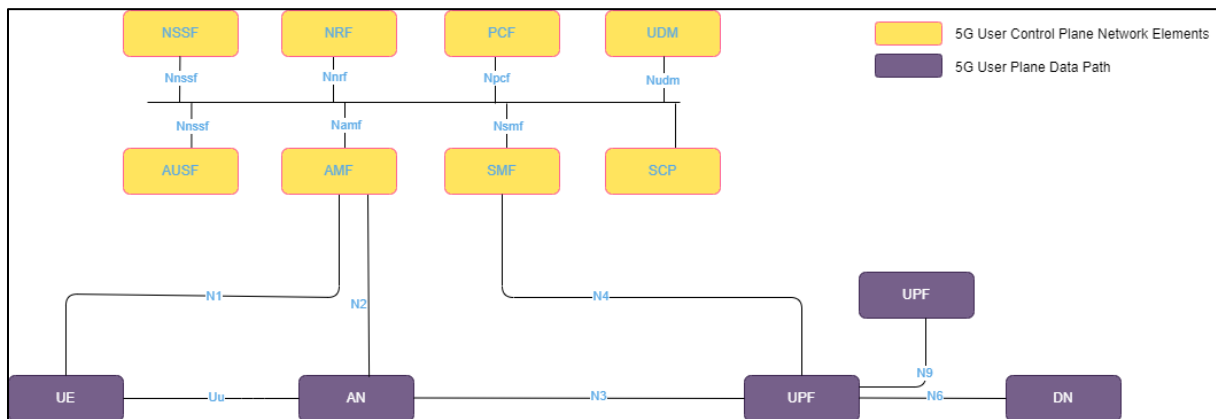


Figure 2-2 5G Service-Based System Architecture (Simplified Version) [5]

5GS Architecture for Non-Roaming Case

The term non-roaming refers to the case in which a user-end is always in the range of its native network. And the connected cell network does not change along with the duration of the service. Since the thesis focuses on the URLLC techniques for non-roaming cases, it is convenient to provide overall system architecture, including corresponding NFs and interfaces. The following figure illustrates the 5GS architecture for this use case.

5GS Architecture can be divided as access network – AN and core network – CN. The access network elements are the User End – UE and the next generation Node B – gNB; NG – RAN (New Generation Radio Access) is the term used for the combination of them. 5G CN involves a set of NFs as shown in reference architecture models. Prior to describing NFs, a brief discussion is essential about the NG – RAN architecture and protocols in the access network.

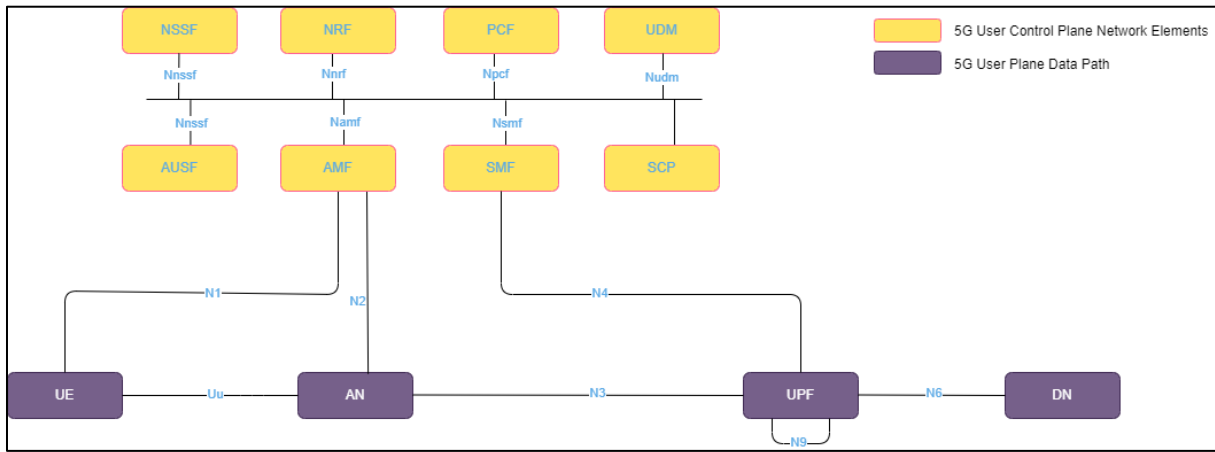


Figure 2-3: 5GS Architecture for Non-roaming case (Simplified Version) [5]

2.1.2. Control / User Plane Separation – CUPS in 5GS

The 5G system architecture is structured to enable the decoupling between the CP and the UP. CUPS applies both the AN and the CN. With the CUPS concept, the dependence between the AN and CN is eliminated, which yields that the network portions can be optimized independently. The control and user data messages are transported over different interfaces, therefore encapsulated by different protocol headers. In AN, user/control plane separation is achieved by allocating distinct radio resources [6].

On the other hand, dedicated transport paths are deployed in CN to apply CUPS. 5G CN functionalities are modularized and distributed to NFs. In addition, the CUPS concept supports various deployment options within the CN. While the centralized approach can be used for the Control Plane's network deployment, the User Plane network deployment can be implemented in various locations within the 5GS. For example, the User Plane process can be handled close to the UE within the services with stringent delay requirements (e.g., URLLC services). The list below shows the NFs' functionalities for main procedures in 5GS.

Main Function		AMF	SMF	UPF
UE Registration	Control Plane Functions	X		
UE Mobility		X		
UE Session Management			X	
QoS for User Plane			X	
User Plane Transport Layer (GTP – U) Establishment	User Plane Functions			X
User Data Processing				X

Table 2-1 NF Functions³

³ The AMF is described in this section. The SMF and the UPF are described in Section 2.2.

2.1.3. NG – RAN

5GS NG – RAN is depicted in Figure 2.4. The air interface is called NG – RA; this is the path between the UE and the gNB. The air interface carriers (radio resources) are called radio bearers. The Xn interface is used for the interconnection of the gNBs. In some use cases, such as Dual Connectivity, more than one gNB may serve a specific UE. The control messages between gNBs are exchanged over the Xn interface to operate in coordination (e.g., PDCP protocol packet duplication and elimination functions require coordination between the gNBs.). Moreover, network elements of NG – RAN have interfaces to interact with the AMF in the CN. These interfaces are used to exchange only control messages for any service in the 5GS. The AMF interacts with the UE over the N1 interface and the gNB over the N2 interface. Finally, the interface between the UE and the UPF that is established to carry application data packets (Uplink – UL and Downlink – DL packets) is the N4 interface.

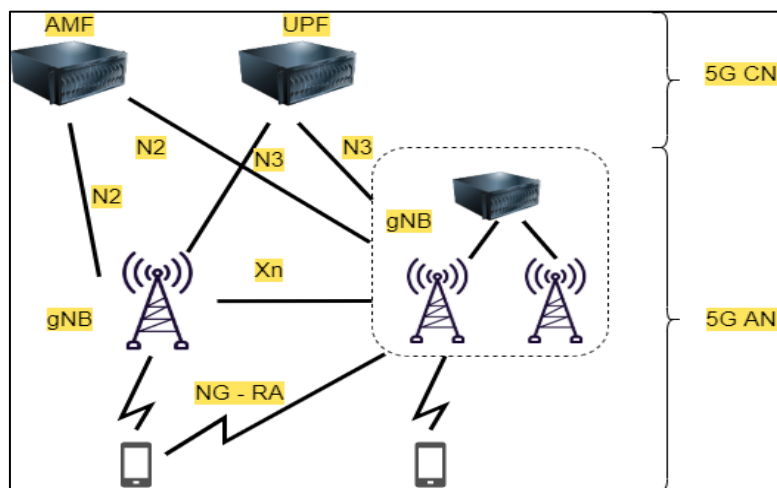


Figure 2-4 NG – RAN in 5GS

Due to CUPS, the CP and the UP protocols are different. The UP protocol stack of NR consists of RLC, PDCP, and SDAP. A new Access Stratum – AS layer is also used to carry SDAP packets above the PDCP.

RLC – Radio Link Control layer performs error correction, MTU segmentation, and re – segmentation as well as the transfer of upper layer PDU payloads.

PDCP – Packet Data Convergence Protocol has such functions as transfer of user data (e.g., application data), header compression, packet duplication/elimination. PDCP control sequence numbering with the RLC.

SDAP – Service Data Adaptation Protocol manages the correspondence between the data radio bearers and a Quality of Service – QoS.

Physical – PHY and Media Access Control – MAC are the same for the user and control planes. The former provides transmission of information over the air, while the former maps the logical channels to transport channels and manage RLC PDUs multiplexing and demultiplexing. RRC protocol and NAS layer are present in the control plane differently than the user plane.

RRC – Radio Resource Control handles the radio resources management by performing radio bearer establishment, modification, and release. The PDCP header encapsulates RRC control messages.

NAS – Non-Access Stratum is a functional layer that involves two basic protocols to support functionalities of the 5GS. These protocols are the 5GS Mobility Management (5GMM) and the 5GS Session Management (5GSM)⁴. NAS layer is utilized the control messages between the UE and the 5G CN.

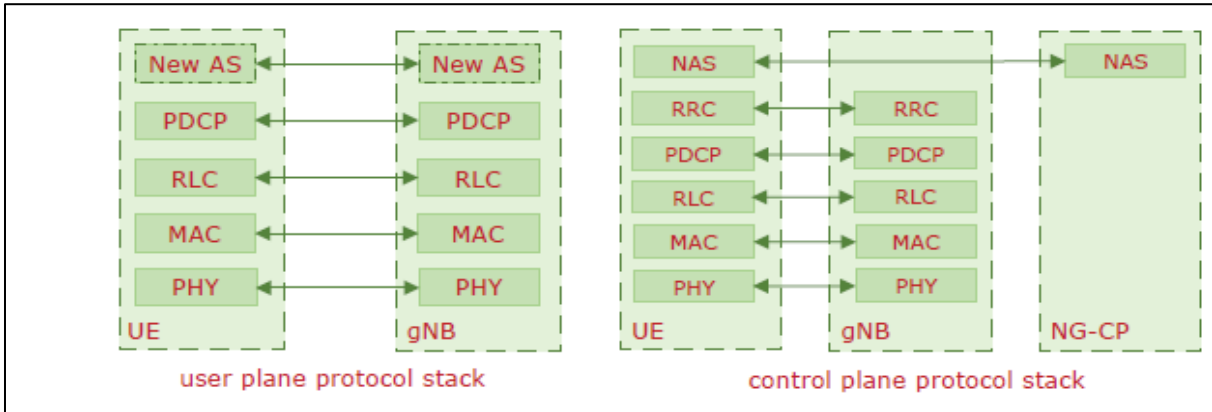


Figure 2-5 User Plane and Control Plane Protocol Stacks in NG – RAN [4]

2.1.4. NFs and Interfaces

This section describes NFs and NF Services introduced in the 5G CN for non-roaming cases. In 5GS, CN elements, called NF, are logical nodes that are programmable to provide services within the system. The CN elements are dedicated physical devices with specialized software and hardware before 5GS.

The interaction between NFs relies on the Client-Server paradigm with Request/Response messages. The NF requests a service called "Service – Consumer"; the NF provides the service

⁴ The payload of 5GSM messages is referred as N1/N2 SM Container.

is called "Service Producer". When the Service – Consumer is willing to consume a specific service, it sends a Request message to the Service Producer, possibly including all related parameters, actions, or information. Then, it waits for the Service Producer to process the request and gives feedback in a Response message. Service Request/Response messages are exchanged over reference points respecting the Service Consumer and Service Producer.

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

AMF—Access and mobility management function⁵

The AMF handles all signaling messages between NG – RAN and 5G CN in general terms. Its main functionalities cover all the procedures for user access and user mobility. The AMF manages the authentication and authorization functionality in cooperation with AUSF and UDM. In Session Management procedures, it forwards the UE request/response to the SMF. It does not play an active role in the management of the User Plane.

The AMF related reference points and service-based interface are

- The N1 reference point is used for all control messages to be exchanged with the UE,
- The N2 reference point is used for all control messages to be exchanged with the gNB,
- The N8 reference point is used to gather user subscription data from the UDM during registration operation,
- The N11 reference point is used for the session control messages with the SMF,
- The N12 reference point is used for the authentication of a user during registration operation,
- The N15 reference point is used for the policy control messages with the PCF,
- The Namf service-based interface is used to access the services provided by the AMF.

⁵ The services of the AMF are out of the context of the thesis.

AUSF—Authentication server function

The AUSF service is handling the authentication procedures in the core network over the Nausf service-based interface. The service requires collecting information from the UE and the UDM to provide security parameters in the UE Registration procedure.

NRF—Network repository function

The NRF serves as a repository for the Service Consumers in the core network over the Nnrf service-based interface. The Service Consumers (e.g., AMF) are only required to be pre-configured with the parameters to discover NRF. The Service Consumer obtains from the NRF about the other NFs. The NRF keeps and provides NF-related data involving some properties such as NF Type and supported NF services and IP Address for discovery and selection procedures.

UDR—Unified data repository⁶

The UDR serves as a database for the subscription data of the UEs over the Nudr service-based interface. It also stores user policies and network policies. The UDM and the PCF access these data by consuming the services of the UDR.

UDM—Unified data management function

The UDM provides its services over the Nudm service-based interface. The UDM services are

- The execution of UE authorization for access to the DN
- The generation of the authentication credentials that are transferred to the AUSF,
- The management of permanent identity privacy. NFs consumes the UDM service to resolve the concealed permanent identity (SUCI) to the real permanent identity (SUPI),
- To keep track of the AMF instances for related UE,
- To keep track of the SMF instances for related PDU Sessions.

PCF—Policy control function

The Npcf is the service-based interface of the PCF. The PCF produces the service which enables to control of UE policies in the Session Management Procedure. The PCF provides information for UE access selection or PDU Session Selection procedures.

⁶ SUCI and SUPI are defined in Chapter 4.

NSSF—Network slice selection function

The NSSF service provides the necessary information to the Service Consumer to select the (set of) network slice instances for a particular UE. Besides, it gives information about AMFs that should serve the UE. The Nssf is the service-based interface of the NSSF.

The main functionalities within the 5GS procedures of the AMF and the SMF are presented in *Table 2-1*. During these procedures, the AMF and the SMF need to cooperate with the other NFs. In other words, the AMF and the SMF consume the services provided by other NFs along with the procedures they are responsible for. *Figure 2-6* depicts the interactions among NFs over service-based interfaces and corresponding protocols. The detailed information for the SMF (and the UPF) is given in the next section.

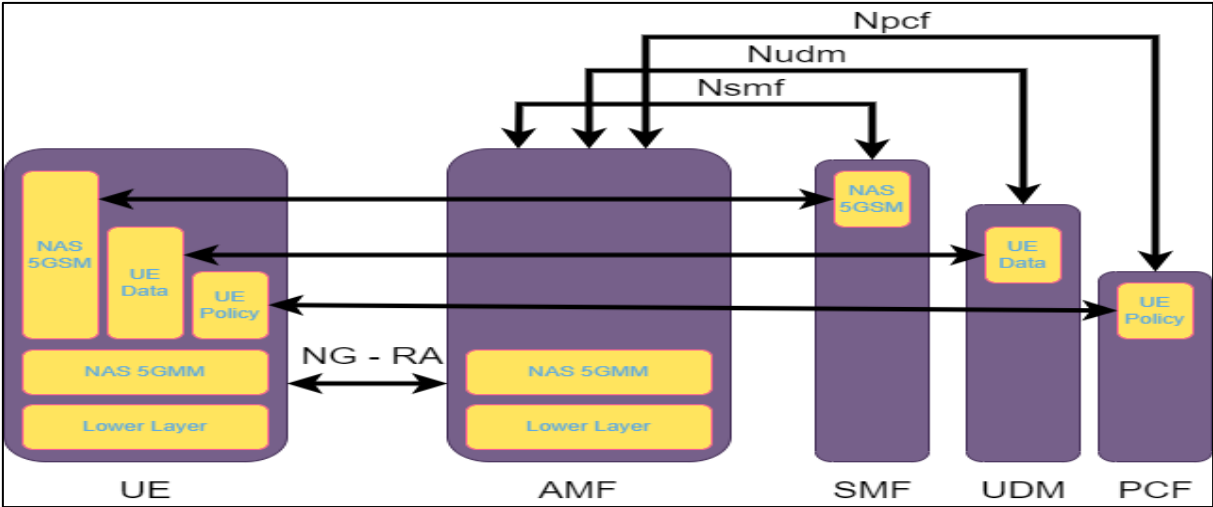


Figure 2-6 5G CN Service-Based Interfaces and services consumed by the AMF

BSF – Binding Support Function

The BSF stores the binding information for a particular PDU Session and discovers the selected PCF according to the binding information. The BSF allows PCFs to register, update, and remove the binding information, allowing NF consumers to find the chosen PCF. The BSF can be deployed standalone or collocated with other NFs, such as PCF, UDR, NRF, and SMF.

2.2. SMF and UPF

2.2.1. SMF—Session management function

In the 5G CN, the SMF handles all Session Management related procedures. The Session Management functionalities are mainly twofold: the setup of the connection between the UE and the DN and the management of the User Plane for the established connection. Once the AMF sends the PDU Session Request, the SMF performs the Session Establishment procedure

in cooperation with the UDM and the PCF. Moreover, the SMF instructs the UPF for the rules that apply to the User Plane Data. These rules are sent with the control messages of Packet Forwarding Control Protocol – PFCP⁷. The SMF interacts with the UPF over N4 Reference Point. It uses N4 to control functions of the UPF(s). Moreover, the messages exchanged for UPF discovery and selection procedures are carried over N4.

The SMF service-based interface, Nsmf, is used by other NFs to access the SMF services. The SMF functions are as follows.

- The control over user sessions, including establishment, modification, and release of sessions,
- The allocation of IP address(es) to UE(s) for IP PDU sessions,
- The management of service and session continuity for various end-user protocol types,
- The configuration of the UPF for a PDU session, in the manner of policies in cooperation with PCF.

2.2.2. SMF Services

The SMF provides two types of services: the Nsmf_PDUSession service and the Nsmf_EventExposure service⁸. The SMF produces the former in order to manage the PDU Sessions, while the latter is to expose events from the SMF. The consumer of the Nsmf_PDUSession service could be the AMF or another SMF⁹. The roaming case could be one of the use cases involving extra SMF insertion to the CN.

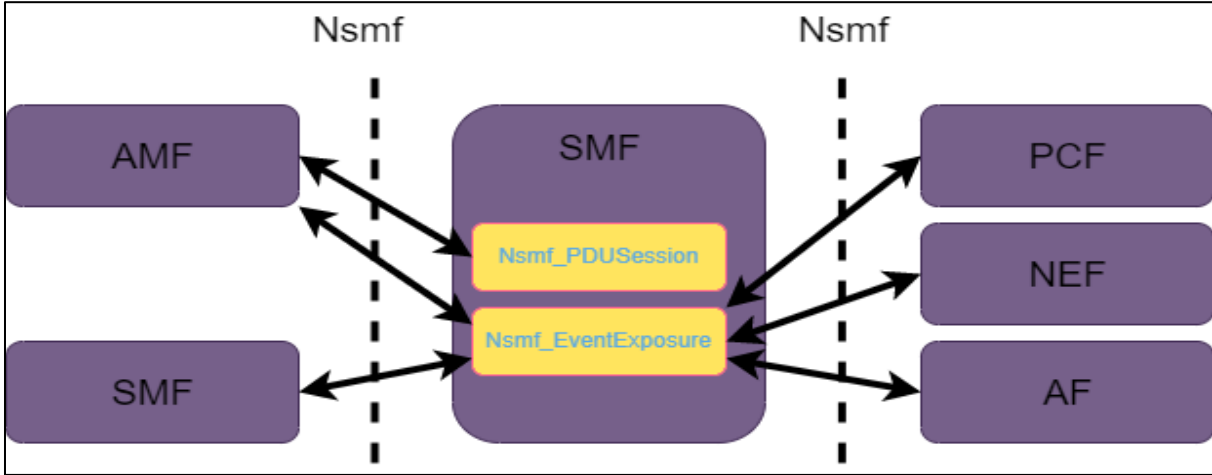


Figure 2-7: SMF Services and Service Consumers

⁷ The details of PFCP are presented in Section 2.2.4.
⁸ Nsmf_EventExposure services are out of the context of the thesis.
⁹ Another SMF refers to one of these: I – SMF, V – SMF, or H – SMF.

Service Name	Service Operations	Operation Semantics
<i>Nsmf_PDUSession</i>	Create	Request/Response
	Update	Request/Response
	Release	Request/Response
	CreateSMContext	Request/Response
	UpdateSMContext	Request/Response
	ReleaseSMContext	Request/Response
	SMContextStatusNotify	Subscribe/Notify
	StatusNotify	Subscribe/Notify
	Context	Request/Response
<i>Nsmf_EventExposure</i>	Subscribe	Subscribe/Notify
	Unsubscribe	
	Notify	

Table 2-2 List of SMF Services [7]

Nsmf_PDUSession_CreateSMContext Service

- The AMF consumes the service to establish an association with the SMF.
- The request message must include the PDU Session parameters SUPI or PEI, DNN, S-NSSAI(s), PDU Session ID, AMF ID (AMF Instance ID), N1 SM container.
- The input message may include optional parameters or additional information such as Subscription for PDU Session Status Notification, indicating that the SUPI has not been authenticated.
- The response message must be an indication as a result of the service.
- The response message may include optional parameters or additional information such as Cause, PDU Session ID, N2 SM information, N1 SM container.

Nsmf_PDUSession_UpdateSMContext Service

- The AMF consumes the service to update the established AMF-SMF association to support a PDU Session and provide SMF N1/N2 SM information received from the UE or the AN.
- The request message must include SUPI.
- The request message may consist of optional parameters or additional information such as PDU Session ID, N1 SM container received from the UE, N2 SM information obtained from the AN.
- The response message must include an indication as a result of the service.

- The response message may include optional parameters or additional information PDU Session ID, CauseN2 SM informationN1 SM container to be transferred to the AN/UE.

Nsmf_PDUSession_ReleaseSMContext Service

- The service is used by the AMF to release the AMF-SMF association for a certain PDU Session when the PDU Session is being released.
- The request message must include SUPI, PDU Session ID.
- The request message may include UE location information, AN type, UE Time Zone.
- The response message must include an indication as a result of the service.
- The response message may include Cause.

Nsmf_PDUSession_SMContextStatusNotify Service

- The operation is used the SMF to notify the AMF when a PDU session is released.
- The request message must include SUPI, PDU Session ID.
- The request message may include Cause.
- The response message must include an indication as a result of the service.
- The response message may not include optional information or additional information

2.2.3. UPF —User plane function

Before 5GS, the architecture of the user plane is relatively fixed since the number and the duty of network elements in the CN are predetermined. As indicated, one of the main features of 5GS is that the CN architecture is flexible and scalable as well as modular. It means that depending on the user plane requirements, the architecture could be modified or scaled up and down. The NF that is responsible for the user plane operations is the UPF. The number of the UPFs along the user plane path is predefined in most use cases. For others, the SMF may determine a UPF to insert in or extract from the NF chain deployed in the CN. Finally, the SMF may decide to change the UPF for an ongoing PDU Session.

UPF Functionalities and Roles

The UPF receives the instructions from the SMF during the Session Establishment procedure. Instructions here refer to a set of rules and parameters regarding the Session to be established. The UPF manages the user plane data according to the rules and parameters sent in PFCP packets. Data management operations in the User Plane are called Packet Processing. The

instructions for the packet processing operations may be modified before or after the Session Establishment procedure. The functionalities of the UPF may vary depending on the role dedicated to it.

In some cases, the UPF may act as a firewall with drop instruction from the SMF, or in others, it may serve as a data bridge forwarding the UL and DL packets. The location of the UPF is also service-dependent. The UPF may be located close to the UE and provide low delayed connectivity with Cloud Computing technologies. The functionalities and the location of a UPF are selection parameters for the SMF. The general functionalities are given by [7] as follows.

- Anchor point for Intra-/Inter-RAT mobility,
- External PDU Session point of interconnecting to Data Network (i.e., N6),
- Packet routing and forwarding,
- Packet inspection (e.g., Application detection),
- UP part of policy rule enforcement, e.g., Gating, Redirection, Traffic steering,
- Lawful intercept (UP collection),
- Traffic usage reporting,
- QoS handling for the UP,
- Uplink Traffic verification (SDF to QoS Flow mapping),
- Transport level packet marking in the uplink and downlink,
- Downlink packet buffering and downlink data notification triggering,
- Sending and forwarding of one or more "end markers" to the source NG-RAN node,
- Functionality to respond to ARP and IPv6 ND requests for the Ethernet PDUs.

The terminology used for the UPF may vary depending on the role of the UPF in the user data path. It is worth mentioning two of them.

PDU Session Anchor – PSA: The UPF terminates the N6 interface toward the DN.

Intermediate UPF – I UPF: The UPF is in the middle between the (R)AN and a PSA. It forwards UL and DL packets between (R)AN and the PSA.

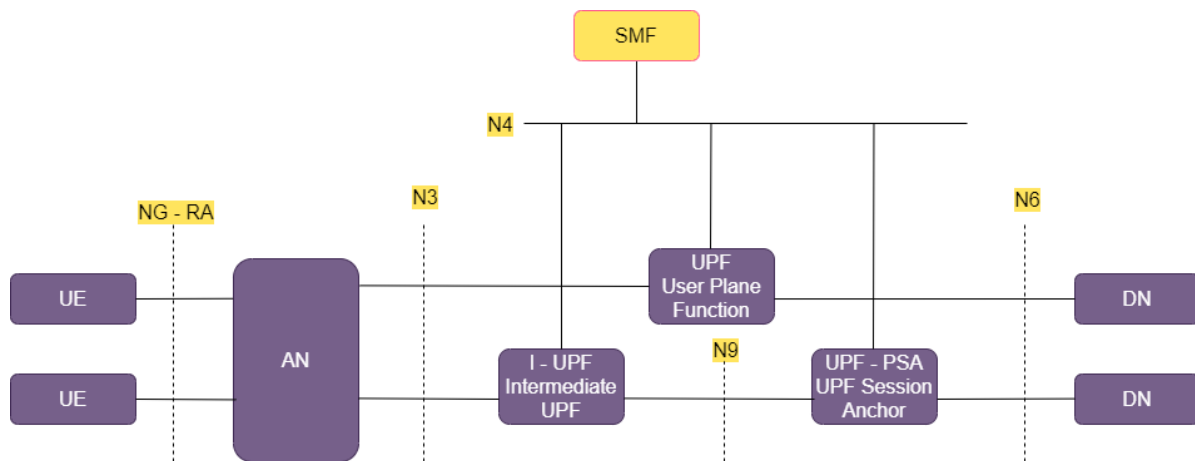


Figure 2-8 User Plane Path for one UPF and two UPFs cases

2.2.4. PFCP – Packet Forward Control Protocol

In the 5GS, Packet Forward Control Protocol (PFCP) is one of the main protocols introduced by 3 GPP. However, it can also be used in 4G/LTE EPC Systems [8]. It is proposed to implement CUPS in the CN. The protocol manages the exchange of the control messages over the N4 reference point between the CP Function (the SMF) and the UP Function (the UPF). CP Function decides the operations be applied to the user data. UP Function performs the operations respecting the instructions received from it. As mentioned before, the instructions are rules and parameters that apply to the specific data flow.

Prior to PDU Sessions' establishment, the SMF should select the UPF(s) utilized in the User Plane to provide communication services to the UE¹⁰. The selection is achieved by the control messages exchanged during the association procedure. Moreover, PFCP supports additional node-related procedures such as node-level configuration and load/overload handling. The session-related procedures are operated to establish or manage the PDU Session(s) when the association is established. PDU Sessions can be triggered either by the UE or the CN¹¹.

PFCP Protocol Stack

While the format of the PFCP message may vary depending on whether it is Node-related and Session-related, all of which contain a header (PFCP message header) and additional Information Element(s). Session-related messages should include SEID (Session Identifier) stored in the SMF and the UPF. The SEID is utilized to map the messages with the related PDU Session.

¹⁰ The UPF discovery and selection procedure is described Section 2.2.6.

¹¹ The UE triggered PDU Session Establishment Procedure is detailed in Section 2.5.1.

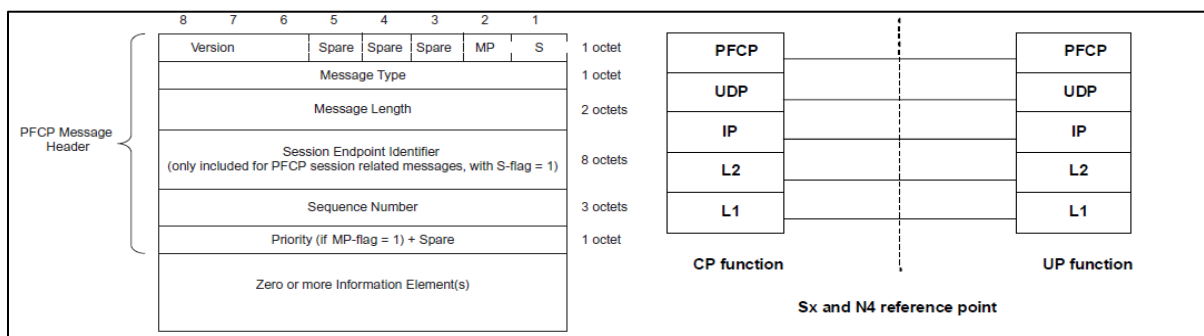


Figure 2-9 PFCP Protocol Stack [8]

Some essential points are highlighted here.

- The current PFCP Protocol only supports version 1.
- The S-flag should be set to 1 if the message includes a SEID. This is the flag that defines the type of PFCP message. It is either a node-related or a session-related.
- MP Flag can be used to declare the priority of the message.
- If it is not required, IE(s) may not be placed in the PFCP message.
- PFCP runs over the N4 interface¹² on the UDP/IP, as depicted in Figure 2 -8. The version of the IP could be IPv4 or IPv6; NFs should support both. UDP port number for the PFCP request Messages is 8805.

PFCP Procedures

In this Section, PFCP procedure types and related messages are summarized. 3GPP defines two kinds of PFCP procedures.

a. Node Related Procedures

These procedures refer to interactions between the SMF and UPF that are utilized to establish an association and control node-level communication and node-level configuration. In addition, they involve node-level reporting between the NFs. Node-related procedures are as follows [8].

- PFCP Association Setup procedure,
- PFCP Association Update procedure,
- PFCP Association Release procedure,
- PFCP Node Report procedure.

¹² PFCP procedures and N4 procedures refer to the same thing in the 5GS.

b. Session Related Procedures

PFCP Session procedures refer to the interactions between the SMF and the UPF to establish a new PDU Session or to manage an existing one. Session related procedures are as follows.

- PFCP Session Establishment procedure,
- PFCP Session Modification procedure,
- PFCP Session Deletion procedure,
- PFCP Session Report procedure.

2.2.5. Node Related Procedures

N4 Association Setup Procedure

The association between the SMF and the UPF is set up by this procedure. To establish an association, the NF should have already discovered the peer or related parameters for discovery in its configuration file. Once the procedure is successfully completed, the SMF can manage the UPF resources for the upcoming PDU Session. Both NFs may initiate the procedure. However, the default behavior is an initiation by the SMF. When the UPF receives the Request message, it should respond to the SMF. The Response message indicates whether the association request is accepted or rejected.

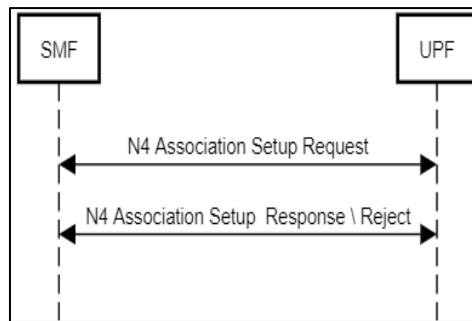


Figure 2-10 N4 Association Setup Procedure [9]

N4 Association Update Procedure:

The procedure modifies an existing association between the SMF and the UPF. The procedure could be initialized by both NFs and expect a response message from the peer.

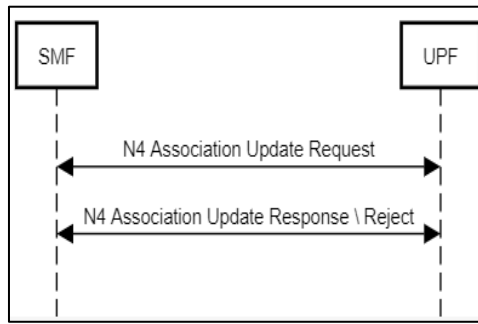


Figure 2-11 N4 Association Release Procedure [9]

N4 Association Release Procedure:

It is used to release an existing association between the SMF and the UPF. The procedure could be initialized by both NFs and expect a response message from the peer.

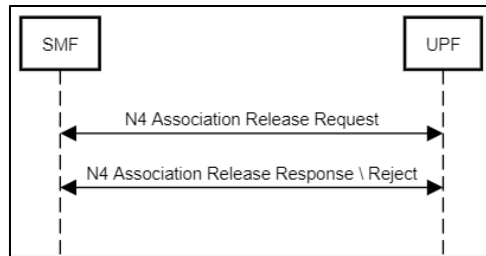


Figure 2-12 N4 Association Release Procedure [9]

N4 Association Report Procedure

It is used by the UPF to indicate the SMF when an event occurs in the UP path. The event refers here that not an event specific to a PDU Session but an event affecting all PDU sessions. Both peers need to insert an ID¹³ in control messages.

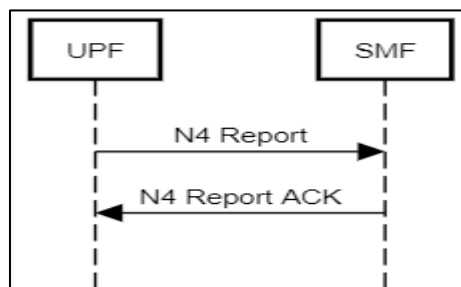


Figure 2-13 N4 Report Procedure [9]

¹³ These are the UPF ID in Report message, and the SMF ID in ACK message.

2.2.6. UPF Selection and Discovery

The SMF is responsible for selecting the UPF(s) in the 5G CN. The selection procedure is not standardized, and it depends on the implementation of the network operator. However, some criteria of the selection procedure can be generalized. In URLLC services, the functionality of packet duplication so the capability of the UPF could be decisive as well as the UPF and the UE location information¹⁴. Moreover, the SMF may not be aware of the full functionalities of a specific UPF, so it learns them during or after when it is associated with the UPF. The selection criteria could be one of or multiple of the following properties [7].

- UPF's dynamic load,
- UPF's relative static capacity among UPFs supporting the same DNN.
- UPF location.
- UE location information,
- The capability of the UPF,
- The functionality required for the UE session,
- Data Network Name (DNN),
- PDU Session Type (e.g., IPv4, IPv6, IPv4v6, Ethernet Type or Unstructured Type),
- SSC mode selected for the PDU Session,
- UE subscription profile in UDM,
- Local operator policies,
- S-NSSAI,
- Access technology used by the UE,
- Information related to user plane topology and user plane terminations.

Prior to UPF Selection, the SMF needs to discover the UPF. The discovery procedure could be predetermined or not. If the SMF is configured so that it has all the required information about where the UPF is and what its functionalities are, then the SMF may initialize the N4 Association procedure. If the SMF is not configured with such information, then it needs to query to the NRF to collect the information for the available UPF(s) in the CN. However, the NRF can provide limited information about the functionalities of the UPF, and the information may not satisfy the selection criteria of the SMF. In such cases, the SMF needs to request UPF(s) to provide such information in the Association Setup Request message.

¹⁴ Packet Duplication improves the reliability and the location information can be used to evaluate end to end delay. These are the fundamental requirements for a URLLC service. URLLC use cases are detailed in Section 3.

2.3. Session Management Procedure

The provision of data connectivity is one of the main tasks of the 5G System. As indicated in Section 2.2.1, the Session Management Procedure includes the management and control of the User Plane. The SMF orchestrates the procedure in coordination with the related NFs¹⁵. The Session Management procedure is not the same for all UEs connected to the 5G Network. The Network Slice on the top of the 5G CN that is wanted to utilize may vary, so the PDU Session types. Session Management should support these varying requirements of the different services, and it should set up and manage the data connectivity respecting these requirements. Session management requires periodic measurement (QoS flows) of communication parameters (e.g., delay) on the user plane, which define the type of the service. The SMF sets the UPF to perform this function and send reports about the ongoing PDU Session. In the 5GS, the SMF and UPF cooperate in order to satisfy QoS on the user plane path. The Information Element – IE Quality Flow Identifier – QFI in PDU DL/UL packets maps the PDU with the QoS, but it may differ for DL and UL packets in the same PDU [10].

2.3.1. PDU Session Concept and PDU Session Types

The common feature in mobile networks is separating the User Plane path for each PDU Session to provide fundamental functionalities of the mobile networks such as per-user security, mobility, QoS, etc. Tunneling protocols achieve the separation for the User Plane data over the transport layer. The tunneling mechanism enables the operators to deploy any transport technology independently from the user plane data (or application data).

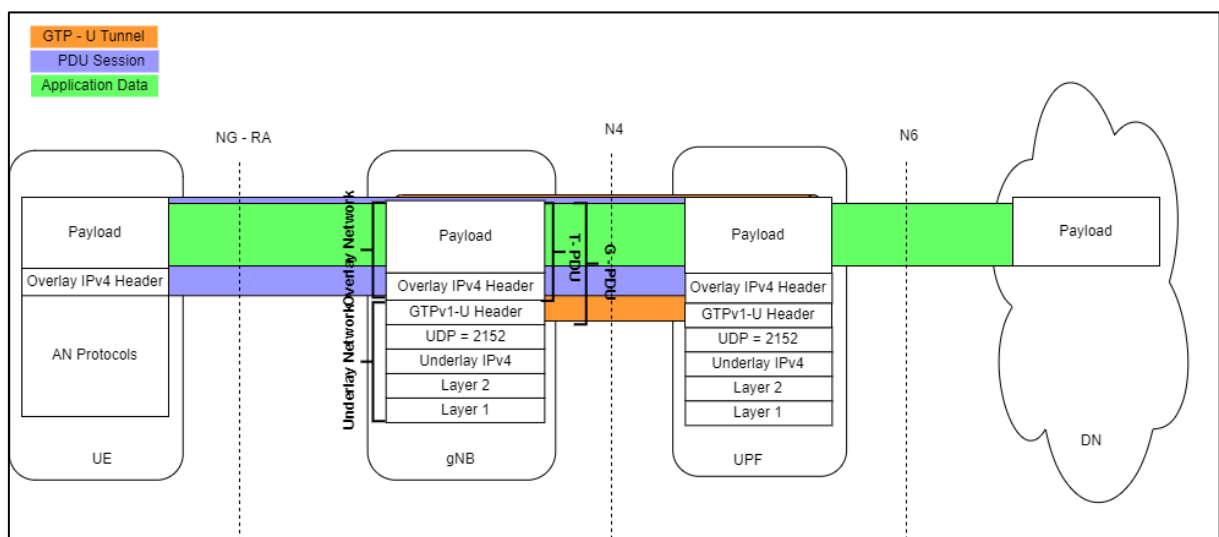


Figure 2-14 PDU Session Concept

¹⁵ The Session Management Procedure involving NFs and services are presented in Section 2.5.

A PDU Session must be established to provide data connectivity for a UE. This refers to a transport path between the UE and 5G CN to reach a particular DN. The name of DN to be connected (Data Network Name – DNN) may be provided by the UE. The PDU Session Establishment procedure can be initiated by a UE or triggered by the network. When a UE starts the procedure, it sends a PDU Session Establishment Request to the AMF via gNB. The request message includes a set of parameters related to the UE and the PDU Session to be served. The payload that includes these parameters is N1 SM Container. Only the registered users can start the procedure¹⁶. Hence, the registration procedure should have been completed before¹⁷. Following the PDU Session establishment, the User Plane path is activated. The UPF process GTP – U tunneling mechanism¹⁸ for a PDU Session.

The PDUs are carried from the UE to 5G CN. The underlying network, called the backbone network, is the radio connection from the UE to gNB and IP network (e.g., transport layer) from gNB to UPF(s). The PDUs carry the application traffic depending on the type of service, e.g., HTTP or video streaming. Overlay and Underlay networking facilitates when the UE roams among different gNBs and UPFs. The concept enables to keep the overlay traffic session between the UE and the server in the DN intact during roaming.

The UE may request the 5GS provide multiple PDU Sessions simultaneously. If accepted and processed by the 5G CN, these PDU Sessions run parallel. An example for the use case of multiple PDU Sessions could be where redundancy support is required to enhance the reliability of the service as in the Dual Connectivity technique. Alternatively, in the case of various services being consumed simultaneously by a single user. In the former cases, the PDU sessions connect the UE to the same DN for all PDU sessions; in the latter, they connect to different DNs.

The main properties of a PDU Session can be summarized as follows.

- **PDU Session Identifier:** An identifier of the PDU Session.
- **Network Slice Identifier - S-NSSAI:** An identifier of the Network Slice on which the PDU Session is to be deployed.
- **Data Network Name – DNN:** Name of the DN to which the PDU Session provides connectivity.
- **PDU Session Type:** The type of protocol of the user data carried by the PDU Session.

¹⁶ The UE could be either in idle or connected mode.

¹⁷ The registration procedure is presented in Section 2.5.1.

¹⁸ GTP – U Tunneling mechanism is introduced in Section 2.4.1.

The supported PDU Session Types in the 5G CN is as follows.

- IP based PDU Session types: IPv4, IPv6, and dual-stack IPv4v6,
- Ethernet PDU Session type,
- Unstructured PDU Session type.

The type of PDU Session is determined as a request of the UE during the Session Establishment procedure. The available PDU Session types within the CN are service operator dependent.

PDU Session Types – IP Based

An IP address must be allocated to the UE during the Session Establishment procedure for IP Based PDU Sessions. The assigned IP address(es) by the 5G CN could be only IPv4 Address, only IPv6 Address, or both depending on request. The UE must set the request depending on its IP Stack capability as well as the configuration and policies that applied to it. The selection needs to be done as follows [7].

- PDU Session Type IPv4 must be requested if the IP Stack of UE only supports IPv4
- PDU Session Type IPv6 must be requested if the IP Stack of UE only supports IPv6
- PDU Session Type IPv4v6 must be requested; the IP Stack is unknown for the UE.
- PDU Session Type IPv4v6 can be requested if the IP Stack of UE supports both IPv4 and IPv6.

It is important to stress that PDU Session Layer runs on the top of the IP Network (or backbone), which provides the IP transport between entities of the 5G CN or between (R)AN and 5G CN. Therefore, the allocated IP Address on the UE is not from the underlying IP network domain. In fact, the backbone is a private IP network, and an IP address from a DN Domain must be allocated to the UE by the SMF or the UPF. The selection of PDU Session Type and allocating an IP address to the UE are the SMF responsibilities.

The SMF selects the PDU Session Type based on

- the configuration and operator policies that applied to it,
- the type of the UE request,
- and the IP version that the DN supports

While PDU Session Type IPv6 may be utilized for the massive type of Communication due to the number of available addresses being much higher, the implementation of¹⁹ the thesis relies on PDU Session Type IPv4. Therefore, the allocation of an IP address is only considered for IPv4.

The allocation of an IPv4 address may be accomplished in two methods.

- An IP address is assigned during the PDU Session Establishment procedure.
- An IP address is obtained from DHCP by the UE after the PDU Session Establishment.

PDU Session Types – Ethernet-Based

The 5GS supports the Ethernet PDU Sessions where it is not valid for the previous generations. This type of PDU Session is intended to provide a Layer 2 Ethernet Data Network connection. It may be requested whenever the UE wants to connect a remote LAN.

Unstructured PDU Session type

The 5GS also supports the Unstructured PDU Session types without making assumptions on the format [7]. This type of PDU Session is intended to be used mainly for IoT messaging protocols such as MQTT or COAP. However, because the format is unknown, the packet processing capabilities of the UPF are highly constrained/limited, and only a single QoS (the default QoS class) can be implemented [7].

2.3.2. N4 Session Management Procedures

The SMF utilizes the N4 Session management procedures to determine the functions of the UPF. For each PDU Session, an N4 session context with an ID²⁰ is created by the SMF. The session context is sent to the UPF during the PDU Session Establishment Procedure. The UPF actions respect the rules in the context of N4 Session ID is stored both in the SMF and the UPF. The SMF can update or remove the related context in the UPF by indicating it with N4 Session ID.

¹⁹ Note: In the 5G System, there is no new technique for IPv4 address allocation; these trivial methods are utilized in 2G, 3G, and 4G/LTE EPC.

²⁰ N4 Session ID

N4 Session Establishment Procedure

To initialize the N4 session context for a PDU Session, N4 Session Establishment Procedure is performed. The steps of the procedure are as follows.

1. The SMF is triggered to start the procedure.
2. The SMF creates the session context for the PDU Session. It assigns an N4 Session ID and a set of rules to the context.
3. The SMF sends the session context to the UPF with the N4 Session Establishment Request message.
4. The UPF sends the related information to the SMF with the N4 Session Establishment Response message.

To identify the session context, both the SMF and the UPF store the N4 Session ID. The SMF also maps N4 Session ID with PDU Session ID.

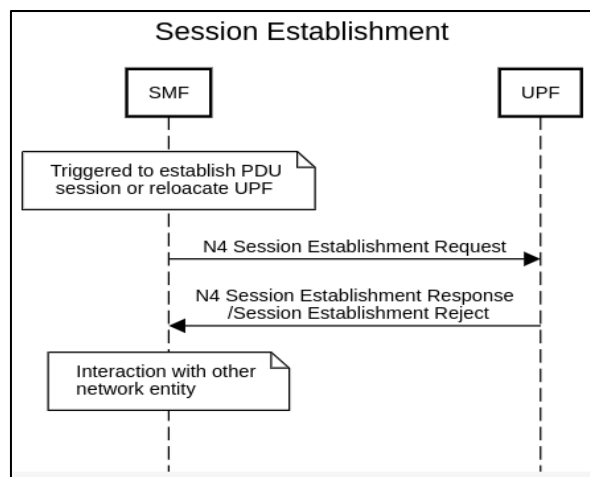


Figure 2-14 PDU Session Establishment Procedure [9]

N4 Session Modification Procedure

To modify the N4 session context, N4 Session Modification Procedure is performed. The steps of the procedure are as follows.

- The SMF is triggered to start the procedure a PDU Session.
- The SMF maps PDU Session ID with N4 Session ID.
- The SMF updates the session context indicating the N4 Session ID.
- The SMF sends the updated session context to the UPF with the N4 Session Modification Request message.
- The UPF sends the related information to the SMF with the N4 Session Modification Response message.

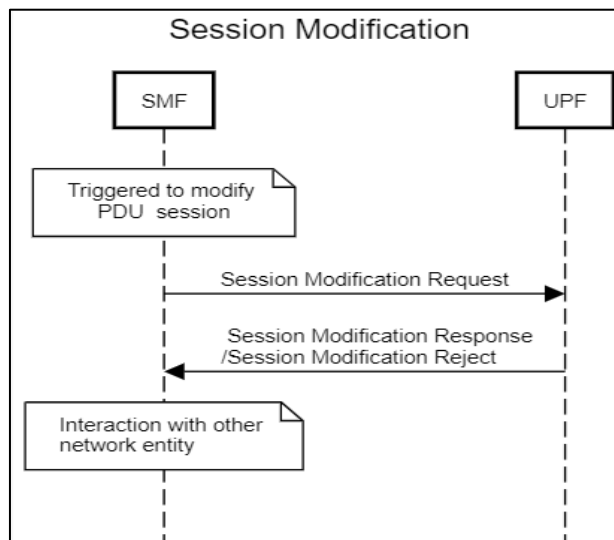


Figure 2-15 PDU Session Modification Procedure [9]

N4 Session Release Procedure

To remove the N4 session context, N4 Session Release Procedure is performed. The steps of the procedure are as follows.

- The SMF is triggered to start the procedure a PDU Session.
- The SMF maps PDU Session ID with N4 Session ID.
- The SMF requests the UPF remove the session context indicating the N4 Session ID.
- The SMF sends the updated session context to the UPF with the N4 Session Modification Request message.
- The UPF sends the related information to the SMF with the N4 Session Modification Response message.

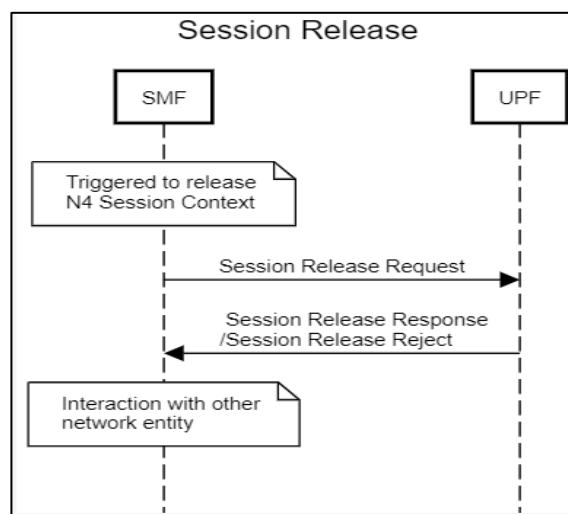


Figure 2-16 PDU Session Release Procedure [9]

N4 Session Level Reporting Procedures

The SMF can request event reports from the UPF during an N4 Session Establishment/Modification procedure. The SMF chooses which event to be reported by configuring the triggers in the UPF. These reports are used for the evaluation of the service parameters. According to them, the SMF may modify or release the ongoing PDU Session. According to [9], the reporting triggers are as follows.

- Usage report,
- Start traffic detection (Characteristics of the traffic are identified in PDR),
- Stop traffic detection (Characteristics of the traffic are identified in PDR),
- Detection of 1st downlink packet for a QoS Flow of a PDU Session with UP Connection deactivated (DL Data Report),
- Detection of PDU Session Inactivity for a specified period (UP Inactivity Report),
- The UL, DL, or round-trip packet delay measurement reporting (Session Report),
- Discard Downlink Traffic detection (DL Data Report),
- Buffered Downlink Traffic detection (DL Data Report).

2.4. User Plane Management

The User Plane is the transport path where UL and DL packets are carried end to end, in other terms, between the UE and the DN. The user plane path involves three parts; a path from the UE to the gNB, where the carriers are radio bearers, a path from the gNB to the UPF over the N3 interface where packets are encapsulated with GTP – U header, and a path from the UPF to DN over the N6 interface. It should be noted that there is no restriction on the number of UPFs that may be inserted into the path. In some cases, additional UPF(s) (e.g., I-UPF) may be inserted between the existing UPF and the DN, e.g., in the case of redundant transmission support on the N3/N9 interface.

2.4.1. GTP – GPRS Tunneling Protocol

GTP stands for GPRS Tunneling Protocol, and it includes two main components, the Control Plane part of GTP (GTP-C) and the User Plane part of GTP (GTP-U). The GTP-C carries control messages in 3G/GPRS and 4G/LTE EPC to control and manage PDN connections and the User Plan tunnels. In contrast, for the control messages in 5G CN, HTTP/2 protocol is used between the SMF and the AMF, NAS layer protocols (5G MM and 5G SM) are used between the AMF and RAN. Thus, GTP – C is no use in 5G CN. On the other hand, The GTP-U carries the user data traffic in 5G CN. It is a tunneling mechanism that runs over UDP/IP path.

G – PDU

Before delving into details, some definitions need to be provided for better understanding.

GTP-PDU stands for GTP Protocol Data Unit (PDU). It includes GTP – U Header and T – PDU.

T-PDU stands for Transport Protocol Data Unit. In simple terms, they are UL or DL data packets. The context of the T-PDUs varies depending on the service/application that the UE wants to utilize., for example, an IP datagram.

GTP – U End Points are network entities between which GTP Tunnel is set up.

Tunnel Endpoint Identifier – TEID identifies GTP – U tunnel endpoints.

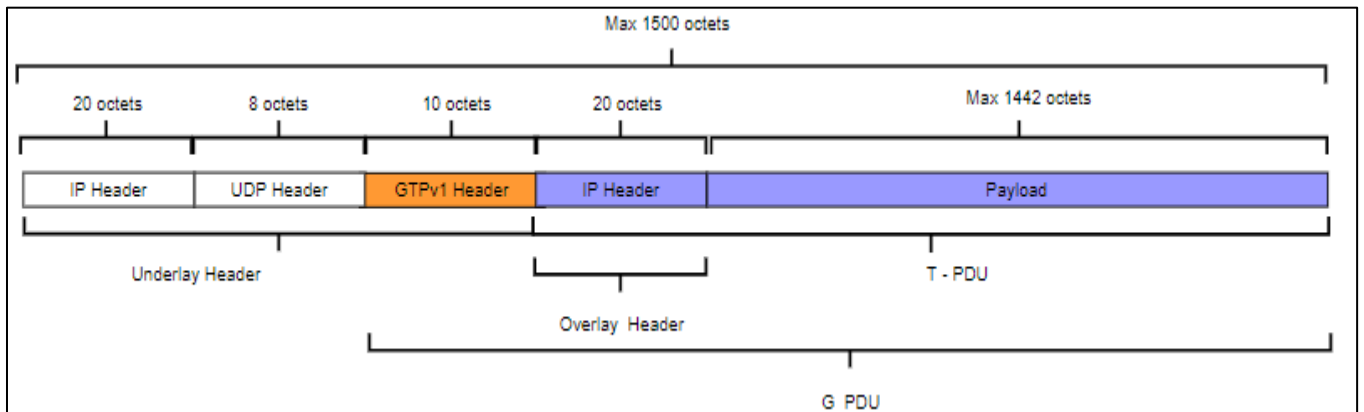


Figure 2-17 G – PDU Protocol Stack

GTP-U header

The GTP-U header involves at least 8 bytes; however, the length is variable. The details of the GTP – U header are explained for version 1. Figure 2 – 19 shows the Information Elements of the GTP – U Header.

Octets	Bits							
	8	7	6	5	4	3	2	1
1	Version		PT	(*)	E	S	PN	
2	Message Type							
3	Length (1 st Octet)							
4	Length (2 nd Octet)							
5	Tunnel Endpoint Identifier (1 st Octet)							
6	Tunnel Endpoint Identifier (2 nd Octet)							
7	Tunnel Endpoint Identifier (3 rd Octet)							
8	Tunnel Endpoint Identifier (4 th Octet)							
9	Sequence Number (1 st Octet) ^{(1) (4)}							
10	Sequence Number (2 nd Octet) ^{(1) (4)}							
11	N-PDU Number ^{(2) (4)}							
12	Next Extension Header Type ^{(3) (4)}							

Figure 2-18 GTP v1 – U Header [11]

The flags and parameters in the GTPv1 – U Header are as follows.

- The version field is dedicated to determining the GTP – U Protocol version. It is 1 for Version 1.

- PT stands for Protocol descriptor, and it is 1 for our use case.
- E stands for Extension Header Flag; when '0', the Next Extension Header does not exist. It should set 1 to carry PDU Session Containers since the context of PDU Session Containers is held in the Next Extension Header. It is worth opening parenthesis for the Next Extension Header Field Value Type of Extension Header. There could be multiple Extension Headers in a GTP – U Header. If this is the case, the PDU Session Container should be first. The Next Extension Header Field value is 1000 0101 for PDU Session Container.
- S stands for Sequence number flag. It is one 1 when the redundant transmission support on N3/N9 interface technique is applied since the packet duplication and the packet elimination rely on the GTP – U sequence number.
- TEID is the destination address of the GTP endpoint where the packets will be sent as defined before. However, all zeroes are set for the Echo Request/Response, Supported Extension Headers notification, and The Error Indication message.

GTP-U Protocol enables an endpoint to receive packets from multiple endpoints. In Dual connectivity²¹, the same TEID belonging to the same UPF could be sent to the master and secondary gNBs to route UL traffic of the same PDU session. However, the selected TEID needs to be in the multicast range. The TEID of Master gNB and the Secondary gNB are different in this use case.

Another important aspect is that the fragmentation of the GTP – U packets affect the QoS. Thus, in order to reduce the underlay layer header, Ethernet-Based or Unstructured PDU Session may be preferred for the cases where large bandwidth is required. For URLLC services, Ethernet-Based or Unstructured PDU Session could be chosen since it may reduce packet processing time in GTP endpoints. Reduction in processing time improves end-to-end delay within the 5GS service.

GTP -U Tunnels

GTP-U Tunnels provide a point-to-point link between GTP-U Endpoints like other tunneling protocols in the literature. A GTP- U endpoint is identified with three parameters [11];

- IP Address,
- UDP Port,
- Tunnel Endpoint Identifier – TEID.

²¹ Dual Connectivity is one of the redundancy techniques of URLLC services, and it is described in Section 3.

UDP/IP is the only path protocol defined to transfer GTP messages in version 1 of GTP. The UDP Destination Port number for GTP-U request messages is 2152. The UDP Source Port value could be chosen other than 2152 to keep the traffic load balance between the GTP -U endpoints.

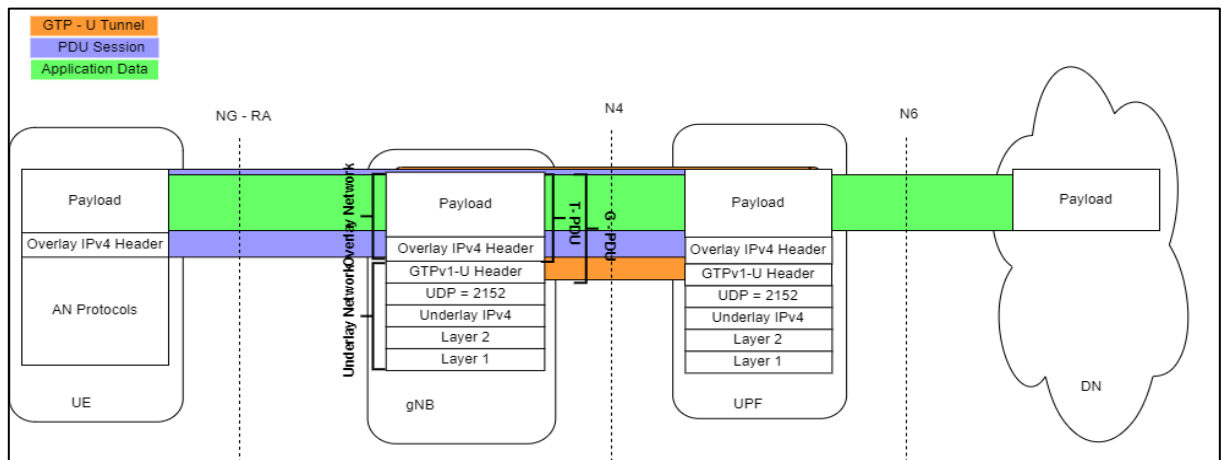


Figure 2-19 GTP -U Tunnel and PDU Session

GTP – U Procedures

T – PDUs, also referred to as inner IP packets in a GTP-U packet, are delivered in IP packets, also referred to as outer GTP-U IP packets, over GTP -U tunnels. In 5G CN, GTP-U Tunnel is established on the top IP Network between the gNB and the UPF. To set up a GTP Tunnel, a GTP endpoint should know the peer's IP Address and TEID to set up a GTP Tunnel. TEID of the UPF is provided to the SMF in the PDU Session Establishment Response message. TEID of the gNB is provided to the UPF in the PDU Session Modification message. Then, GTP – U Tunnel is established between the gNB and the UPF.

The procedure runs as following steps.

1. The SMF is triggered to establish a PDU Session and provide a Network Instance with the coordination of NSSF.
2. The SMF sends the PDU Session Establishment Request message, including the request for TEID allocation for the upcoming GTP – U Tunnel set up.
3. The UPF indicates the assigned TEID to the SMF in the PDU Session Establishment Response message.
4. The SMF gives the required information for tunnel setup to the AMF.
5. The AMF forwards the information to the gNB.
6. The gNB assigns a TEID for the upcoming GTP – U Tunnel and sends it to the AMF.
7. The AMF forwards this information to the SMF.
8. The SMF updates the UPF with the PDU Session Modification Request message.

9. GTP – U Tunnel is established between the gNB and the UPF.

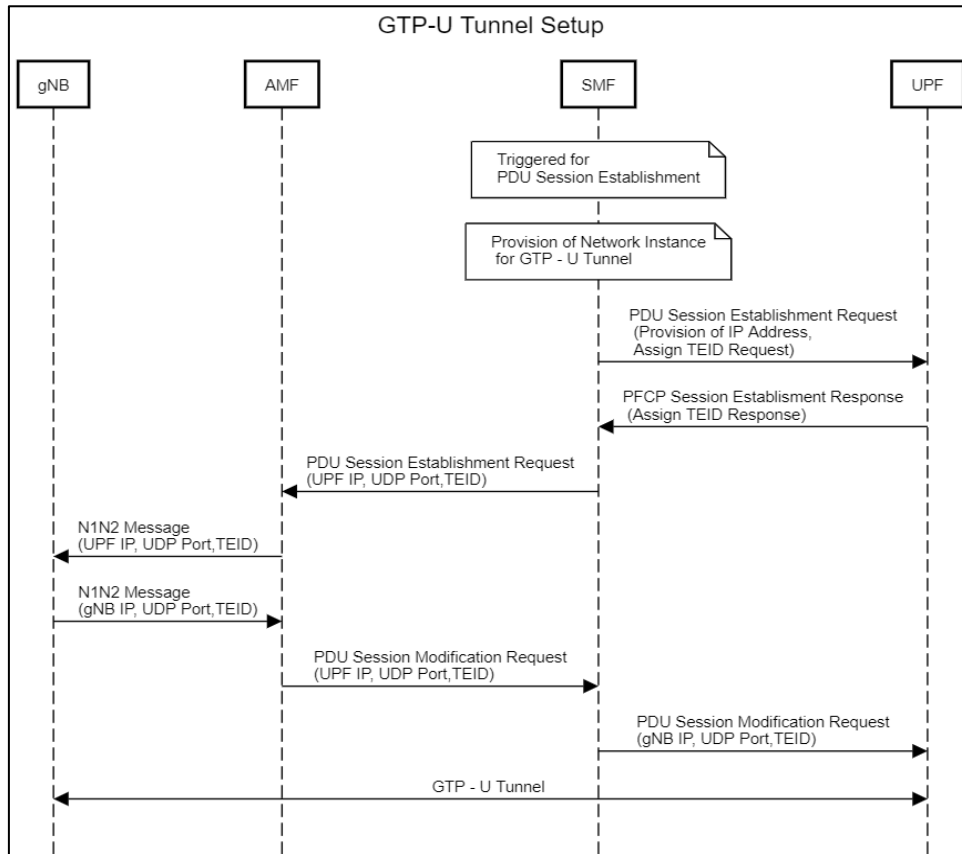


Figure 2-20 GTP – U Tunnel Setup Procedure

Once the GTP – U Tunnel is set up, the peers exchange some control messages for tunnel management. There is no rule about which GTP endpoints start the procedure. GTP – U Message Types are as follows.

- Echo-Request
- Echo Response
- Error Indication
- Supported Extension Headers Notification
- Tunnel Status
- End Marker

Other than Echo Response messages, the source port of the GTP – U control messages could be set dynamically. Echo Response messages need to use the UDP Source Port value of the Echo Request messages as the UDP Destination Port value.

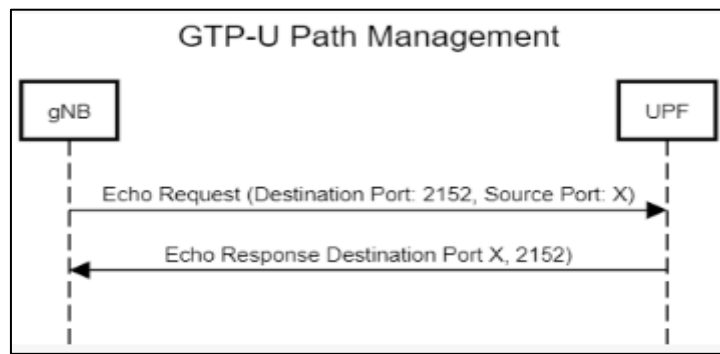


Figure 2-21 GTP – U Echo Request / Response Messages

2.4.2. Packet Forwarding Model

The context (PCF Session context) within the PCF Session Related messages contains some parameters that indicate which packets the action applies and a set of rules that tell the action to be taken for matched packets. In the 5G CN, a PCF Session context could be standalone (not tied to any PDU Session) or tied to a specific PDU Session. The rules in the PCF Session context are included as Information Elements, and they are categorized as follows.

- PDR: Packet Detection Rule,
- FAR: Forwarding Action Rule,
- URR: Usage Reporting Rule,
- BAR: Buffering Action Rule,
- MAR: Multi-Access Rule,
- QER: QoS Enforcement Rule.

While the further investigation of rules can be found in the following sections, the SMF handles the packet processing by adding, deleting, or modifying the PCF Session context sent to the UPF. One or more PDRs are constructed and delivered by the SMF for each PDU session. Moreover, each PDR may contain multiple FARs, QERs, URR, and/or BARs. The association between the rules and the packet processing flow in the UPF is depicted in Figure 2 -23.

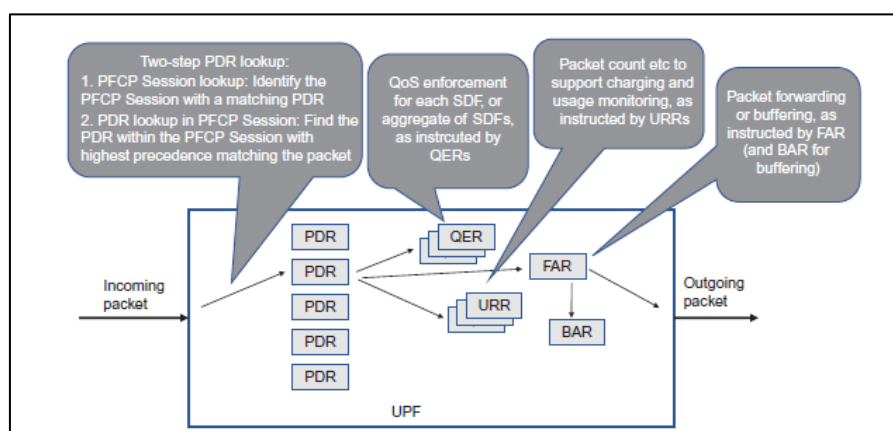


Figure 2-22 Packet Processing in the UPF [7]

PDR and PDR Handling

Packet Detection Rules: The set of rules contains information for detecting the incoming data packets and specific processing rules that apply to detected packets. Distinct PDRs are utilized for uplink and downlink.

Packet Detection Information – PDI: It indicates the key filtering elements that match with a specific packet, e.g., IP address, port information.

A PCF Session Context may include more than one PDRs. Each PDR should contain one PDI and one or more FARs, but it may not contain URR, BAR, MAR, or QER. Multiple PDRs may refer to the same FAR(s) (or URR(s), BAR(s), MAR(s) or QER(s)).

If an incoming packet does not match any PDRs inserted in the UPF, then the UPF drops packets. If there is a match, then the packet is processed respecting the instructions in FAR(s) (or URR(s), BAR(s), MAR(s), or QER(s)).

The PDR handling is processed according to the information provided in the PDI. The PDI should provide the following information to the UPF for packet processing.

- The source interface of the incoming packets. The source interface value is
 - Access for UL packets,
 - Core for DL packets,
 - SMF for the packets coming to the SMF
- Local F-TEID,
- Instance,
- UE IP address(es),
- Optionally, one or more QFI(s), Ethernet Packet Filter(s), and/or Ethernet PDU Session Information can be provided in the 5G System.

FAR

Forwarding Action Rules – FAR: The set of rules contains information for the incoming data packets' actions. Multiple PDRs may point to the same Far. The UPF selects the FAR according to the FAR ID in the PDR. For each PDR, there could be multiple FAR.

PDR		FAR	
<i>PDR ID</i>		<i>FAR ID</i>	
<i>Precedence</i>		<i>Action</i>	
<i>Packet detection Information (information Against which Incoming packets are matched)</i>	<i>Source interface</i>	<i>Forwarding Parameters</i>	<i>Network instance</i>
	<i>UE IP address</i>		<i>Destination Interface</i>
	<i>Network instance</i>		<i>Outer header Creation</i>
	<i>CN tunnel info</i>		<i>Send end marker packet(s)</i>
	<i>Packet filter set</i>		<i>Transport level Marking</i>
	<i>Application ID</i>		<i>Forwarding policy</i>
	<i>QoS flow ID</i>		<i>Request for Proxying in UPF</i>
	<i>Ethernet PDU session Information</i>		<i>Container for header enrichment</i>
<i>Outer header removal</i>	<i>Outer header Creation</i>	<i>Duplicating parameters</i>	<i>Destination Interface</i>
<i>FAR ID</i>	<i>Transport level Marking</i>		<i>Event-based reporting</i>
			<i>Linked URR ID(s)</i>

Table 2-3 PDR and FAR Parameters

Apply Action

Action: It indicates the action to be taken for the incoming packets. The Action could be;

- Forward, for forwarding incoming packets,
- Duplicate, for duplication of the incoming packets in the redundant transmission,
- Buffer, for buffering the incoming packets (e.g., during mobility),
- Drop, for dropping the packets (e.g., for security reasons).

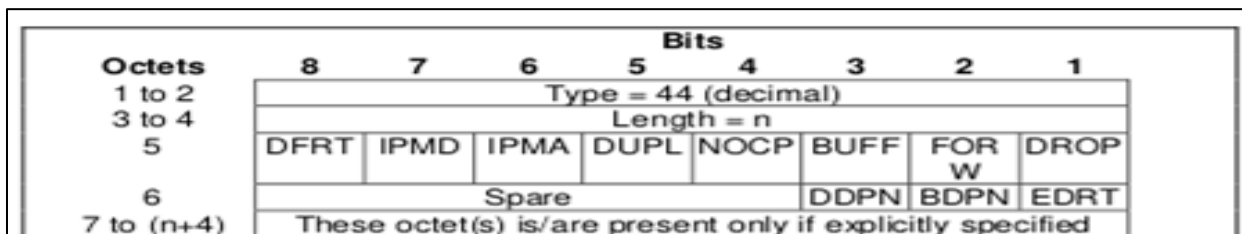


Table 2-4 Outline of Apply Action [8]

Forwarding Parameter - Destination Interface indicates where incoming packets will be forwarded. Destination Interface value could be one of these.

- Access for UL packets,
- Core for DL packets,
- SMF for the packets coming to the SMF

On Octet 5, Bit 8 – DFRT (Duplicate for Redundant Transmission) should be set to 1 when the SMF requests packet duplication due to Dual Connection²².

On Octet 6, Bit 1 – EDRT (Eliminate Duplicate Packets for Redundant Transmission) should be set to 1 when the SMF requests packet elimination due to redundant transmission.

Packet duplication and packet elimination are processed according to instructions within the Red. Trans. Forwarding Parameters IE and the Red. Trans. Detection Parameters IE.

Octet 1 and 2		Redundant Transmission Forwarding Parameters IE Type = 270 (decimal)					
Octets 3 and 4		Length = n					
Information elements	P	Condition / Comment	Appl.				IE Type
			Sx a	Sx b	Sx c	N4	
Outer Header Creation	M	This IE shall be present if the UP function is required to perform the redundant transmission of the outgoing packet. If present, it shall contain the F-TEID of the remote GTP-U peer for redundant transmission.	-	-	-	X	Outer Header Creation
Network Instance for Redundant Transmission	C	This IE shall be included if the GTP-U tunnel used for redundant transmission uses a different network Instance than the Network Instance used for the primary GTP-U tunnel.	-	-	-	X	Network Instance

Table 2-5 Redundant Transmission Forwarding Parameters IE in FAR [8]

Octet 1 and 2		Redundant Transmission Detection Parameters IE Type = 255 (decimal)					
Octets 3 and 4		Length = n					
Information elements	P	Condition / Comment	Appl.				IE Type
			Sx a	Sx b	Sx c	N4	
Local F-TEID for Redundant Transmission	M	This IE shall identify the local F-TEID to match for an incoming packet for redundant transmission. The CP function shall set the CHOOSE (CH) bit to 1 if it requests the UP function to assign a local F-TEID to the PDR.	-	-	-	X	F-TEID
Network Instance for Redundant Transmission	C	This IE shall be included if the Local F-TEID for Redundant Transmission uses a different network Instance than the Network Instance used for the Local F-TEID for the primary GTP-U tunnel.	-	-	-	X	Network Instance

Table 2-6 Redundant Transmission Detection Parameters IE in FAR [8]

Usage Reporting Rule (URR)

The set of rules configures the UPF for how the measurement is processed. The UPF should report the measurement to the SMF. The measurement could be based on packets, bytes, or the user.

Buffering Action Rule (BAR)

The set of rules contains information for how buffering of packets shall be done, e.g., in case a handover

²² This is only used in 4G/LTE EPC System.

QoS Enforcement Rule (QER)

The set of rules contains information on managing QoS for the packets.

2.5. System Procedures

2.5.1. Registration Procedure

The Registration procedure is processed to connect a UE to the 5G System; therefore, the UE is authorized and enabled to utilize the 5GS services. Although it is first the requirement for a UE to use the 5GS services, the Registration procedure may be processed to use other functionalities, which is stated by the type of the Registration procedure itself. The type of registration could be one of these.

- Initial Registration to the 5G System,
- Mobility Registration,
- Periodic Registration Update,
- Emergency Registration.

Moreover, a UE or the 5G CN can trigger the Registration procedure. This section covers a simplified version of a UE-triggered registration procedure since it must be processed in the implementation. The description of the steps of the procedure is as follows.

1. NAS Registration Request Message is transmitted from the UE to the (R)AN. the UE shall include its SUCI in the Registration Request
The (R)AN selects the AMF that the message will be forwarded. The selection could be applied based on Requested N-SSAI or according to the default AMF in the configuration. After the selection process is completed, the (R)AN forwards the message to the AMF.
2. The AMF manages the authorization process in coordination with AUSF and UDM.
The AMF processed the selection of the AUSF based on SUPI or SUCI.
The UE authentication is initiated by invoking an AUSF as a request of the AMF.
3. The AMF processed the selection of the UDM based on SUPI.
UDM may select a UDR instance.
The AMF registered itself as a serving AMF for this UE.
The AMF gets subscription data and processes the subscription of the UE in the UDM.
4. The UE request to connect the 5GS for the first time; therefore, the SMF Selection and PDU Session Establishment procedure are initialized.
5. After the registration procedure is completed successfully, the AMF creates and then sends the NAS Registration Accept message to the (R)AN.
The (R)AN forwards the message to the UE.

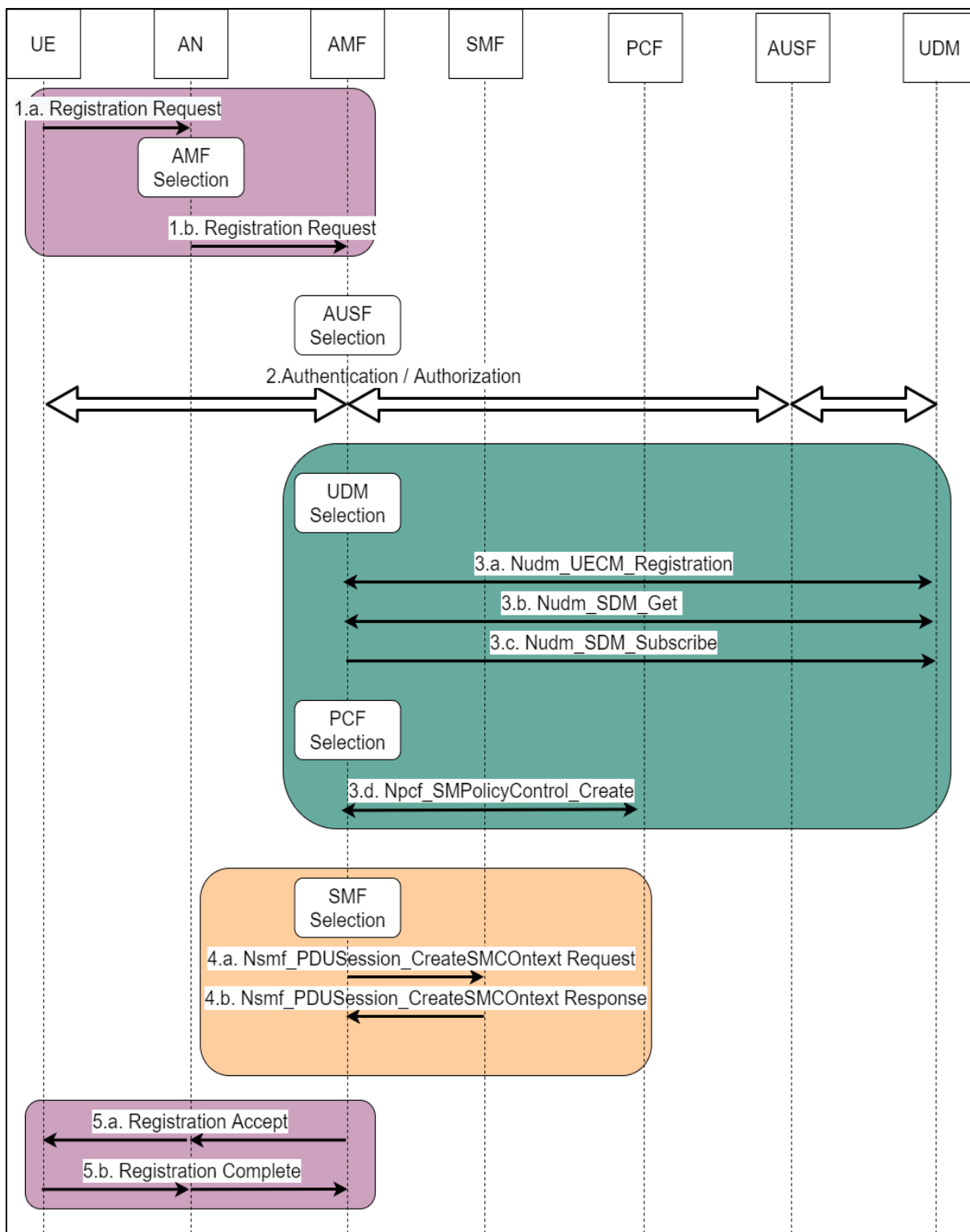


Figure 2-23 The call flow of a UE-triggered registration procedure

2.5.2. Service Request Procedure

The Service Request procedure is processed to enable a UE to utilize a specific service(s) of the 5G System. It can be triggered by the UE or the 5G CN; in either case, the request is sent to

the AMF. The UE can start the procedure in the CM-IDLE state or CM - CONNECTED state. Depending on the state of the UE, the type of requested service varies. The Service Request procedure is initiated

- To send user data when the UE is in CM-IDLE state,
- To activate User Plane connection for PDU Sessions when the UE is in CM-CONNECTED state,

This section covers a simplified version of a UE triggered Service Request procedure since it must be processed in the implementation. Once the UE requests a service, it should not ask another one before completing the procedure. The Service Request procedure may be initiated to utilize other services (e.g., to request emergency services fallback); however, they are out of the scope of the thesis.

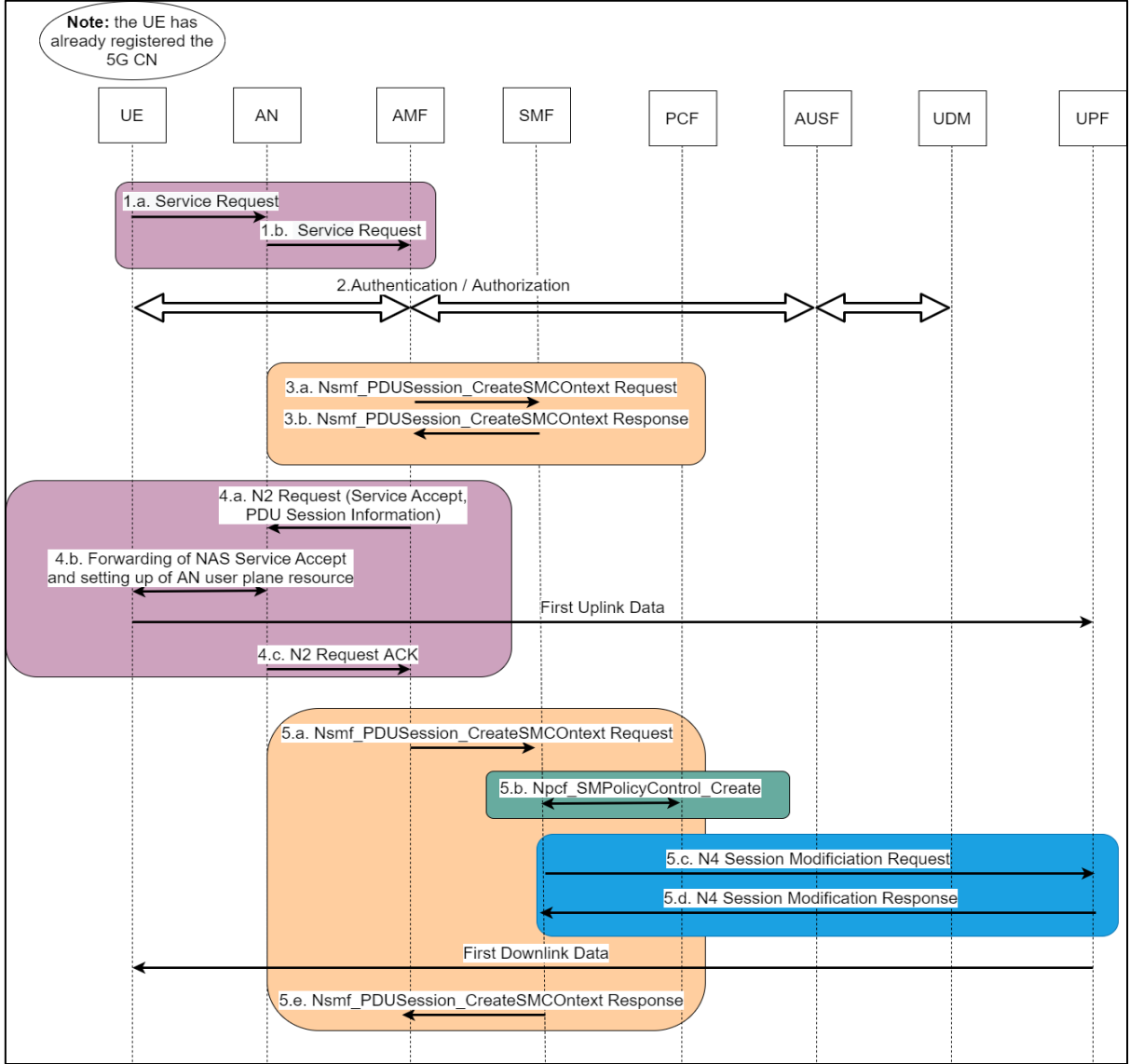


Figure 2-24 The call flow of a UE-triggered Service Request procedure

The description of the steps of the procedure is as follows.

1. NAS Service Request message is transmitted from the UE to the (R)AN. The UE must provide PDU ID(s) to identify PDU Sessions that are activated or released. (R)AN forwards the message to the AMF.
2. The AMF may process re-authorization for the UE in coordination with AUSF and UDM.
3. The AMF processed the selection of the serving SMF(s) to be notified based on PDU ID(s). The SMF provides all the UPF(s) and the PDU Session(s) information. After the Service Request procedure is completed successfully, the AMF creates NAS Service Accept message, including all the information about the UPF(s) and the PDU Session(s), and then send it to the (R)AN. The (R)AN forwards the message to the UE. After this step, the UE has all the information for the UL path. The UE starts to send UL data towards the UPF.
4. The AMF informs the SMF(s) of the result of the User Plane establishment and the (R)AN tunnel endpoint identifier(s).
5. The SMF updates the location of the UE on the PCF. The SMF provides the UPF(s) with all the PDU Session(s) information to be modified. After this step, the UPF(s) has/have all the information for the DL path. The UPF starts to send DL data towards the UE.

2.5.3. Session Management Procedures

PDU Session Establishment

The PDU Session Establishment procedure may be initiated by a UE or the 5G CN. Although the procedure can also be triggered due to handover or emergency services, this section only covers establishing a new PDU Session for Non – roaming and a UE-triggered case. The description of the steps of the procedure and the use cases for the implementation is as follows.

1. NAS PDU Session Establishment Request Message is transmitted from the UE to the (R)AN.

The UE provides the following parameters [7];

- Request Type (Initial request for the implementation),
- PDU session ID (generated by the UE),
- Requested PDU Session Type (IPv4 for the implementation),
- Requested SSC mode (1 for non–roaming case),
- 5GS M Capability (not a part of implementation),
- PCO (Used to request reliable data service, not a part of implementation),

- SM PDU DN Request Container, [Number of Packet Filters] (for the QoS, not a part of implementation),
- Header Compression Configuration,
- UE Integrity Protection Maximum Data Rate (not a part of implementation),
- Always-on PDU Session Requested (not a part of implementation).

2. The AMF processes the selection of the SMF (pre-configured for the implementation).

3. The AMF consumes the Nsmf_PDUSession_CreateSMContext Request service with the parameters it gets from the NAS message

- SUPI,
- DNN (chosen either by the UE or the AMF among the allowed NSSAI)²³,
- S-NSSAI(s) (selected by either by the UE or the AMF among the allowed DNN)²³,
- PDU Session ID,
- AMF ID (AMF Instance ID),
- PCF ID.

4. The SMF consumes Nudm_SDM_Get service by utilizing the parameters the SUPI, DNN, and S-NSSAI,

5. The SMF provides Nsmf_PDUSession_CreateSMContext Response service.

6. The SMF may process re-authorization for the UE in coordination with AUSF and UDM (not a part of implementation),

7a. The SMF processes PCF selection (pre-configured for the implementation),

7b. The SMF may process SM Policy Association Establishment (not a part of implementation),

8. The SMF processes the selection of the UPF (pre-configured for the implementation),

9. The SMF may process SM Policy Association Modification (not a part of implementation),

10.a. The selected UPF receives the Session Establishment Request from the SMF. The message includes a set of rules for packet forwarding, packet detection, buffering, QoS parameters, and these rules/parameters can be modified after the session establishment.

The Redundant Transmission Support can be applied during or after the Session Establishment procedure and requested by the UE or decided by the SMF. If the UE requests the redundant transmission, the SMF informs the UPF to provide two CN Tunnel Info, indicating one tunnel is redundant. Moreover, packet duplication and packet elimination are performed by the UPF with the indication from the SMF.

10.b. The UPF sends to the SMF a response, including the CN Tunnel Info(s) depending on the request. CN Tunnel Info includes TEID at the 5G CN side. An IP address can also be provided if the SMF requests it.

²³ Pre-configured for the implementation

11. The SMF provides all information to the AMF required for data transfer.
12. The AMF informs the (R)AN that the PDU Session Establishment Request is accepted. The corresponding PDU Session is identified with PDU ID inside the message.
13. The (R)AN informs the UE that the PDU Session Establishment Request is accepted. This message includes all the related information for the PDU Session and the data connectivity.

The allocation of AN Tunnel Info is performed in this step by (R)AN itself. AN Tunnel Info includes TEID at the 5G AN side. If the redundant transmission is applied, (R)AN allocates two AN Tunnel Info for the PDU Session. Moreover, If Dual Connectivity is deployed, which also includes two tunnels as the redundant transmission, the Master (R)AN node may assign some (zero or more) QFIs. AN Tunnel Information consists of the TEID and assigned QFIs. AN Tunnel Information is set up to the Master (R)AN and the Secondary (R)AN.

In some use cases of the redundant transmission support, I – UPFs can be inserted between the PSA UPF and NG – RAN. UPFs assign their CN Tunnel Info and send this information to the SMF by indicating which CN Tunnel Info is for the redundant path.

14. The (R)AN sends AN Tunnel Info(s) and QFIs to the AMF.
 15. The AMF forwards the AN Tunnel Info(s) and QFIs to the SMF.
 - 16a. The SMF updates the UPF with the AN Tunnel Info and corresponding forwarding rules. SMF uses QFIs to determine QoS for each user profile and informs the UPF.
 - 16b. The UPF provides an N4 Session Modification Response to the SMF.
- If multiple UPFs are used in the PDU Session, the UPF in step 16 refers to the UPF terminating the N3. After this step, the UPF delivers any DL packet to the UE.

The redundant transmission can be applied as a decision of the SMF even if the UE does not request it. If this is the case, the UPF sends reports to the SMF respecting the measurement defined in QER (e.g., end-to-end latency). According to these reports, the SMF may decide to perform redundant transmission for one or more QoS Flows of the PDU; the SMF also indicates the UPF and the gNB²⁴ to perform packet duplication for the QoS Flow(s) in downlink direction by forwarding rules.

17. The SMF sends a PDU Session Update Response message to the AMF to complete the update operation.

²⁴ Via the AMF.

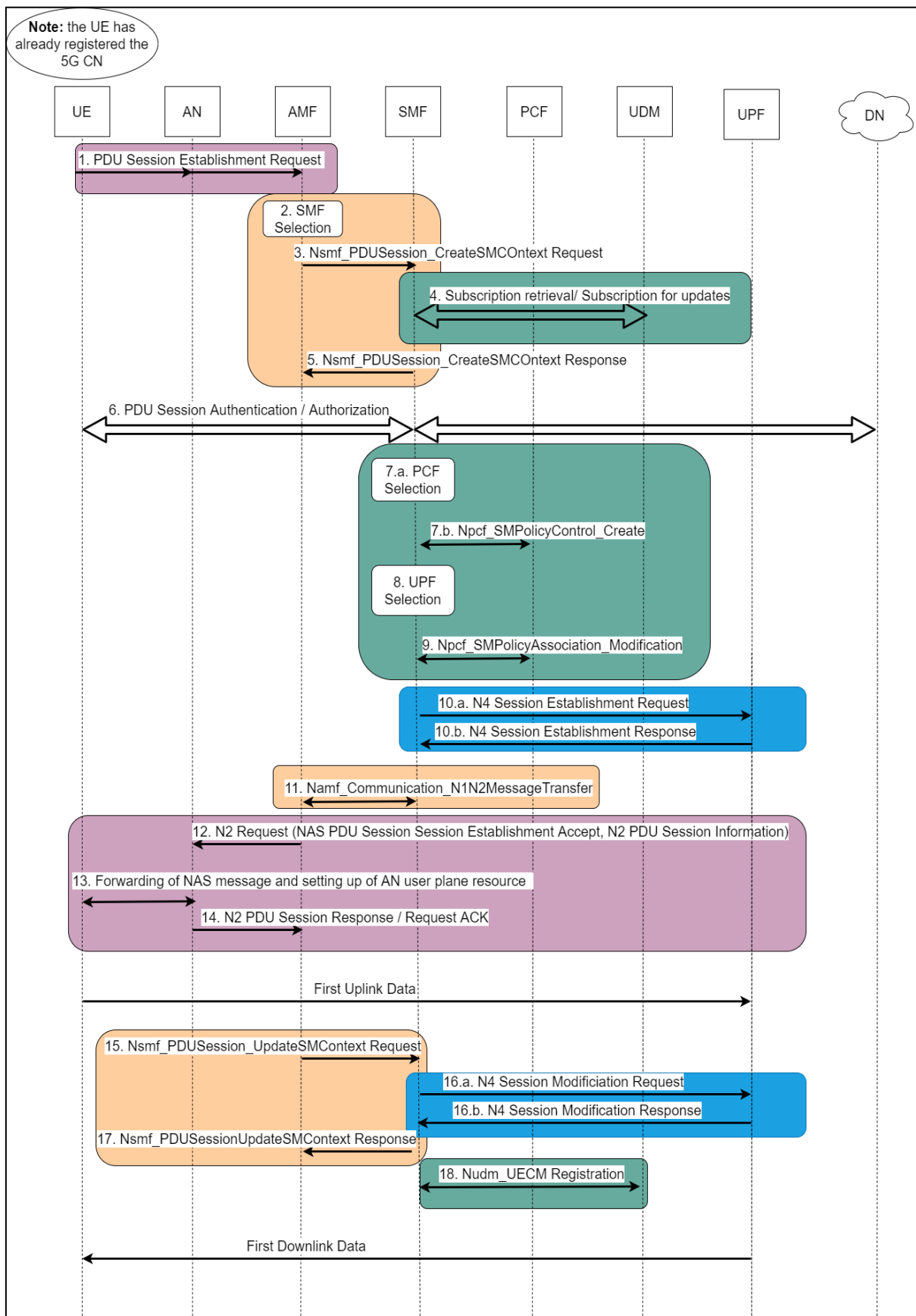


Figure 2-25 The call flow of a UE-triggered PDU Session Establishment Procedure

3 Ultra-Reliable Low Latency Communication – URLLC

One of the fundamental service categories of the 5GS is URLLC services. The service category enables the integration of new applications and use cases to 5GS. These are mission-critical applications with stringent reliability and latency characteristics. To satisfy reliability requirements, Dual Connectivity in 4G/LTE EPC System is introduced in 3GPP Release 14, where the latency requirement is set as 4 msec. With 3GPP Release 15, the reliability and the latency requirements became more challenging.

The system latency includes the delay on the radio interface, the delay in the CN and the path delay between the 5CN and the DN server, and data processing delay in each node along the user plane path. Some of these system delay contributors could be decreased within the 5GS (e.g., the location of the selected UPF affects the system latency). On the other hand, reliability is measured by the packet loss on the user plane path. The improvement techniques are based on packet duplication [12], which requires redundancy. The redundancy increases within packet processing time the mobile systems unavoidably. This explains that the intrinsic trade-off exists between reliability and latency. As mentioned, the Dual Connectivity is the first proposal for redundancy techniques, and the techniques require improvements on the AN where the redundancy is obtained with radio bearers. In addition, they involve the coordination between the Master gNB and the Secondary gNB over the Xn interface to handle the packet duplication on the radio interface. Packet duplication/elimination processing time and the path delay on the Xn interface degrade the overall system's latency performance. On the other hand, in 3GPP Release 16, new redundancy techniques apply to the 5G CN.

Scenario	Max. allowed end-to-end latency (note 2)	Survival time	Communication service availability (note 3)	Reliability (note 3)	User experienced data rate	Payload size (note 4)	Traffic density (note 5)	Connection density (note 6)	Service area dimension (note 7)
Discrete automation	10 ms	0 ms	99,99%	99,99%	10 Mbps	Small to big	1 Tbps/km ²	100 000/km ²	1000 x 1000 x 30 m
Process automation – remote control	60 ms	100 ms	99,9999%	99,999%	1 Mbps up to 100 Mbps	Small to big	100 Gbps/km ²	1 000/km ²	300 x 300 x 50 m
Process automation – monitoring	60 ms	100 ms	99,9%	99,9%	1 Mbps	Small	10 Gbps/km ²	10 000/km ²	300 x 300 x 50
Electricity distribution – medium voltage	40 ms	25 ms	99,9%	99,9%	10 Mbps	Small to big	10 Gbps/km ²	1 000/km ²	100 km along power line
Electricity distribution – high voltage (note 2)	5 ms	10 ms	99,9999%	99,999%	10 Mbps	Small	100 Gbps/km ²	1 000/km ² (note 8)	200 km along power line
Intelligent transport systems – infrastructure backhaul	30 ms	100 ms	99,9999%	99,999%	10 Mbps	Small to big	10 Gbps/km ²	1 000/km ²	2 km along a road

NOTE 1: Currently realised via wired communication lines.
NOTE 2: This is the maximum end-to-end latency allowed for the 5G system to deliver the service in the case the end-to-end latency is completely allocated to the 5G system from the UE to the Interface to Data Network.
NOTE 3: Communication service availability relates to the service interfaces, reliability relates to a given system entity. One or more retransmissions of network layer packets may take place in order to satisfy the reliability requirement.
NOTE 4: Small: payload typically ≤ 256 bytes
NOTE 5: Based on the assumption that all connected applications within the service volume require the user experienced data rate.
NOTE 6: Under the assumption of 100% 5G penetration.
NOTE 7: Estimates of maximum dimensions; the last figure is the vertical dimension.
NOTE 8: In dense urban areas.
NOTE 9: All the values in this table are targeted values and not strict requirements. Deployment configurations should be taken into account when considering service offerings that meet the targets.

Table 3-1 Performance requirements for URLLC Services [13]

3.1. Support for Ultra-Reliable Low Latency Communication

The techniques to support ultra-reliable and low latency communication at the 5G CN are defined in 3GPP TS 123.501. In general, these techniques rely on redundant transmission and duplication of the data flows in order to increase the overall reliability of the service. These techniques are as follows.

- Dual Connectivity (DC) based on Redundant UP Path,
- Redundant transmission on N3/N9 interfaces,
- Redundant transmission at the transport layer.

The 5GS service that provides one of the services above Established PDU Sessions should be in always-on mode. Packet duplication and packet elimination should be performed in the core network. The SMF can decide on redundant transmission support. The SMF may choose to set -up a redundant path analyzing the QoS flows within the PDU Session. Moreover, the redundant path can be deployed as a request from the UE before the PDU Session Establishment procedure. A network entity (NF or gNB) identifies the packets with their sequence numbers. The duplicated packets have the same sequence number, and the elimination is processed based on the sequence numbers of the packets. Whenever a network entity performs elimination, it forwards²⁵ one of the packets and discards the other.

3.2. Dual Connectivity based end to end Redundant User Plane Paths

As the name indicates, the dual connectivity technique states that a UE establishes two connections towards NG – RAN(s), therefore two PDU Sessions over the 5G network. The dual connection can be established between a UE and two NG – RANs, Master NG - RAN and Secondary NG – RAN, or between a UE and an NG -RAN over two paths. In both cases, the UE has only one interface toward the AMF, and two distinct radio bearers provide redundancy. To deploy such a redundant path for the user plane, the UE, the (R)AN, and the UPF should support the dual connectivity.

Even the UE triggers the dual connectivity procedure; the SMF is in charge of accepting the request or not. The decision operation is coordinated with related NF (e.g., The SMF checks the user policy with the PCF). If the request is accepted, two disjoint end-to-end paths, so two PDU Sessions are established for the UE.

Dual connectivity can be applied on both IP-based and Ethernet-based PDU Sessions. The UPFs on the redundant paths are connected to the same DN by using possibly different

²⁵ If there is not more actions to be applied that data flow.

routes [7]. The UE Route Selection Policy (URSP) or local configuration on the UE with PDCP should be implemented to manage redundancy.

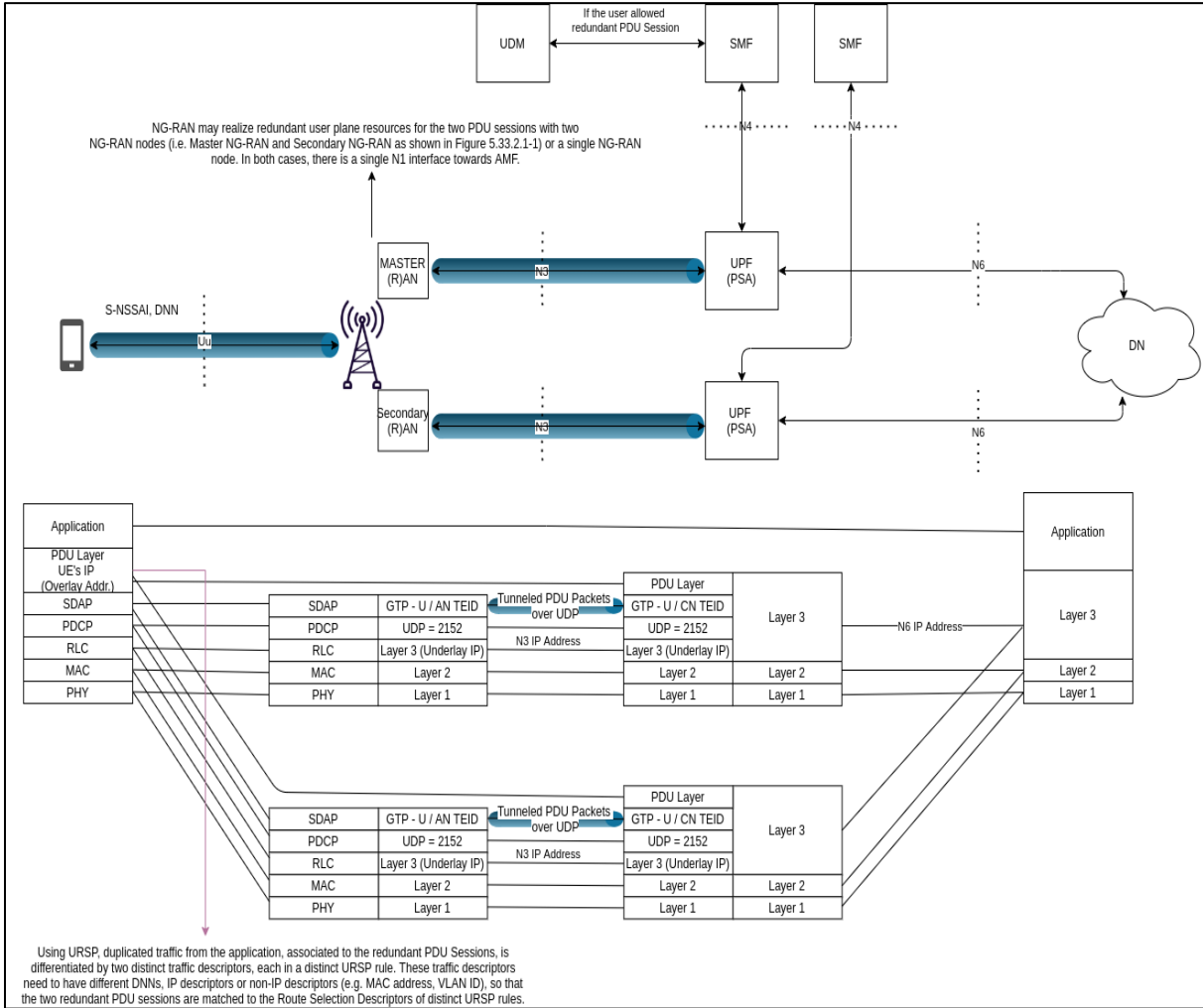


Figure 3-1 5G CN deployment for Dual Connectivity including Protocol Stack

The steps for dual connectivity can be summarized as below.

1. The UE requests the establishment of redundant PDU Sessions, providing one DNN and one S-NSSAI for each PDU session. DNN is the same for PDU Sessions, whereas S-NSSAIs are different.
2. The AMF performs the SMF selection procedure.
3. The AMF forwards the UE request to the selected SMF.
4. The SMF consumes the UDM subscription data get service.
5. The SMF consumes the PCF user policy get service.
6. The SMF accepts or refuses the request by considering S-NSSAI, DNN, User Subscription Data, and User Policies.

If The SMF accepts the request, it determines the RSN (Redundancy Sequence Number) for the PDU Sessions by taking these parameters as inputs.

7. The SMF provides the RSN to the NG – RAN. The RSN value indicates the requirements of the user plane for the redundant PDU Sessions.
8. The NG – RAN decides on establishing redundant paths based on the RSN parameter, its resources, and its local configuration. The Local configuration on the NG – RAN should support the appropriate UPF selection for disjoint paths.
9. If the NG – RAN rejects to set up the redundant path, the established PDU Session should be released.
10. If the NG – RAN accepts dual connectivity, the connection should be provided based on the RSN and RAN configuration.

3.3. Support of redundant transmission on N3/N9 interfaces

Redundant transmission support on N3/N9 interfaces technique refers to setting up two tunnels on only the N3 interface or both N3 and N9 interfaces. The redundant path can be set up during or after a URLLC QoS Flow establishment. It is the SMF that determines whether the Communication requires a redundant path or not. This technique provides redundancy with GTP – U Tunnels in a single PDU session. The SMF should have such functionality to do that. The AMF process the selection procedure of the convenient SMF based on S-NSSAI and DNN. The SMF decides to provide redundant transmission support on N3/N9 interfaces if it ensures that the following conditions satisfy the requirements of the URLLC services [5];

- The reliability of the NG – RAN node is high enough,
- The reliability of UPF is high enough,
- The reliability of CN NFs are high enough,
- The reliability of a single N3 Tunnel is not high enough.

When the SMF decides to provide redundant transmission support, it indicates NG – RAN and PSA UPF since the redundant tunnel is deployed between them on the top of the same underlying network in the 5G CN. In this way, the differentiation is implemented on the transport layer path, which is multiple GTP – U tunnels in the 5GS. Two disjoint tunnels are associated with a single PDU Session where different QoS Flows may be applied.

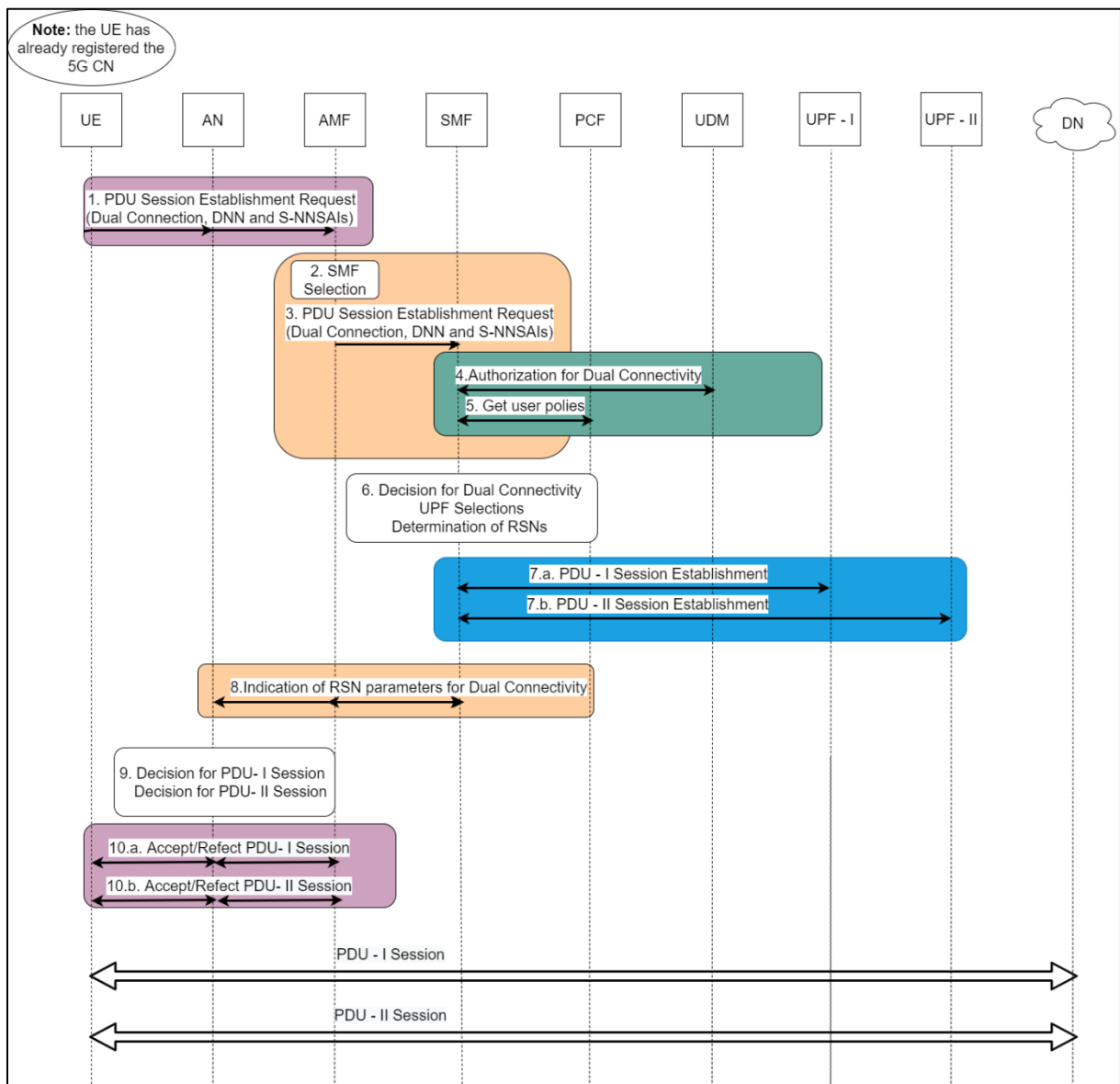


Figure 3-2 System Procedure for Dual Connectivity

To route the duplicated data flows via the redundant tunnel, different routing information should be provided to the NG – RAN as well as the parameters related to redundant GTP – U tunnel such as TEID of CN. The routing information can be provided by the SMF or the UPF, depending on the configuration during the PDU Session Establishment. TEID is determined by the UPF and shared with the SMF in the Session Modification Response message. The capability of redundant transmission support can be configured per network slice or per SMF service area in the NG – RAN.

The operation runs as follows.

1. NG – RAN duplicates the packets of UL Data Flows, and it assigns the same GTP-U sequence number to them.
2. NG – RAN forwards the duplicated packets towards the UPF - PSA throughout the redundant tunnel.
3. The UPF eliminates one of the packets with the same GTP-U sequence number and forwards the other towards DN.
4. For the DL, the operation is processed in the reverse direction.

The flow of duplicated packets may not arrive in the same order as the original ones. Even if this is the case, reordering is not required for the redundant transmission.

If the UE connects to a new NG – RAN (e.g., due to mobility) while there is an ongoing redundant transmission on the previously connected NG – RAN, and the new NG – RAN does not support the redundant transmission, then the SMF may decide to release the QoS flow which is subject to redundant transmission.

3.3.1. N3/N9 GTP – U Tunnel Setup

Since the redundant transmission support on N3/N9 interfaces technique provides redundancy with GTP – U Tunnels on the same underlying network, the operation of the GTP – U Tunnel setup requires further investigation. This section analyzes the use case of GTP – U tunnels for the URLLC.

1. The SMF provides UL PDR FAR to UPF and asks it to assign TEID for GTP – U Tunnel.
2. The UPF sends the UL TEID info in the Session Establishment Request message.
3. The SMF provides the UL TEID info to the AN and receives the DL TEID information in the N1N2 Response message.
4. Once the GTP – U Tunnel is set up, the AN provides QoS Flow(s) for the UL and DL traffic.
5. The SMF decides to redundant transmission support and initiates to GTP – U Tunnel establishment procedure for the second tunnel following the same steps as the first one.

The UE could request the redundant path before the PDU Session Establishment procedure. In this case, the SMF starts the procedure to establish two GTP – U tunnels in advance²⁶.

²⁶ The SMF requests of two TEIDs in the PDU Session Establishment Request message.

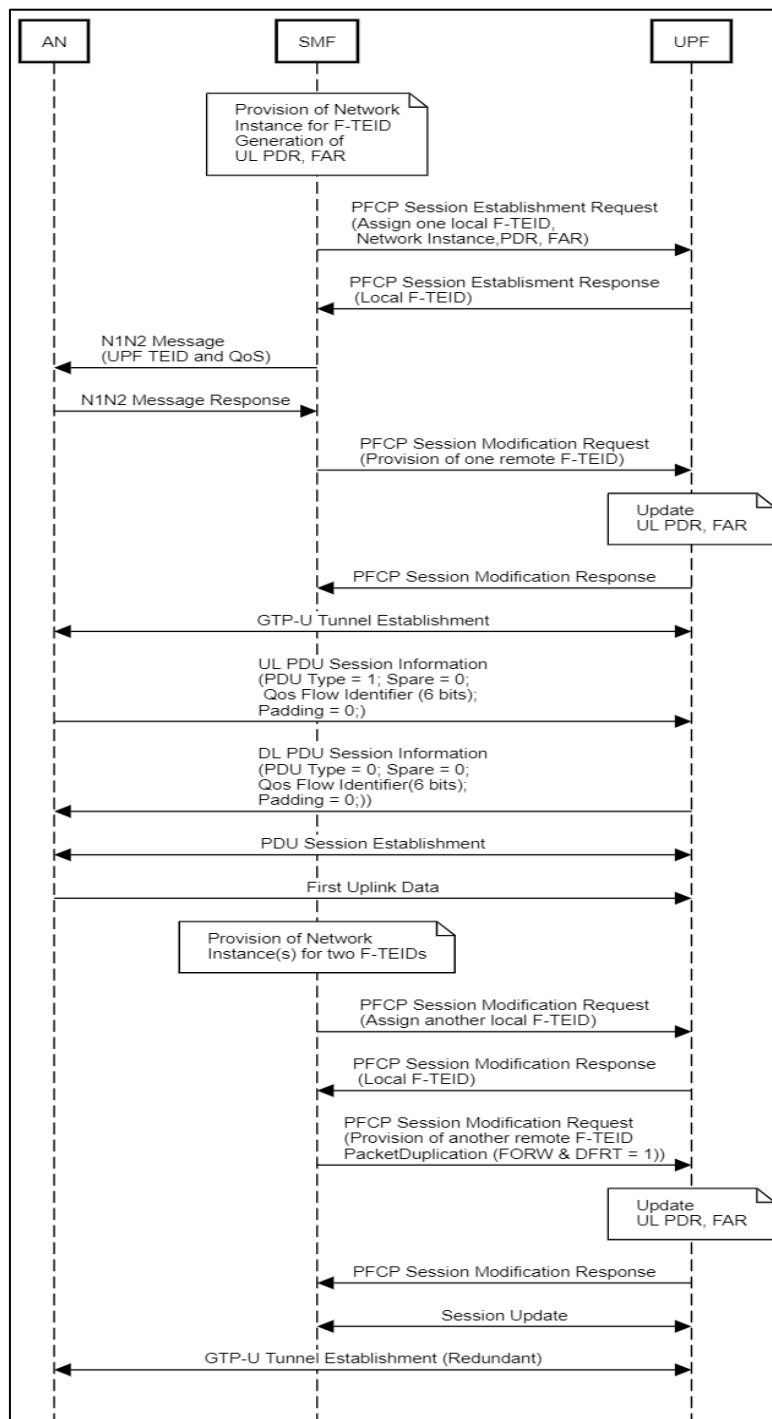


Figure 3-3 Redundant GTP – U Tunnel Setup Procedure

3.3.2. Redundant Transmission support only on N3 Interface

This use case refers to the redundant tunnel is set – up on the N3 interface. As indicated, the UPF and the NG – RAN should have redundant support functionality. Figure 3 – 6 depicts the NFs, and the connections involved in such a case.

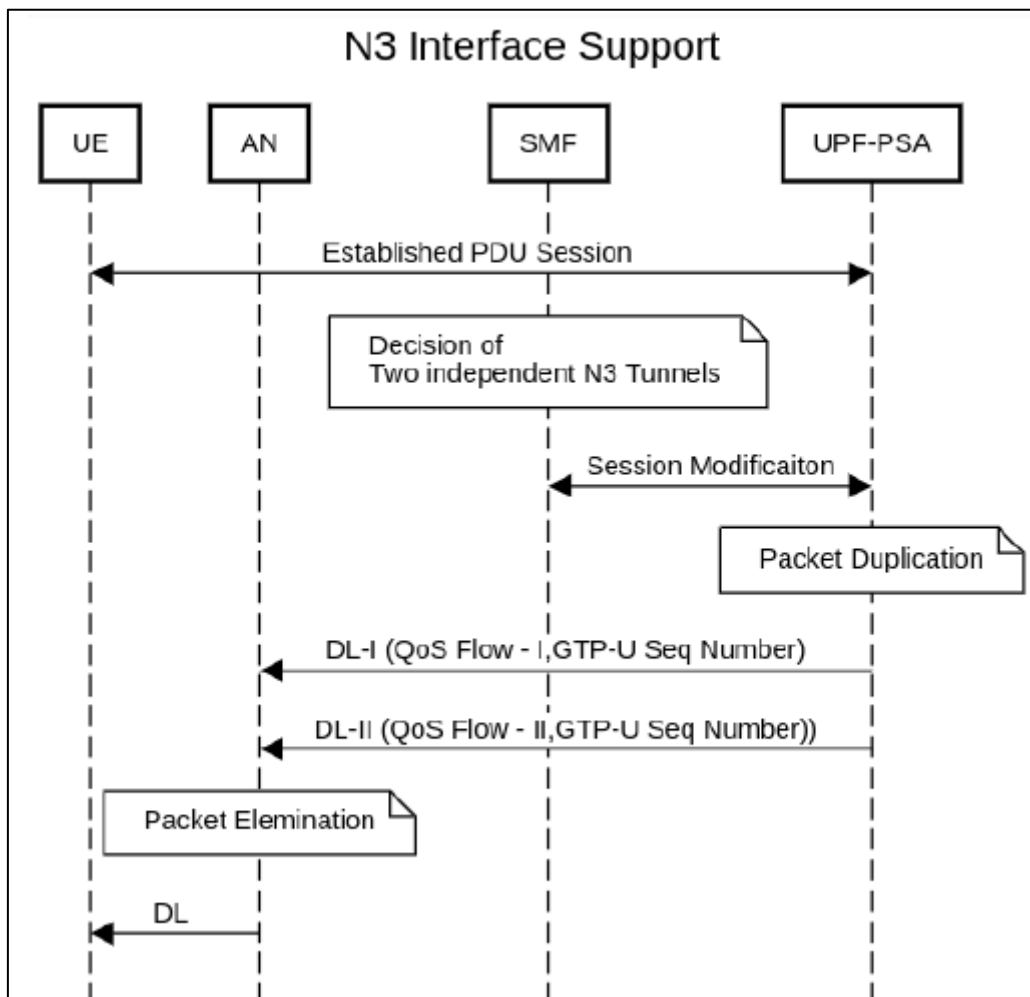


Figure 3-4 The call flow for Redundancy support on N3 Interface

The procedure steps for the redundant transmission support on N3 can be summarized as listed.

1. The UE establishes a URLLC service,
2. The AMF selects the SMF accordingly,
3. The SMF decides that the connection requires support on the N3 interface,
4. The SMF indicates NG – RAN and UPF,
5. The UPF provides TEID CN for the redundant tunnel to the SMF,
6. The SMF provides Routing Info, Tunnel Info of the redundant tunnel to the NG – RAN,

3.3.3. Redundant Transmission support on N3/N9 Interface

This use case refers to that the redundant tunnel is set – up on both N3 and N9 interfaces. This requires Intermediate UPFs to be involved in the CN architecture. I – UPF does not have to

necessarily support the redundant transmission support functionality. The duplication and elimination are processed the UPF – PSA where I – UPFs²⁷ are connected via N9 tunnels.

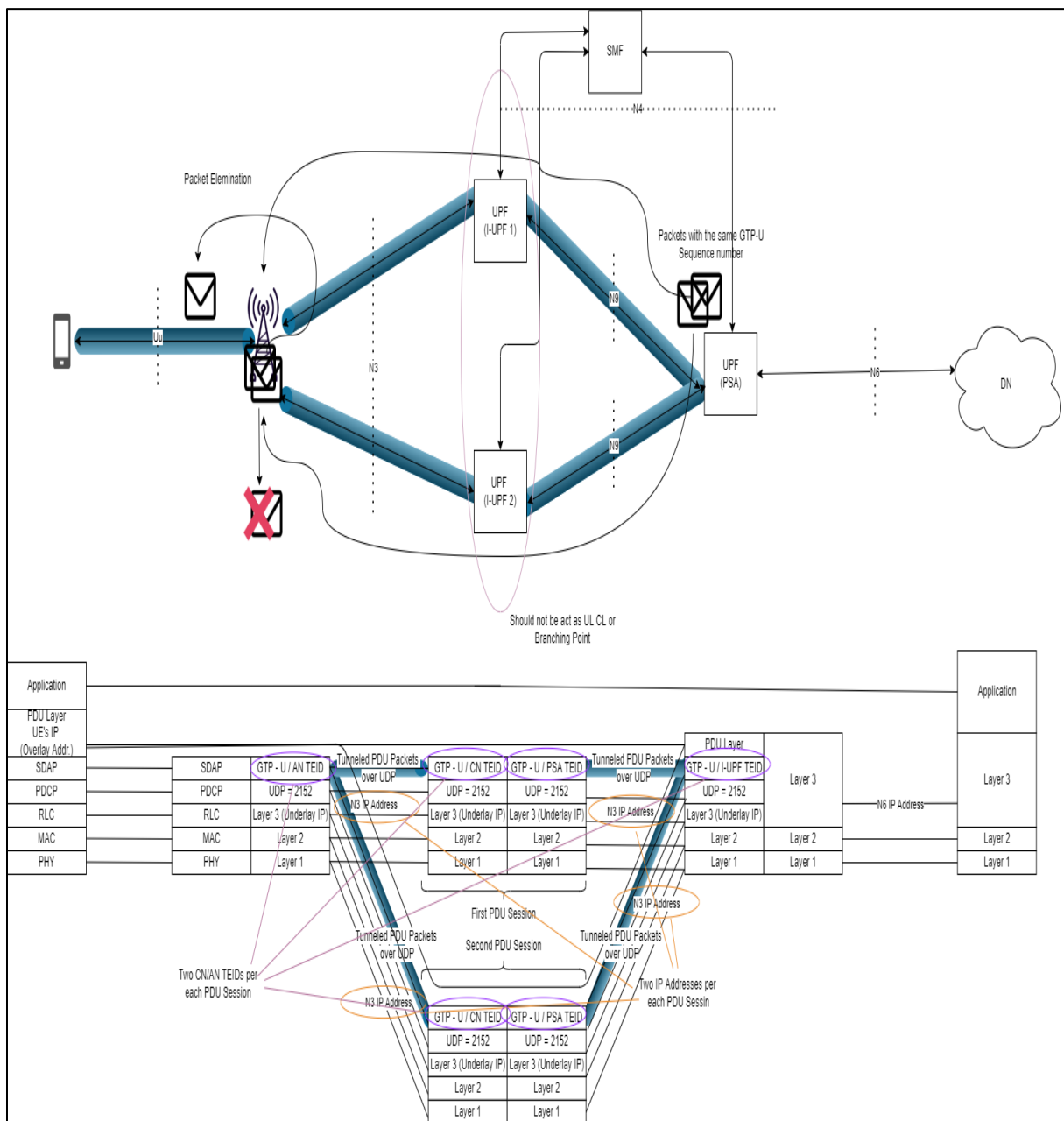


Figure 3-5 Redundancy support on N3/N9 Interfaces

²⁷ The I-UPFs inserted on one leg of the redundant paths shall not behave in an UL CL or Branching Point role.

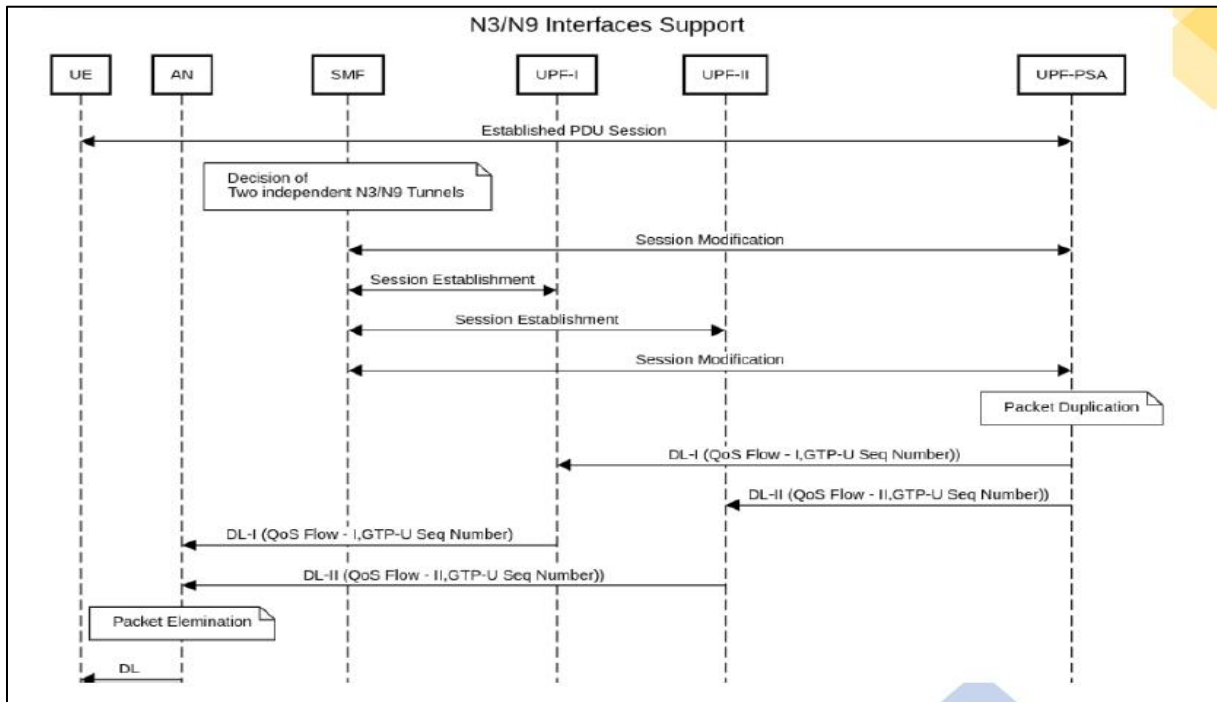


Figure 3-6 The call flow for Redundancy support on N3/N9 Interfaces

3.4. Support for redundant transmission at the transport layer

Redundant transmission support at transport layer technique refers to that redundancy is provided over the same GTP – U tunnel for the same PDU Session but the NG – RAN and UPF have such functionality to provide two disjoint transport paths within the same GTP – U tunnel with packet duplication and elimination capabilities. Figure 3 – 7 depicts the NFs and the connections involved in such a case.

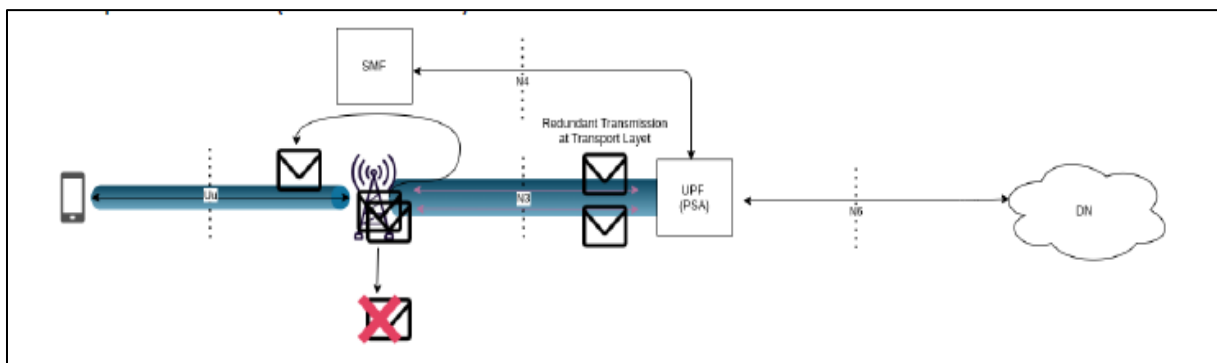


Figure 3-7 Redundancy support on the Transport Layer

The procedure steps for the redundant transmission support at the transport layer are listed below.

1. The UE establishes a URLLC service.
2. The AMF selects the SMF accordingly.

3. The SMF selects a UPF capable of providing redundant transmission support at the transport layer. The information that the UPF supports the redundant transmission support at the transport layer can be inserted into the SMF configuration. Otherwise, the UPF must indicate the SMF in the N4 Association Setup Response message.
4. The SMF decides that the connection PDU requires redundant transmission support at the transport layer.
5. The SMF indicates NG – RAN and UPF.
6. The service starts.

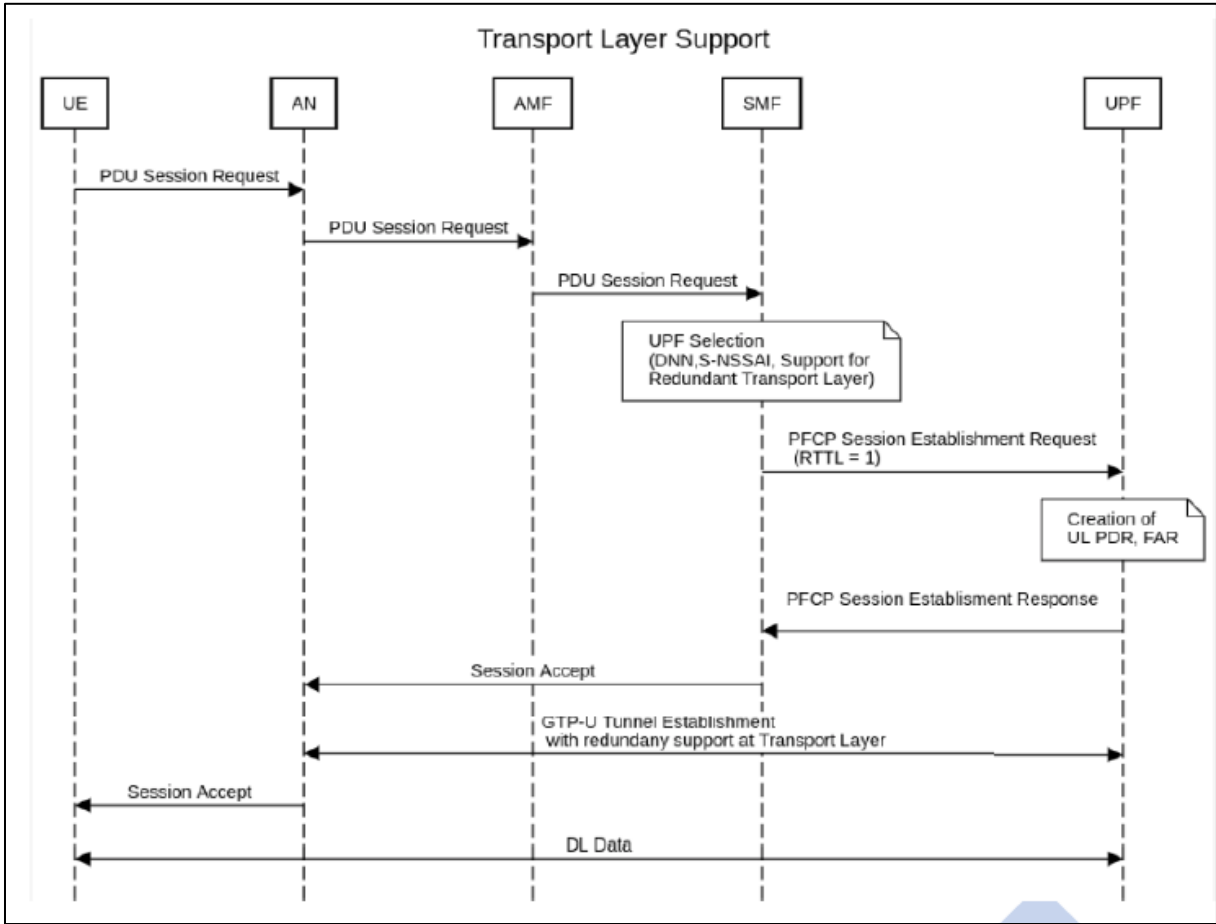


Figure 3-8 The call flow for Redundancy support on the Transport Layer

	Dual Connectivity	Support on N3/N9	Support on Transport
PDU Session(s)	2	1	1
Duplication / Elimination	Redundancy Sequence Number	GTP - U Sequence Number	Implementation dependent
The decision for the deployment	by the UE	by the SMF	by the UE/SMF

Table 3-2 Comparison of Techniques

URLLC QoS Monitoring

The calculation of UL, DL, or round-trip packet delay for a QoS Flow is operated by the UPF. In the case of redundant transmission support on N3/N9 interfaces, packet delay monitoring for both UL paths is performed by The UPF too. And then, it reports the packet delay of the two UP paths to the SMF, respectively.

4 Open-source Software Tools and Testing Environment

The theoretical aspects of the 5G CN focusing on URLLC have been discussed in the previous chapters. This chapter will introduce the software tools used to virtualize the RAN and 5G CN and implement the redundant transmission.

4.1. 5G AN Software Tool – UERANSIM

UERANSIM is an open-source implementation of the 5G Standalone UE (3GPP Access) and the 5G Standalone RAN (gNB), or simply implementing a 5G mobile phone and a base station. As it is indicated in [14], "UERANSIM introduces the world's first and only open source 5G – SA UE and gNB implementation.

4.1.1. Overview

The virtualized 5G AN - UERANSIM has three interfaces.

- **Control Interface:** The N2 interface between (R)AN and AMF implements SCTP protocol with default port number 38412.
- **User Interface:** The N4 between (R)AN and UPF implements GTP protocol with assigned port numbers 2152.
- **Radio Interface:** The NG - RA interface between UE and (R)AN implements RLS protocol with assigned port number 4997.

Two protocols/layers are defined for the Control Plane.

- NAS layer (related protocols²⁸) is for the control messages between the UE and the gNB,
- NGAP protocol is for the control messages between the gNB (AN) and the AMF (CN).

The features of the UERANSIM are limited. It does not provide all the services defined for the 5G System. According to [14], the features implemented within the UERANSIM are as follows.

- NAS Features
 - Primary Authentication and Key Agreement
 - Security Mode Control
 - Identification
 - Generic UE Configuration Update
 - Initial and Periodic Registration
 - UE and network-initiated De-registration
 - UE initiated PDU session establishment
 - UE and network triggered PDU session release
 - Service Request
 - Paging

²⁸ See section 2.1.3.

- For NAS Security,
- All integrity and ciphering algorithms are implemented. (IA1, IA2, IA3, EA1, EA2, EA3)
- All primary authentication and key agreement procedures are implemented. (5G AKA, EAP AKA')
- For identification,
- NAI and ciphered SUCI are not implemented yet. All other identity types can be used (IMSI, IMEI, IMEISV)
- For registration,
- Initial registration, mobility update registration, periodic registration is implemented.
- Emergency registration is not implemented yet.
- NGAP Features
- PDU Session Resource Setup
- PDU Session Resource Release
- Initial Context Setup
- UE Context Release (NG-RAN node initiated and AMF initiated)
- UE Context Modification
- Initial UE Message
- Paging
- Downlink NAS Transport
- Uplink NAS Transport
- NAS Non-Delivery Indication
- Reroute NAS Request
- NG Setup
- Error Indication

User Plane supports IPv4, and GTP – U protocol is top of it. RLC is the only protocol implemented for the Radio Interface.

4.1.2. Configuration

To utilize UERANSIM, elements of the implementation, namely UE and gNB, should be appropriately configured. The configuration parameters are defined in [14] as follows.

UE Configuration

SUPI - Subscription Permanent Identifier: It is the International Mobile System Identifier – IMSI number of the UE since only IMSI as a SUPI is supported. The parameter must be filled.

MCC: Mobile Country Code value of HPLMN (Home PLMN - Public Land Mobile Network). MCC should be consistent with SUPI.

MNC: Mobile Network Code value of HPLMN (Home PLMN). It is either two digits or three digits. MCC should be consistent with SUPI.

KEY: This is the permanent subscription key assigned for this specific UE.

OP: Operator code value (in either OP or OPC).

AMF: It is the Authentication Management Field (AMF).

IMEI - International Mobile Equipment Identity: IMEI number of the device. It is used when SUPI is not provided.

GNB: The list of gNBs' IP addresses.

SEARCH LIST: The UE selects a gNB from this list. The IP address and the linkIp parameter in the gNB configuration file must match to establish a connection between the UE and the gNB.

UACAIC: Unified Access Control (UAC), Access Identities Configuration (AIC) shall be configured in this field.

UACACC: Unified Access Control (UAC), Access Control Class (ACC) shall be configured in this field.

SESSIONS: This is the list of initial PDU sessions. These sessions are automatically established after the UE successfully registers to the 5G core network. Multiple PDU sessions can be added to this list. (No more than 15). Only IPv4 is supported for now as a PDU session type, apn field is optional and specifies the Access Point Name, slice parameter is optional and sets the S-NSSAI for this specific PDU session. In addition, the sd parameter is optional as well, but sst is always required if the slice parameter is included.

CONFIGURE NSSAI: The configured NSSAI for this UE by HPLMN (Home PLMN).

DEFAULT NSSAI: The default NSSAI for the UE.

INTEGRITY/CIPHERING: It sets integrity and ciphering algorithms for a specific UE. Typically, UERANSIM supports all algorithms, but part of them can be disabled manually in the configuration.

INTEGRITY: This field specifies the maximum data rate of integrity protection for the user plane for this UE. Possible values are 64kbps and full as

MAX RATE: It is specified by 3GPP TS 24.501.

gNB Configuration

MCC/MNC: They are explained in UE Configuration. They should match with the values in the UE configuration file.

NCI: It is the NR Cell Identity. Also, length specifies the bit length of the gNB ID, which is a part of this NR Cell Identity.

TAC: Tracking Area Code (TAC) value for the cell.

LINKIP: Local IP for radio link simulation for the gNB. The UE should connect to this IP address for radio simulation. The IP address should match with the gnbSearchList parameter.

NGAPIP: The N2 interface IP Address of the gNB. AMF will send SCTP packets to this IP address.

GTPIP: The N3 interface IP address of the gNB. UPF will send GTP packets to this IP address.

GTP: To separate the GTP listening address and external access address, it can override the DL User-Plane Transport Layer.

ADVERTISE IP: The optional address which is advertised to UPF. It is used in the case of a NAT between gNB and UPF.

AMF CONFIG: List of AMF IP addresses and port information for the N2 (NGAP) interface. The peer of the ngapIp address.

SLICES: It is explained in the UE configuration.

IGNORE STREAMIDS: Some core networks may not handle SCTP stream numbers properly. Set this field to true if you want to ignore such errors.

4.2. 5G CN Software Tool – Open5GS

Open5GS is an open-source implementation of the core network of either the 4G/LTE EPC System or the 5G System. C-language is used for the implementation of 5GC and EPC. In addition, WebUI is implemented in Node.JS and React, which can be utilized for testing purposes.

4.2.1. Overview

The open-source software of Open5GS consists of a set of software modules that implement some services of NFs of the 4G/5G NSA and 5G SA architecture. These modules can be installed with packet managers in Debian/Ubuntu²⁹. Once the modules are installed in the Linux system, they can be managed as daemons with service manager systemd.

The modules are the following:

AMF Daemon – amfd: It simulates some functionalities of the AMF by providing the N2 interface towards the eNB and Service Base Interface – SBI (N8, N11, N12, N15) for interactions with the other NFs in the 5G CN.

SMF Daemon – smfd: It simulates some functionalities of the SMF by providing the N11 interface towards the AMF, the N4 interface towards the UPF, and Service Base Interface – SBI (N7, N10) for interactions with the other NFs in the 5G CN.

²⁹ In the thesis, the environment for the implementation is Ubuntu, version 18.04.3 LTS (Long Term Support).

UPF Deamon – upfd: It simulates some functionalities of the UPF by providing the N11 interface towards the SMF and the N3 interface towards the gNB.

NRF Deamon – nrfd: It simulates some functionalities of the NRF by providing Service Base Interface – SBI (N22) for interactions with the other NFs in the 5G CN.

AUSF Deamon – ausfd: It simulates the some functionalities of the AUSF by providing Service Base Interface – SBI (N12, N14) for interactions with the other NFs in the 5G CN.

UDM Deamon – udmd: It simulates some functionalities of the AUSF by providing Service Base Interface – SBI (N12, N14) for interactions with the other NFs in the 5G CN.

PCF Deamon – pcf³⁰: It simulates some functionalities of the PCF by providing Service Base Interface – SBI (N7, N15, N35) for interactions with the other NFs in the 5G CN.

NSSF Deamon – nssfd: It simulates some functionalities of the NSSF by providing Service Base Interface – SBI (N22) for interactions with the other NFs in the 5G CN.

UDR Deamon – pcf: It simulates some functionalities of the UDR by providing Service Base Interface – SBI (N35, N36) for interactions with the other NFs in the 5G CN.

BSF Deamon – bsfd: It simulates some functionalities of the BSF.

This thesis intends to implement URLLC Service(s) by utilizing redundancy techniques within the 5G System. For the implementation, 5G SA architecture is used. To be consistent, only the components of Open5GS required to implement 5G SA³¹ are detailed.

In the implementation, MongoDB³² is utilized to store UE data used for the authentication and authorization of the UE when it requests to connect the 5G System.

³⁰ The daemon was modified by **Daniele Rossi** to fix the bug on the software in case of multiple PDU Sessions for multiple users. The modified pcf could become the main part of the Open5GS software.

³¹ 5G SA stands for 5G Standalone network.

³² Mongo DB is an integrated suite of cloud database [18].

VM1

This machine is used to implement a UE and a gNB by exploiting the features of UERANSIM. The UE and the gNB are connected via 127.0.0.0/8 IP network. Thus, the machine simulates AN of the 5G System. On the other hand, the N2 interface of the gNB needs to bind the interface of the VM1 that is connected to VM2 so the CN of the 5G System. This binding is instructed in the configuration file of the gNB. Moreover, the gNB needs to discover the AMF connected to the 5G CN. IP address and the port number must be pre-configured on the GNB for the discovery and association establishment processes between the GNB and the AMF.

```
linkIp: 127.0.0.1 # gNB's local IP address for Radio Link Simulation (Usually same with local IP)
ngapIp: 10.15.2.41 # gNB's local IP address for N2 Interface (Usually same with local IP)
gtpIp: 10.15.2.41 # gNB's local IP address for N3 Interface (Usually same with local IP)
# List of AMF address information
amfConfigs:
# - address: 127.0.0.5
#   port: 38412
- address: 10.15.2.53
  port: 38412
```

Figure 4-3 The gNB configuration

On the other hand, the UE needs to find the gNB to register to the 5GS and utilize the services. The UE also needs to be instructed on how to find the GNB. The connection between the UE and the gNB is implemented via the local IP network of the VM1.

```
# List of gNB IP addresses for Radio Link Simulation
gnbSearchList:
- 127.0.0.1
```

Figure 4-4 The UE configuration

In VM1, the interface ens3 is used for both N2 and N3 interfaces of the 5G System architecture. Additionally, the SSH connection between VM1 and VM2 is over this interface. When a UE requests an establishment of a PDU Session towards the DN, a GTP tunnel is established between the UE in the 5G AN and the UPF in 5G CN. The underlying IP network is the same network infrastructure used between the gNB and the AMF. To implement the GTP Tunnel, a universal TUN/Tap DEVICE driver is utilized on both VMs. The interface tunnel is called ogstun.

The description of the TUN/TAP device according to [16] is below.

“TUN/TAP provides packet reception and transmission for user space programs. It can be seen as a simple Point-to-Point or Ethernet device. Instead of receiving packets from physical media, it receives them from the user space program and instead of sending packets via physical media writes them to the user space program.”

The network infrastructure and related interfaces of the VM1 are depicted in Fig 4 -4. Interfaces and the bindings are listed in *Table 4 – 2*.

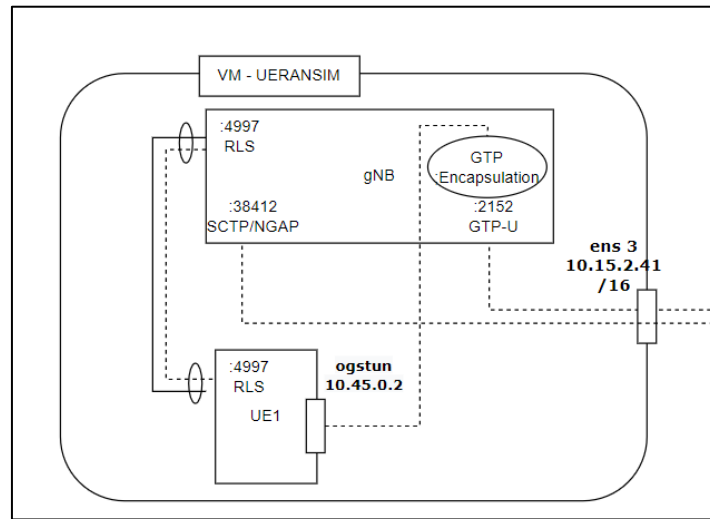


Figure 4-5 VM1

Int name	IP Address	Function
VM1 – lo	127.0.0.1/8	Loopback Address of VM1
VM1 – ens3	10.15.2.41/16	for N4
gNB-gtpu	10.15.2.41/16	:2152 for N4
gNB-ngap	10.15.2.41/16	:38412 for N2
gNB-RLS	127.0.0.1/8	:4997 for Uu
UE - RLS	127.0.0.2/8	:4997 for Uu
UE - ogstun	10.45.0.2/32	Tunnel Interface

Table 4-1 VM1 Interfaces

VM2

This machine is used for the implementation of NFs and also the database for the users (MongoDB); in other words, it is the bridge for the UE to reach the DN. Thus, the machine simulates the CN of the 5G System by exploiting the features of Open5GS. The implementation of the 5G CN has involved the deployment of most of the NFs, as they are detailed in Chapter 2. As expected, the connection and the interactions within the VM2 are more complex than the VM1.

The NFs are connected through their Service Base Interfaces – SBI to 127.0.0.0/8 IP network. All control messages between NFs are exchanged through this network. Besides, the NFs collect the required information to discover other NFs by consulting the NRF. IP address and port number for SBI and the NRF need to be defined in the configuration file of the AMF. In fact, network parameters of the NRF need to be defined in the configuration file of each NFs.

Moreover, the AMF needs to know the network-related parameters of the gNB for the same reason described for the gNB. On the other hand, the N2 interface of the AMF needs to bind the interface of the VM2 that is connected to VM1 so the AN of the 5G System. This binding also is instructed in the configuration file of the AMF.

```
nrf:
  sbi:
    - addr: 127.0.0.10
      port: 7777
amf:
  sbi:
    - addr: 127.0.0.5
      port: 7777
  ngap:
    - addr: 10.15.2.53
      port: 38412
```

Figure 4-6 The AMF Configuration

On the other hand, the SMF and the UPF need to be appropriately configured to operate the procedure, such as association or PDU session establishment on the N4 interface. Thus, both NFs are instructed to be able to discover each other and to establish PFCP sessions. In the configuration file of the SMF, network parameters of the NRF need to be defined for the same reasons as the AMF. GTP – U Tunnel related parameters for the possible establishment of GTP -U between the SMF and UPF. This tunnel is utilized when the SMF requests the UPF to forward determined data flows due to packet processing or QoS. The SMF needs to provide an IP address to the UE during the PDU Session establishment. The IP allocation is done on the subnet defined in the configuration.

Moreover, the association between the related interface and the name of the data network domain needs to be done. This information is provided to the UPF for a specific PDU Session to handle the packet processing. And finally, the IP address of the domain name server is configured to identify the destination address of the UL packets.

The UPF is the peer for the UE in GTP – U Tunnel, so it needs to be directly connected to the VM1. In the UPF, decapsulated GPDUs are forwarded to the DN. For correct operation, the association between the related interface and the name of the data network domain needs to be configured. Note that the parameters for NRF discovery do not place in the configuration file since the SMF initiates the node level association.

```

smf:
  sbi:
    - addr: 127.0.0.4
      port: 7777
  gtpu:
    - addr: 127.0.0.4
      port: 2152
  subnet:
    - addr: 10.45.0.1/16
      dnn: internet
  dns:
    - 8.8.8.8
    - 8.8.4.4
  nrf:
    sbi:
      - addr:
        - 127.0.0.10
        port: 7777
  upf:
    pfcf:
      - addr: 127.0.0.7
        port: 8805
        dnn: [internet]

```

Figure 4-7 The SMF Configuration

```

upf:
  pfcf:
    - addr: 127.0.0.7
      port: 8805
  gtpu:
    - addr: 10.15.2.53
      port: 2152
  subnet:
    - addr: 10.45.0.1/16
      dnn: internet
      dev: ogstun
  smf:
    pfcf:
      - addr: 127.0.0.4
        port: 8805

```

Figure 4-8 The UPF Configuration

The network infrastructure and related interfaces of the VM2 are depicted in Figure 4-8. In VM2, the interface ens3 is for the same functionalities explained for VM1(for N2 and N3 interfaces and SSH connection). The interface ens4 is utilized for the N6 interface of the 5GS and the SSH connection between the local machine and the VM2. Interfaces and the bindings are listed in Table 4-3.

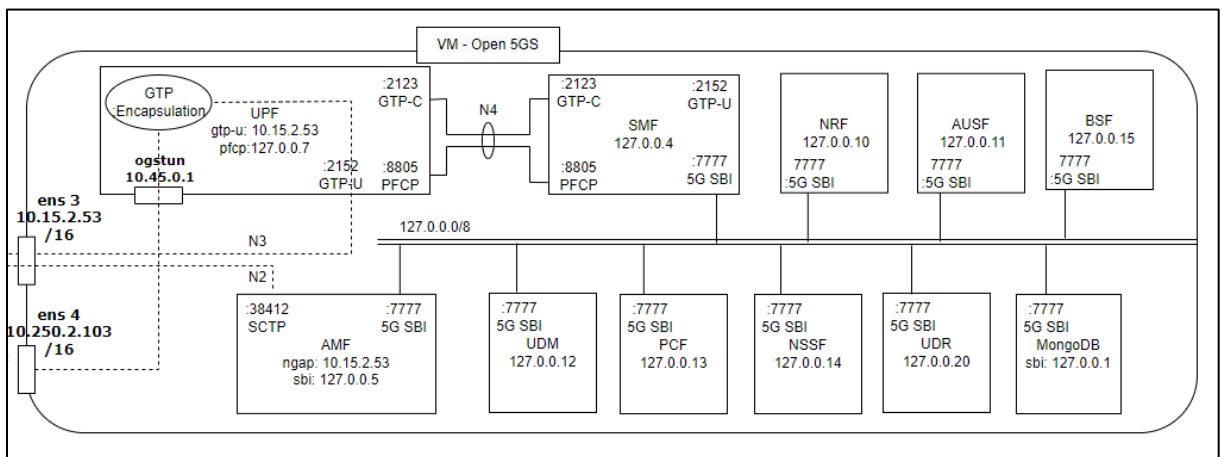


Figure 4-9 VM2

Int name	IP Address:Port	Function
VM2 – ens3	10.15.2.53/16	for N4
VM2 – ens4	10.250.2.103/16	for N6
SMF-gtpu	127.0.0.4:2152	for N4u (Sxu)
SMF-pfcp	127.0.0.4:8805	for N4 (Sxb)
SMF-sbi	127.0.0.4:7777	for 5G SBI (N7, N10, N11)
AMF-ngap	127.0.0.5:38412	for N2
AMF-sbi	127.0.0.5:7777	for 5G SBI (N8, N12, N11)
UPF-pfcp	127.0.0.7:8805	for N4
UPF-gtpu	127.0.0.7:2152	for N3
NRF-sbi	127.0.0.10:7777	for 5G SBI (N22)
AUSF-sbi	127.0.0.11:7777	for 5G SBI (N12, N13)
UDM-sbi	127.0.0.12:7777	for 5G SBI (N10, N13, N35)
PCF-sbi	127.0.0.13:7777	for 5G SBI (N7, N15, N35)
NSSF-sbi	127.0.0.14:7777	for 5G SBI (N22)
BSF-sbi	127.0.0.15:7777	for 5G SBI
UDR-sbi	127.0.0.20:7777	for 5G SBI (N35, N36)
MongoDB	127.0.0.1	http://localhost:3000

Table 4-2 VM2 Interfaces

IPTables

Iptable rules determine the packet processing rules for the interfaces of the VM2.

```
ubuntu@onur-vm:/var/log/open5gs$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -s 10.45.0.0/16 -j ACCEPT
-A INPUT -i ogstun -j ACCEPT
-A FORWARD -i ogstun -o ens4 -j ACCEPT
-A FORWARD -i ens4 -o ogstun -j ACCEPT
-A FORWARD -i ens4 -o ogstun
-A FORWARD -i ogstun -o ens4
```

```
ubuntu@onur-vm:/var/log/open5gs$ sudo iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
-A INPUT -s 10.45.0.0/16 -j ACCEPT
-A INPUT -i ogstun -j ACCEPT
-A FORWARD -i ogstun -o ens4 -j ACCEPT
-A FORWARD -i ens4 -o ogstun -j ACCEPT
-A FORWARD -i ens4 -o ogstun
-A FORWARD -i ogstun -o ens4
```

4.3.2. Setting to Work

In this section, the software components (daemons) of Open5Gs and UERANSIM are activated on VM1 and on VM2 step by step in order to demonstrate the interaction and procedures among NFs, and between 5G AN and 5G CN. The outcomes of the log file of the daemons are attached to ease the explanation of the concepts.

VM2 is the host machine for virtualizing the 5G CN, and the NFs deployed in it need to be started to provide 5GS services to the gNB and the UE hosted by VM1. Thus, the implementation begins with a set of operations on VM2.

- Nrfd is activated on VM2.

```
sudo systemctl start open5gs-nrfd
```

The NRF should be active since other NFs utilize it to obtain the information for discovering and selecting corresponding NFs to provide or consume services. After being started, Each NF registers itself to the NRF. The NF provides an IP and a port address for the discovery procedure with the interaction.

```
01/18 18:46:54.737: [nrf] INFO: [01ca42b0-788f-41ec-ac86-2b88cee08c4e] NF registred [Heartbeat:10s] (./src/nrf/nf-sm.c:193)
01/18 18:47:11.915: [nrf] INFO: [0c07ea48-788f-41ec-96a1-5bddf4ff3a97] NF registred [Heartbeat:10s] (./src/nrf/nf-sm.c:193)
01/18 18:47:49.507: [nrf] INFO: [226fdcfa-788f-41ec-99ab-adb0da80e38a] NF registred [Heartbeat:10s] (./src/nrf/nf-sm.c:193)
01/18 18:48:11.022: [nrf] INFO: [2f42f26e-788f-41ec-9a90-b59f7c8ce77e] NF registred [Heartbeat:10s] (./src/nrf/nf-sm.c:193)
01/18 18:48:21.333: [nrf] INFO: [35684360-788f-41ec-b23f-e553ecd9c263] NF registred [Heartbeat:10s] (./src/nrf/nf-sm.c:193)
01/18 18:49:35.339: [nrf] INFO: [6184506a-788f-41ec-819f-112594d79a56] NF registred [Heartbeat:10s] (./src/nrf/nf-sm.c:193)
01/18 18:51:04.273: [nrf] INFO: [96866cd0-788f-41ec-b35b-8d2a4e0d979a] NF registred [Heartbeat:10s] (./src/nrf/nf-sm.c:193)
01/18 18:51:16.087: [nrf] INFO: [9d838a7c-788f-41ec-9e17-87315d68b5ce] NF registred [Heartbeat:10s] (./src/nrf/nf-sm.c:193)
```

Figure 4-10 Log output of Nrfd (after all NFs are activated)

- Bsfed is activated on VM2.

```
sudo systemctl start open5gs-bsfd
```

The BSF daemon is required for binding PDU sessions and related policies. Once started, It registers itself to the nrfd.

```
Open5GS daemon v2.3.6
01/18 18:48:11.019: [app] INFO: Configuration: '/etc/open5gs/bsf.yaml' (./lib/app/ogs-init.c:129)
01/18 18:48:11.019: [app] INFO: File Logging: '/var/log/open5gs/bsf.log' (./lib/app/ogs-init.c:132)
01/18 18:48:11.021: [sbi] INFO: nhttp2_server() [127.0.0.15]:7777 (./lib/sbi/nhttp2-server.c:145)
01/18 18:48:11.022: [app] INFO: BSF initialize...done (./src/bsf/app.c:31)
01/18 18:48:11.022: [bsf] INFO: [2f42f26e-788f-41ec-9a90-b59f7c8ce77e] NF registred [Heartbeat:10s] (./src/bsf/nf-sm.c:200)
```

Figure 4-11 Log output of Bsfed

- Nssfd is activated on VM2.

```
sudo systemctl start open5gs-nssfd
```

This daemon is required for network slice selection. Once started, it registers itself to the Nrfd.

```

Open5GS daemon v2.3.6
01/18 18:47:49.504: [app] INFO: Configuration: '/etc/open5gs/nssf.yaml' (./lib/app/ogs-init.c:129)
01/18 18:47:49.504: [app] INFO: File Logging: '/var/log/open5gs/nssf.log' (./lib/app/ogs-init.c:132)
01/18 18:47:49.505: [sbi] INFO: nhttp2_server() [127.0.0.14]:7777 (./lib/sbi/nhttp2-server.c:145)
01/18 18:47:49.506: [app] INFO: NSSF initialize...done (./src/nssf/app.c:31)
01/18 18:47:49.507: [nssf] INFO: [226fdcfa-788f-41ec-99ab-adb0da80e38a] NF registered [Heartbeat:10s] (./src/nssf/nf-sm.c:199)

```

Figure 4-12 Log output of Nssfd

- Udrd and MongoDB are activated on VM2.

```

sudo systemctl start open5gs-udrd
sudo systemctl start mongod

```

Udrd is required for a UE subscription. It registers itself to the Nrfd. MongoDB should be active to enable the UDR to collect the necessary information about the UEs during the registration procedure.

```

Open5GS daemon v2.3.6
01/18 18:46:54.732: [app] INFO: Configuration: '/etc/open5gs/udr.yaml' (./lib/app/ogs-init.c:129)
01/18 18:46:54.732: [app] INFO: File Logging: '/var/log/open5gs/udr.log' (./lib/app/ogs-init.c:132)
01/18 18:46:54.736: [dbi] INFO: MongoDB URI: 'mongodb://localhost/open5gs' (./lib/dbi/ogs-mongoc.c:130)
01/18 18:46:54.736: [sbi] INFO: nhttp2_server() [127.0.0.20]:7777 (./lib/sbi/nhttp2-server.c:145)
01/18 18:46:54.737: [app] INFO: UDR initialize...done (./src/udr/app.c:31)
01/18 18:46:54.738: [udr] INFO: [01ca42b0-788f-41ec-ac86-2b88cee08c4e] NF registered [Heartbeat:10s] (./src/udr/nf-sm.c:199)

```

Figure 4-13 Log output of Udrd

- Udmd is activated on VM2.

```

sudo systemctl start open5gs-udmd

```

This daemon is required for a UE to be authorized. It registers itself to the Nrfd and establishes an association with the Udrd.

```

Open5GS daemon v2.3.6
01/18 18:47:11.913: [app] INFO: Configuration: '/etc/open5gs/udm.yaml' (./lib/app/ogs-init.c:129)
01/18 18:47:11.913: [app] INFO: File Logging: '/var/log/open5gs/udm.log' (./lib/app/ogs-init.c:132)
01/18 18:47:11.914: [sbi] INFO: nhttp2_server() [127.0.0.12]:7777 (./lib/sbi/nhttp2-server.c:145)
01/18 18:47:11.914: [app] INFO: UDM initialize...done (./src/udm/app.c:31)
01/18 18:47:11.915: [udm] INFO: [0c07ea48-788f-41ec-96a1-5bddf4ff3a97] NF registered [Heartbeat:10s] (./src/udm/nf-sm.c:199)
01/18 18:59:44.644: [app] WARNING: Try to discover [UDR] (./lib/sbi/path.c:111)
01/18 18:59:44.645: [udm] INFO: [01ca42b0-788f-41ec-ac86-2b88cee08c4e] (NF-discover) NF registered (./src/udm/nrf-handler.c:284)
01/18 18:59:44.645: [udm] INFO: [01ca42b0-788f-41ec-ac86-2b88cee08c4e] (NF-discover) NF Profile updated (./src/udm/nrf-handler.c:330)

```

Figure 4-14 Log output of Udmd

- Ausfd is activated on VM2.

```

sudo systemctl start open5gs-ausfd

```

This daemon is required for a UE to be authenticated. It registers itself to the Nrfd and establishes an association with the Udmd.

```

Open5GS daemon v2.3.6
01/18 18:48:21.331: [app] INFO: Configuration: '/etc/open5gs/ausf.yaml' (./lib/app/ogs-init.c:129)
01/18 18:48:21.331: [app] INFO: File Logging: '/var/log/open5gs/ausf.log' (./lib/app/ogs-init.c:132)
01/18 18:48:21.332: [sbi] INFO: nhttp2_server() [127.0.0.11]:7777 (./lib/sbi/nhttp2-server.c:145)
01/18 18:48:21.332: [app] INFO: AUSF initialize...done (./src/ausf/app.c:31)
01/18 18:48:21.334: [ausf] INFO: [35684360-788f-41ec-b23f-e553ecd4c263] NF registered [Heartbeat:10s] (./src/ausf/nf-sm.c:199)
01/18 18:59:44.642: [app] WARNING: Try to discover [UDM] (./lib/sbi/path.c:111)
01/18 18:59:44.643: [ausf] INFO: [0c07ea48-788f-41ec-96a1-5bddf4ff3a97] (NF-discover) NF registered (./src/ausf/nrf-handler.c:283)
01/18 18:59:44.643: [ausf] INFO: [0c07ea48-788f-41ec-96a1-5bddf4ff3a97] (NF-discover) NF Profile updated (./src/ausf/nrf-handler.c:329)

```

Figure 4-15 Log output of Ausfd

- Pcfcd is activated on VM2.

```
sudo systemctl start open5gs-pcfd
```

This daemon is required for the SMF to get policies for a specific UE. It registers itself to the Nrfnd and establishes an association with the Udrd and Bsfd.

```
open5gs daemon v2.3.6
01/18 18:49:35.339: [app] INFO: Configuration: '/etc/open5gs/pcf.yaml' (./lib/app/ogs-init.c:129)
01/18 18:49:35.339: [app] INFO: File Logging: '/var/log/open5gs/pcf.log' (./lib/app/ogs-init.c:132)
01/18 18:49:35.338: [dbi] INFO: MongoDB URI: 'mongodb://localhost/open5gs' (./lib/dbi/ogs-mongoc.c:130)
01/18 18:49:35.338: [sbi] INFO: nghttp2_server() [127.0.0.13]:7777 (./lib/sbi/nghttp2-server.c:145)
01/18 18:49:35.338: [app] INFO: PCF initialize...done (./src/pcf/app.c:31)
01/18 18:49:35.340: [pcf] INFO: [6184506a-788f-41ec-819f-112594d79a56] NF registred [Heartbeat:10s] (./src/pcf/nf-sm.c:199)
01/18 18:49:45.258: [app] INFO: Signal-NUM[28] received (window changed) (./src/main.c:64)
01/18 18:59:44.090: [app] WARNING: Try to discover [UOR] (./lib/sbi/path.c:111)
01/18 18:59:44.091: [pcf] INFO: [01ca42b0-788f-41ec-ac86-2b88cee08c4e] (NF-discover) NF registred (./src/pcf/nrf-handler.c:286)
01/18 18:59:44.091: [pcf] INFO: [01ca42b0-788f-41ec-ac86-2b88cee08c4e] (NF-discover) NF Profile updated (./src/pcf/nrf-handler.c:346)
01/18 18:59:44.090: [app] WARNING: Try to discover [BSF] (./lib/sbi/path.c:111)
01/18 18:59:44.090: [pcf] INFO: [2f42f26e-788f-41ec-9a90-b59f7c8ce77e] (NF-discover) NF registred (./src/pcf/nrf-handler.c:286)
01/18 18:59:44.091: [pcf] INFO: [2f42f26e-788f-41ec-9a90-b59f7c8ce77e] (NF-discover) NF Profile updated (./src/pcf/nrf-handler.c:346)
01/18 19:26:45.561: [app] INFO: Signal-NUM[28] received (window changed) (./src/main.c:64)
```

Figure 4-16 Log output of Pcfnd

- Upfd and Smfd are activated on VM2.

```
sudo systemctl start open5gs-upfd
sudo systemctl start open5gs-smfd
```

Once the Upfd starts, it tries to establish an association with the Smfd according to the SMF's IP and port addresses in its configuration³³. Until the Smfd starts, the Upfd cannot receive the Association Response message from the peer. When the Smfd starts, it registers itself to the Nrfnd³⁴. In addition, GTP and PFCP servers are initialized; after that moment, the Smfd is ready to establish an association with the Upfd.

```
Open5GS daemon v2.3.6
01/18 19:07:14.976: [app] INFO: Configuration: '/etc/open5gs/smf.yaml' (./lib/app/ogs-init.c:129)
01/18 19:07:14.976: [app] INFO: File Logging: '/var/log/open5gs/smf.log' (./lib/app/ogs-init.c:132)
01/18 19:07:15.055: [gtp] INFO: gtp_server() [127.0.0.4]:2123 (./lib/gtp/path.c:31)
01/18 19:07:15.055: [gtp] INFO: gtp_server() [127.0.0.4]:2152 (./lib/gtp/path.c:31)
01/18 19:07:15.055: [pfc] INFO: pfc_server() [127.0.0.4]:8805 (./lib/pfc/path.c:31)
01/18 19:07:15.055: [pfc] INFO: ogs_pfc_connect() [127.0.0.7]:8805 (./lib/pfc/path.c:60)
01/18 19:07:15.056: [sbi] INFO: nghttp2_server() [127.0.0.4]:7777 (./lib/sbi/nghttp2-server.c:145)
01/18 19:07:15.056: [app] INFO: SMF initialize...done (./src/smf/app.c:31)
01/18 19:07:15.058: [smf] INFO: PFCP associated (./src/smf/pfc-sm.c:174)
01/18 19:07:15.058: [smf] INFO: [d91f777e-7891-41ec-8705-11ae50875750] NF registred [Heartbeat:10s] (./src/smf/nf-sm.c:200)
```

Figure 4-17 Log output of Smfd

```
01/18 19:07:05.021: [pfc] WARNING: [88] LOCAL No Response. Give up! for step 1 type 5 peer [127.0.0.4]:8805 (./lib/pfc/xact.c:618)
01/18 19:07:08.516: [upf] WARNING: Retry to association with peer [127.0.0.4]:8805 failed (./src/upf/pfc-sm.c:107)
01/18 19:07:15.056: [upf] INFO: PFCP associated (./src/upf/pfc-sm.c:173)
01/18 19:07:16.021: [pfc] WARNING: [89] LOCAL No Response. Give up! for step 1 type 5 peer [127.0.0.4]:8805 (./lib/pfc/xact.c:618)
01/18 19:07:33.546: [upf] INFO: UE F-SEID[CP:0x6 UP:0x1] APN[Internet] PDN-Type[1] IPv4[10.45.0.2] IPv6[] (./src/upf/context.c:366)
01/18 19:07:40.442: [upf] INFO: [added] Number of UPF-Sessions is now 4 (./src/upf/context.c:161)
01/18 19:07:40.443: [upf] INFO: UE F-SEID[CP:0x9 UP:0x2] APN[Internet] PDN-Type[1] IPv4[10.45.0.3] IPv6[] (./src/upf/context.c:366)
```

Figure 4-18 Log output of Upfd

³³ The Upfd association attempts fail before the Smfd starts.

³⁴ The registration is still required because NFs other than the UPF do not have the address information of the SMF in their configurations.

- Amfd and gNB are activated on VM2 and VM1, respectively.

```
sudo systemctl start open5gs-amfd
build/nr-gnb -c config/open5gs-gnb.yaml
```

```
01/18 19:07:10.721: [app] INFO: Configuration: '/etc/open5gs/amf.yaml' (./lib/app/ogs-init.c:129)
01/18 19:07:10.721: [app] INFO: File logging: '/var/log/open5gs/amf.log' (./lib/app/ogs-init.c:132)
01/18 19:07:10.723: [sbi] INFO: nghttp2_server() [127.0.0.5]:7777 (./lib/sbi/nghttp2-server.c:145)
01/18 19:07:10.723: [amf] INFO: ngap_server() [10.15.2.53]:38412 (./src/amf/ngap-sctp.c:54)
01/18 19:07:10.724: [sctp] INFO: AMF initialize...done (./src/amf/app.c:33)
01/18 19:07:10.725: [amf] INFO: [d693776c-7891-41ec-aa9d-bf8d3e7d9f4e] NF registered [Heartbeat:10s] (./src/amf/nf-sm.c:199)
01/18 19:07:15.058: [amf] INFO: [d91f777e-7891-41ec-8705-11ae50875750] (NRF-notify) NF registered (./src/amf/nrf-handler.c:182)
01/18 19:07:15.058: [amf] INFO: [d91f777e-7891-41ec-8705-11ae50875750] (NRF-notify) NF Profile updated (./src/amf/nrf-handler.c:201)
01/18 19:07:24.797: [amf] INFO: gNB-M2 accepted[10.15.2.41]:34698 in ng-path module (./src/amf/ngap-sctp.c:106)
01/18 19:07:24.797: [amf] INFO: gNB-M2 accepted[10.15.2.41] in master_sm module (./src/amf/amf-sm.c:608)
01/18 19:07:24.797: [amf] INFO: [GNB] max_num_of_ostreams : 30 (./src/amf/context.c:854)
01/18 19:07:24.797: [amf] INFO: [Added] Number of gNBs is now 1 (./src/amf/context.c:870)
01/18 19:07:33.307: [amf] INFO: InitialUEMessage (./src/amf/ngap-handler.c:361)
```

Figure 4-19 Log output of Amfd (after gNB is activated)

The AMF binds to port 38412 of the VM2 – ens3 interface to the Ngap server. This server is required to exchange control messages with the gNB. It registers itself to the NRF as other NFs. When gNB is started, it sends an NG Setup Request message to the AMF and receives NG Setup Response. NGAP messages are encapsulated in SCTP. These messages show that the initial configuration exchange between gNB and the AMF is successfully completed.

```
ubuntu@onur-access:~/UERANSIM$ build/nr-gnb -c config/open5gs-gnb.yaml
UERANSIM v3.2.4
[2022-01-18 19:07:24.794] [sctp] [info] Trying to establish SCTP connection... (
10.15.2.53:38412)
[2022-01-18 19:07:24.798] [sctp] [info] SCTP connection established (10.15.2.53:
38412)
[2022-01-18 19:07:24.798] [sctp] [debug] SCTP association setup ascId[16]
[2022-01-18 19:07:24.798] [ngap] [debug] Sending NG Setup Request
[2022-01-18 19:07:24.799] [ngap] [debug] NG Setup Response received
[2022-01-18 19:07:24.799] [ngap] [info] NG Setup procedure is successful
[2022-01-18 19:07:33.301] [rrc] [debug] UE[1] new signal detected
[2022-01-18 19:07:33.308] [rrc] [info] RRC Setup for UE[1]
[2022-01-18 19:07:33.308] [ngap] [debug] Initial NAS message received from UE[1]
[2022-01-18 19:07:33.332] [ngap] [debug] Initial Context Setup Request received
[2022-01-18 19:07:33.555] [ngap] [info] PDU session resource(s) setup for UE[1]
count[1]
```

Figure 4-20 Log output of gNB (after UE is activated)

- UE is activated on VM1.

```
build/nr-ue -c config/open5gs-ue.yaml
```

As soon as the UE is initialized, it starts the registration procedure by sending the NAS Registration Request message to the gNB in order to connect to the 5G network. The context of the message is transferred to the AMF with additional information. When the registration procedure is successfully completed, the UE starts the PDU Session establishment procedure by NAS PDU Session Resource Setup Request.

```

^Cubuntu@onur-access:~/UERANSIM$ sudo build/nr-ue -c config/opensgs-ue.yaml
UERANSIM v3.2.4
[2022-01-18 19:07:33.301] [nas] [info] UE switches to state [MM-DEREGISTERED/PLMN-SEARCH]
[2022-01-18 19:07:33.302] [rrc] [debug] New signal detected for cell[1], total [1] cells in coverage
[2022-01-18 19:07:33.302] [nas] [info] Selected plmn[001/01]
[2022-01-18 19:07:33.302] [rrc] [info] Selected cell plmn[001/01] tac[1] category[SUITABLE]
[2022-01-18 19:07:33.302] [nas] [info] UE switches to state [MM-DEREGISTERED/PS]
[2022-01-18 19:07:33.302] [nas] [info] UE switches to state [MM-DEREGISTERED/NORMAL-SERVICE]
[2022-01-18 19:07:33.302] [nas] [debug] Initial registration required due to [MM-DEREG-NORMAL-SERVICE]
[2022-01-18 19:07:33.307] [nas] [debug] UAC access attempt is allowed for identity[0], category[NO_sig]
[2022-01-18 19:07:33.308] [nas] [debug] Sending Initial Registration
[2022-01-18 19:07:33.308] [nas] [info] UE switches to state [MM-REGISTER-INITIATED]
[2022-01-18 19:07:33.308] [rrc] [debug] Sending RRC Setup Request
[2022-01-18 19:07:33.308] [rrc] [info] RRC connection established
[2022-01-18 19:07:33.308] [rrc] [info] UE switches to state [RRC-CONNECTED]
[2022-01-18 19:07:33.309] [nas] [info] UE switches to state [CM-CONNECTED]
[2022-01-18 19:07:33.317] [nas] [debug] Authentication Request received
[2022-01-18 19:07:33.321] [nas] [debug] Security Mode Command received
[2022-01-18 19:07:33.321] [nas] [debug] Selected Integrity[2] ciphering[0]
[2022-01-18 19:07:33.332] [nas] [debug] Registration accept received
[2022-01-18 19:07:33.332] [nas] [info] UE switches to state [MM-REGISTERED/NORMAL-SERVICE]
[2022-01-18 19:07:33.332] [nas] [debug] Sending Registration Complete
[2022-01-18 19:07:33.333] [nas] [info] Initial Registration is successful
[2022-01-18 19:07:33.333] [nas] [debug] Sending PDU Session Establishment Request
[2022-01-18 19:07:33.333] [nas] [debug] UAC access attempt is allowed for identity[0], category[NO_sig]
[2022-01-18 19:07:33.536] [nas] [debug] Configuration Update Command received
[2022-01-18 19:07:33.556] [nas] [debug] PDU Session Establishment Accept received
[2022-01-18 19:07:33.556] [nas] [info] PDU Session establishment is successful PDU Session ID[1]
[2022-01-18 19:07:33.579] [app] [info] Connection setup for PDU session[1] is successful, TUN interface[uesintun0, 10.45.0.2] is up.

```

Figure 4-21 Log output of UE including the Registration and PDU Session Establishment Procedures

SUCI is used to identify, authorize, and authenticate the UE profile. The AMF coordinates the registration with AUSF, UDM, and PCF by sharing the UE SUCI. When the registration procedure is successfully completed, the AMF receives a PDU session request. It indicates the SMF which is already selected.

```

01/18 19:07:33.307: [amf] INFO: [Added] Number of gNB-UEs is now 1 (./src/amf/context.c:2005)
01/18 19:07:33.307: [amf] INFO: RAN_UE_NGAP_ID[1] AMF_UE_NGAP_ID[1] TAC[1] CellID[0x10] (./src/amf/ngap-handler.c:500)
01/18 19:07:33.308: [amf] INFO: [suci-0-001-01-0000-0-0-0000000000] Unknown UE by SUCI (./src/amf/context.c:1385)
01/18 19:07:33.308: [gmm] INFO: Registration request (./src/amf/gmm-sm.c:130)
01/18 19:07:33.308: [gmm] INFO: [suci-0-001-01-0000-0-0-0000000000] SUCI (./src/amf/gmm-handler.c:156)
01/18 19:07:33.308: [app] WARNING: Try to discover [AUSF] (./lib/sbi/path.c:111)
01/18 19:07:33.309: [amf] INFO: [35684360-788f-41ec-b23f-e553ecd4c263] (NF-discover) NF registered (./src/amf/nrf-handler.c:342)
01/18 19:07:33.310: [amf] INFO: [35684360-788f-41ec-b23f-e553ecd4c263] (NF-discover) NF Profile updated (./src/amf/nrf-handler.c:402)
01/18 19:07:33.320: [app] WARNING: Try to discover [UDM] (./lib/sbi/path.c:111)
01/18 19:07:33.321: [amf] INFO: [0c07ea48-788f-41ec-96a1-5bdd44ff3a97] (NF-discover) NF registered (./src/amf/nrf-handler.c:342)
01/18 19:07:33.321: [amf] INFO: [0c07ea48-788f-41ec-96a1-5bdd44ff3a97] (NF-discover) NF Profile updated (./src/amf/nrf-handler.c:402)
01/18 19:07:33.326: [app] WARNING: Try to discover [PCF] (./lib/sbi/path.c:111)
01/18 19:07:33.327: [amf] INFO: [6184506a-788f-41ec-819f-112594d79a56] (NF-discover) NF registered (./src/amf/nrf-handler.c:342)
01/18 19:07:33.327: [amf] INFO: [6184506a-788f-41ec-819f-112594d79a56] (NF-discover) NF Profile updated (./src/amf/nrf-handler.c:402)
01/18 19:07:33.533: [gmm] INFO: [imsi-0010100000000000] Registration complete (./src/amf/gmm-sm.c:1015)
01/18 19:07:33.533: [amf] INFO: [imsi-0010100000000000] Configuration update command (./src/amf/nas-path.c:389)
01/18 19:07:33.533: [gmm] INFO: UTC [2022-01-18T19:07:33] Timezone[0]/DST[0] (./src/amf/gmm-build.c:506)
01/18 19:07:33.533: [gmm] INFO: LOCAL [2022-01-18T19:07:33] Timezone[0]/DST[0] (./src/amf/gmm-build.c:511)
01/18 19:07:33.533: [amf] INFO: [Added] Number of AMF-Sessions is now 1 (./src/amf/context.c:2017)
01/18 19:07:33.533: [gmm] INFO: UE SUPFI[imsi-0010100000000000] DNN[internet] S_NSSAI[SST:1 SD:0xffff] (./src/amf/gmm-handler.c:1042)

```

Figure 4-22 Log output of Amfd (after UE is activated)

Figure 4-22 shows that the registration and the PDU Session establishment procedures require exchanging NAS messages between the AMF and the UE after SCTP and NGAP associations.

PFPCP Association Setup

Figure 4-23 shows that node-level control messages between the SMF and UPF are exchanged to establish the association. These messages are PFCP Association Setup Request and PFCP Association Setup Response. PFCP Association Setup Request is sent by the SMF (127.0.0.4) to the UPF (127.0.0.7). When the association is completed, PFCP Heartbeat messages are

exchanged periodically. Both NFs use the same UDP port 8805 for node-level control messages³⁵.

No.	Time	Source	Destination	Protocol	Length	Info
102	30.893361	VM1 10.15.2.41	VM2 10.15.2.53	SCTP	82	INIT
103	30.893874	10.15.2.53	10.15.2.41	SCTP	306	INIT_ACK
104	30.894100	10.15.2.41	10.15.2.53	SCTP	278	COOKIE_ECHO
105	30.894147	10.15.2.53	10.15.2.41	SCTP	50	COOKIE_ACK
112	30.894629	10.15.2.41	10.15.2.53	NGAP	130	NGSetupRequest
113	30.894652	10.15.2.53	10.15.2.41	SCTP	62	SACK (Ack=0, Arwnd=106430)
114	30.895260	10.15.2.53	10.15.2.41	NGAP	118	NGSetupResponse
115	30.895419	10.15.2.41	10.15.2.53	SCTP	62	SACK (Ack=0, Arwnd=106442)
187	39.404602	10.15.2.41	10.15.2.53	NGAP/NAS-5GS	138	InitialUEMessage, Registration request
190	39.412551	10.15.2.53	10.15.2.41	NGAP/NAS-5GS	146	DownlinkNASTransport, Authentication request
193	39.413363	10.15.2.41	10.15.2.53	NGAP/NAS-5GS	146	UplinkNASTransport, Authentication response
195	39.416401	10.15.2.53	10.15.2.41	NGAP/NAS-5GS	126	DownlinkNASTransport, Security mode command
199	39.417418	10.15.2.41	10.15.2.53	NGAP/NAS-5GS	186	UplinkNASTransport
200	39.427705	10.15.2.53	10.15.2.41	NGAP/NAS-5GS	230	InitialContextSetupRequest
203	39.428215	10.15.2.41	10.15.2.53	NGAP	98	InitialContextSetupResponse
222	39.630359	10.15.2.53	10.15.2.41	SCTP	62	SACK (Ack=4, Arwnd=106496)
223	39.630602	10.15.2.41	10.15.2.53	NGAP/NAS-5GS	238	UplinkNASTransport
224	39.631038	10.15.2.53	10.15.2.41	NGAP/NAS-5GS	142	DownlinkNASTransport
229	39.645812	10.15.2.53	10.15.2.41	NGAP/NAS-5GS	242	PDUSessionResourceSetupRequest
230	39.646127	10.15.2.41	10.15.2.53	SCTP	62	SACK (Ack=5, Arwnd=106317)
231	39.651580	10.15.2.41	10.15.2.53	NGAP	102	PDUSessionResourceSetupResponse
240	39.854381	10.15.2.53	10.15.2.41	SCTP	62	SACK (Ack=7, Arwnd=106496)
255	45.146346	10.15.2.53	10.15.2.41	SCTP	98	HEARTBEAT
256	45.146561	10.15.2.41	10.15.2.53	SCTP	98	HEARTBEAT_ACK
292	46.312382	10.15.2.41	10.15.2.53	NGAP/NAS-5GS	138	InitialUEMessage, Registration request
295	46.317060	10.15.2.53	10.15.2.41	NGAP/NAS-5GS	146	DownlinkNASTransport, Authentication request
298	46.317721	10.15.2.41	10.15.2.53	NGAP/NAS-5GS	146	UplinkNASTransport, Authentication response
299	46.321087	10.15.2.53	10.15.2.41	NGAP/NAS-5GS	126	DownlinkNASTransport, Security mode command
304	46.321806	10.15.2.41	10.15.2.53	NGAP/NAS-5GS	186	UplinkNASTransport
305	46.329116	10.15.2.53	10.15.2.41	NGAP/NAS-5GS	230	InitialContextSetupRequest
310	46.329523	10.15.2.41	10.15.2.53	NGAP	98	InitialContextSetupResponse

Figure 4-23 Capture on ens3 interfaces of VM1 and VM2

No.	Time	Source	Destination	Protocol	Length	Info
129	22.099421	127.0.0.4	127.0.0.7	ICMP	101	Destination unreachable (Port unreachable)
155	28.097630	127.0.0.7	127.0.0.4	PFCP	73	PFCP Association Setup Request
156	28.097647	127.0.0.4	127.0.0.7	ICMP	101	Destination unreachable (Port unreachable)
255	30.597875	127.0.0.7	127.0.0.4	PFCP	73	PFCP Association Setup Request
256	30.597892	127.0.0.4	127.0.0.7	ICMP	101	Destination unreachable (Port unreachable)
277	33.100514	127.0.0.7	127.0.0.4	PFCP	73	PFCP Association Setup Request
278	33.100530	127.0.0.4	127.0.0.7	ICMP	101	Destination unreachable (Port unreachable)
287	34.636572	127.0.0.4	127.0.0.7	PFCP	72	PFCP Association Setup Request
288	34.636710	127.0.0.7	127.0.0.4	PFCP	78	PFCP Association Setup Response
450	45.640665	127.0.0.7	127.0.0.4	PFCP	58	PFCP Heartbeat Request
451	45.640847	127.0.0.4	127.0.0.7	PFCP	58	PFCP Heartbeat Response
452	45.640929	127.0.0.4	127.0.0.7	PFCP	58	PFCP Heartbeat Request
453	45.640974	127.0.0.7	127.0.0.4	PFCP	58	PFCP Heartbeat Response

Figure 4-24 N4 Node Level Messages – PFCP Association³⁶

4.3.3. PDU Session and Data Connection

PFCP Session Establishment

PCFP Session related messages carry information for establishing a PDU Layer between the UE and the UPF. Moreover, the SMF configures the UPF for data flows by setting the rules for each interface. The PFCP Session Establishment messages do not involve all necessary UPF setup and packet processing information. This is because the PFCP Session Establishment Request message is sent before the SMF receives AN related information from the AMF.

³⁵ The same IP addresses and UDP ports are used for the Session Related control messages.

³⁶ The Upfd association attempts fail before the Smfd starts.

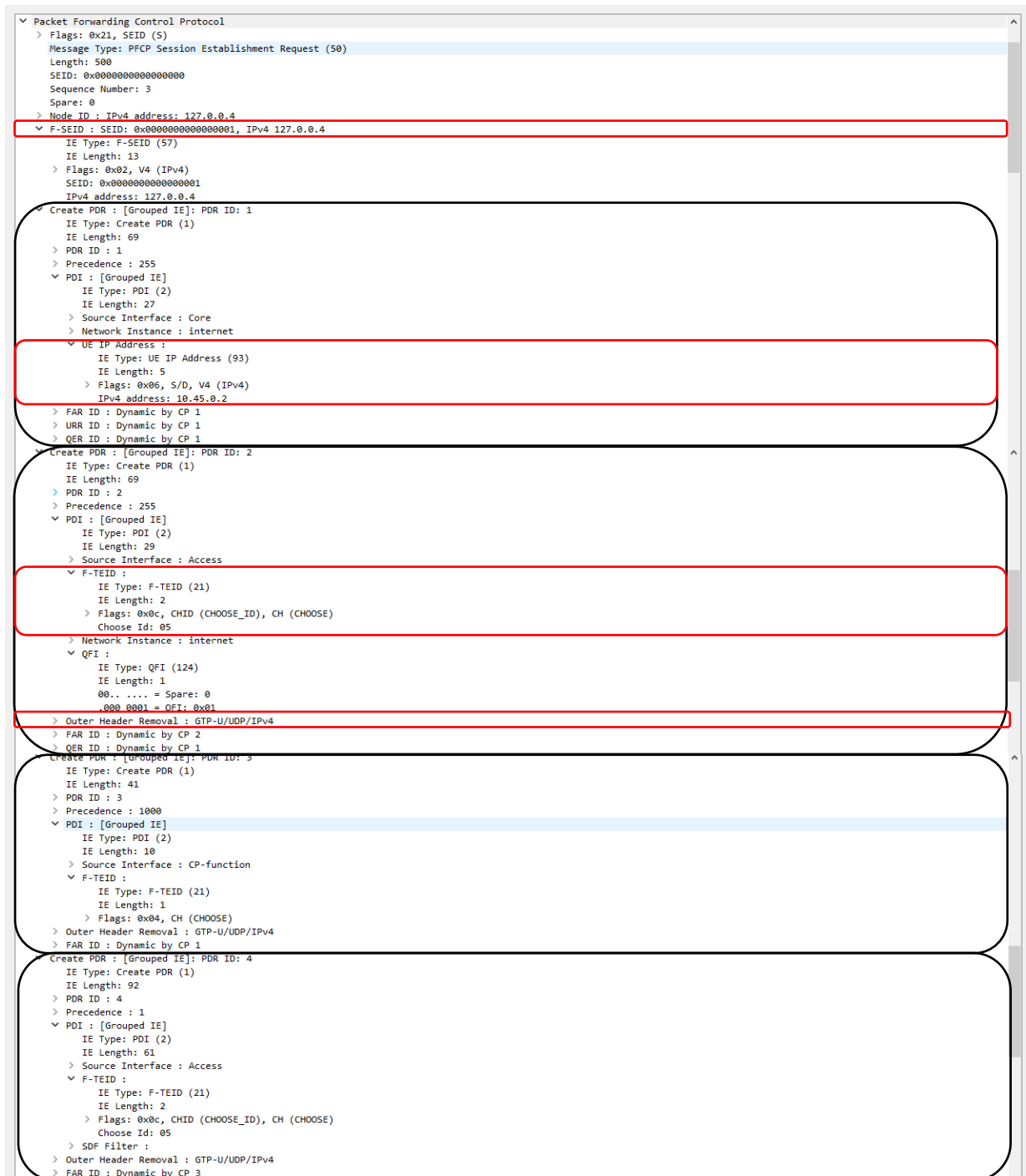


Figure 4-25 PFCP Session Establishment Request (PDRs)

Session Establishment Request includes the following.

- SEID: The value in the SMF for this session. The SMF should indicate the value to the UPF for the response messages. SEID value of the SMF may be different than UPF for the same session.
- PDR 1 (FAR 1, URR 1, and QER 1) includes rules to apply to the Core (DL) Interface of the UPF and the destination IP for DL packets

- PDR 2 (FAR 2 and QER 1) includes rules to apply to the Access (UL) Interface of the UPF asks the UPF to assign TEID for UL packets. In addition, it assigns QFI for UL packets.
- PDR 3 (FAR 1³⁷) is to apply to the CP Function (GTP – U between the SMF) Interface of the UPF.
- PDR 4³⁸ (FAR 3) is to apply to the Access (UL) Interface of the UPF.
- By FAR 1, the SMF requests the UPF to notify and buffer data flows on Core and CP.
- By FAR 2 and FAR 3, the SMF sets the forwarding parameters for Access and CP.
- PDN type implies the PDN Layer is based on IPv4.

```

Create FAR : [Grouped IE]: FAR ID: Dynamic by CP 1
IE Type: Create FAR (3)
IE Length: 18
> FAR ID : Dynamic by CP 1
  Apply Action :
  IE Type: Apply Action (44)
  IE Length: 1
  0... .. = DFRT (Duplicate for Redundant Transmission): False
  .0... .. = IPMD (IP Multicast Deny): False
  ..0... .. = IPMA (IP Multicast Accept): False
  ...0... .. = DUPL (Duplicate): False
  ....1... = NOCP (Notify the CP function): True
  ......1.. = BUFF (Buffer): True
  .....0.. = FORW (Forward): False
  .....0.. = DROP (Drop): False
  > BAR ID : 1
Create FAR : [Grouped IE]: FAR ID: Dynamic by CP 2
IE Type: Create FAR (3)
IE Length: 22
> FAR ID : Dynamic by CP 2
  Apply Action :
  IE Type: Apply Action (44)
  IE Length: 1
  0... .. = DFRT (Duplicate for Redundant Transmission): False
  .0... .. = IPMD (IP Multicast Deny): False
  ..0... .. = IPMA (IP Multicast Accept): False
  ...0... .. = DUPL (Duplicate): False
  ....0... = NOCP (Notify the CP function): False
  ......0.. = BUFF (Buffer): False
  .....0.. = FORW (Forward): True
  .....0.. = DROP (Drop): False
  Forwarding Parameters : [Grouped IE]
  IE Type: Forwarding Parameters (4)
  IE Length: 5
  Destination Interface : Core
  IE Type: Destination Interface (42)
  0000 .... = Spare: 0
  ....0001 = Interface: Core (1)
Create FAR : [Grouped IE]: FAR ID: Dynamic by CP 3
IE Type: Create FAR (3)
IE Length: 36
> FAR ID : Dynamic by CP 3
  Apply Action :
  IE Type: Apply Action (44)
  IE Length: 1
  0... .. = DFRT (Duplicate for Redundant Transmission): False
  .0... .. = IPMD (IP Multicast Deny): False
  ..0... .. = IPMA (IP Multicast Accept): False
  ...0... .. = DUPL (Duplicate): False
  ....0... = NOCP (Notify the CP function): False
  ......0.. = BUFF (Buffer): False
  .....1.. = FORW (Forward): True
  .....0.. = DROP (Drop): False
  Forwarding Parameters : [Grouped IE]
  IE Type: Forwarding Parameters (4)
  IE Length: 19
  Destination Interface : CP- Function
  IE Type: Destination Interface (42)
  IE Length: 1
  0000 .... = Spare: 0
  ....0011 = Interface: CP- Function (3)
  Outer Header Creation :
  IE Type: Outer Header Creation (84)
  IE Length: 10
  Outer Header Creation Description: GTP-U/UDP/IPv4 (256)
  TEID: 0x00000001
  IPv4 Address: 127.0.0.4
> Create URR : [Grouped IE]: URR ID: Dynamic by CP 1
> Create QER : [Grouped IE]: QER ID: Dynamic by CP 1
> Create BAR : [Grouped IE]: BAR ID: 1
> PDN Type : IPv4
[Response In: 744]

```

Figure 4-26 PCF Session Establishment Request (FARs)

³⁷ Multiple PDRs may map to the same FAR as explained in Section 2.4.2.

³⁸ Multiple PDRs/FARs may be assigned to the same interface as explained in Section 2.4.2.



Figure 4-27 PFCP Session Establishment Response

The UPF assigns TEIDs for the user plane and the SMF connection in the message. It should be noted that the UPF sets SEID as 0x06 for this session.

N4 Session ID - SEID		0x01			
PDR ID – Rule ID		1	2	3	4
Outer Header Removal		X	GTP-U/UDP/IPv4	GTP-U/UDP/IPv4	GTP-U/UDP/IPv4
Packet detection Information	Source interface	Core (DL)	Access (UL)	CP Function	Access (UL)
	CN Tunnel Info	X	TEID: 0x02, Ipv4 Add: 10.15.2.53	TEID: 0x017, Ipv4 Add: 10.15.2.53	TEID: 0x02, Ipv4 Add: 10.15.2.53
	UE IP address	10.45.0.2	X	X	X
FAR ID		1	2	1	3
List of URR ID(s)		1	x	x	x
List of QER ID(s)		1	1	x	x

Table 4-3 PDR IEs (After PFCP Session Establishment Request Message)

N4 Session ID - SEID		0x01		
Rule ID - FAR ID		1	2	3
Action		NOCP, BUFF	FORW	FORW
Forwarding Parameters	Destination Int	X	Core	CP – Function
	Outer Header Creation	X	X	Desc: GTP-U/UDP/IPv4, TEID: 0x08, Ipv4 Add: 127.0.0.4

Table 4-4 FAR(s) IEs (After PFCP Session Establishment Request Message)

PCFP Session Modification

Table 4-5 summarizes the actions to be applied to data flows. The UPF is not configured yet for forwarding actions for the DL packets. In the Session Modification Request, the SMF updates FAR 1.

```

Message Type: PCFP Session Modification Request (52)
Length: 52
SEID: 0x0000000000000006
Sequence Number: 4
Spare: 0
Update FAR : [Grouped IE]: FAR ID: Dynamic by CP 1
  IE Type: Update FAR (10)
  IE Length: 36
  FAR ID : Dynamic by CP 1
  Apply Action :
    IE Type: Apply Action (44)
    IE Length: 1
    0... .. = DFRT (Duplicate for Redundant Transmission): False
    .0... .. = IPMD (IP Multicast Deny): False
    ..0... .. = IPMA (IP Multicast Accept): False
    ...0... .. = DUPL (Duplicate): False
    ....0... .. = NOCP (Notify the CP function): False
    ..0... .. = BUFE (Buffer): False
    ....1... .. = FORW (Forward): True
    ....0... .. = DROP (Drop): False
  Update Forwarding Parameters : [Grouped IE]
    IE Type: Update Forwarding Parameters (11)
    IE Length: 19
    Destination Interface : Access
      IE Type: Destination Interface (42)
      IE Length: 1
      0000 .... = Spare: 0
      ....0000 = Interface: Access (0)
    Outer Header Creation :
      IE Type: Outer Header Creation (84)
      IE Length: 10
      Outer Header Creation Description: GTP-U/UDP/IPv4 (256)
      TEID: 0x00000001
      IPv4 Address: 10.15.2.41
[Response In: 774]
  
```

Figure 4-28 PCFP Session Modification Request

```

Packet Forwarding Control Protocol
  Flags: 0x21, SEID (5)
  Message Type: PCFP Session Modification Response (53)
  Length: 17
  SEID: 0x0000000000000001
  Sequence Number: 4
  Spare: 0
  Cause : Request accepted(success)
[Response To: 773]
[Response Time: 0.000050000 seconds]
  
```

Figure 4-29 PCFP Session Modification Response

N4 Session ID - SEID		0x01		
Rule ID - FAR ID		1	2	3
Action		NOCP, BUFE, FORW	FORW	FORW
Forwarding Parameters	Destination Int	Access	Core	CP – Function
	Outer Header Creation	Desc: GTP-U/UDP/IPv4, TEID: 0x01, Ipv4 Add: 10.15.2.41	X	Desc: GTP-U/UDP/IPv4, TEID: 0x08, Ipv4 Add: 127.0.0.4

Table 4-5 Updated FAR(s) IEs

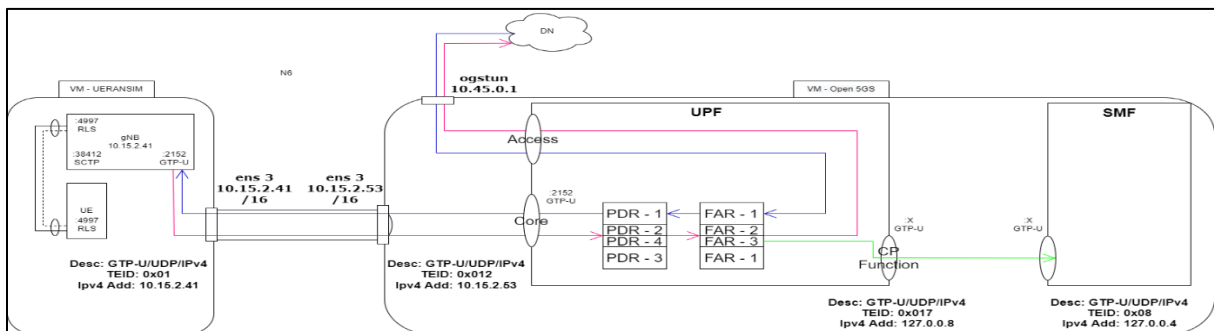


Figure 4-30 Packet Processing in the UPF

Connectivity to the DN

A ping session is started from the UE to control the user plane connection.

```
ubuntu@onur-access:~$ ping -I uesimtun0 www.google.com
PING www.google.com (142.250.184.68) from 10.45.0.2 uesimtun0: 56(84) bytes of data:
64 bytes from mil141s03-in-f4.1e100.net (142.250.184.68): icmp_seq=1 ttl=111 time=8.95 ms
64 bytes from mil141s03-in-f4.1e100.net (142.250.184.68): icmp_seq=2 ttl=111 time=8.98 ms
64 bytes from mil141s03-in-f4.1e100.net (142.250.184.68): icmp_seq=3 ttl=111 time=8.17 ms
64 bytes from mil141s03-in-f4.1e100.net (142.250.184.68): icmp_seq=4 ttl=111 time=8.87 ms
```

Figure 4-31 Ping Session

The overlay and underlay network concept can be seen in Figures 4-31 and 4-32.

```
> Frame 19: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface sshdump.exe, id 0
> Ethernet II, Src: fa:16:3e:e2:8f:3b (fa:16:3e:e2:8f:3b), Dst: fa:16:3e:33:af:34 (fa:16:3e:33:af:34)
> Internet Protocol Version 4, Src: 10.15.2.41, Dst: 10.15.2.53
> User Datagram Protocol, Src Port: 2152, Dst Port: 2152
< GPRS Tunneling Protocol
  < Flags: 0x34
  < Message Type: T-PDU (0xff)
  < Length: 92
  < TEID: 0x00000002 (2)
  < Next extension header type: PDU Session container (0x85)
  < Extension header (PDU Session container)
  < Extension Header Length: 1
  < PDU Session Container
  < Next extension header type: No more extension headers (0x00)
< Internet Protocol Version 4, Src: 10.45.0.2, Dst: 216.58.206.68
0100 .... = Version: 4
... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 84
Identification: 0x45c0 (17856)
> Flags: 0x40, Don't fragment
... 0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 64
Protocol: ICMP (1)
Header Checksum: 0x443b [validation disabled]
[Header checksum status: Unverified]
Source Address: 10.45.0.2
Destination Address: 216.58.206.68
> Internet Control Message Protocol
```

Figure 4-32 Echo-Request – Overlay and Underlay Network

```
> Frame 20: 142 bytes on wire (1136 bits), 142 bytes captured (1136 bits) on interface sshdump.exe, id 0
> Ethernet II, Src: fa:16:3e:33:af:34 (fa:16:3e:33:af:34), Dst: fa:16:3e:e2:8f:3b (fa:16:3e:e2:8f:3b)
> Internet Protocol Version 4, Src: 10.15.2.53, Dst: 10.15.2.41
> User Datagram Protocol, Src Port: 2152, Dst Port: 2152
< GPRS Tunneling Protocol
  < Flags: 0x34
  < Message Type: T-PDU (0xff)
  < Length: 92
  < TEID: 0x00000001 (1)
  < Next extension header type: PDU Session container (0x85)
  < Extension header (PDU Session container)
  < Extension Header Length: 1
  < PDU Session Container
  < Next extension header type: No more extension headers (0x00)
< Internet Protocol Version 4, Src: 216.58.206.68, Dst: 10.45.0.2
0100 .... = Version: 4
... 0101 = Header Length: 20 bytes (5)
> Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 84
Identification: 0x0000 (0)
> Flags: 0x00
... 0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 110
Protocol: ICMP (1)
Header Checksum: 0x9bfb [validation disabled]
[Header checksum status: Unverified]
Source Address: 216.58.206.68
Destination Address: 10.45.0.2
> Internet Control Message Protocol
```

Figure 4-33 Echo Reply – Overlay and Underlay Network

5 Implementation and Proof of Concept – Redundant Transmission Support on N3 Interface Technique in 5G CN

5.1. Motivation

In the previous Chapter, 5GS Registration and PDU Session Establishment procedures are implemented with software tools for 5GS deployed on virtual machines. The software modules of the Opens5GS that simulate the basic functionalities of NFs in the 5G CN, and the software modules of UERANSIM that simulate the basic functionalities of the UE and the gNB in the 5G AN, have already been configured to implement such procedures. The functionalities of those virtualization tools do not provide all the services introduced for 5GS in 3GPP's specifications. Unfortunately, redundancy support for URLLC services is among the functions not supported by these tools. Whereas the thesis aims to implement one of these redundancy support techniques, observe and analyze their outcomes, and test end-to-end connectivity to whether an improvement in terms of reliability is achieved for the user plane path.

As explained in Chapter 4, in 3GPP's Release 16, Redundant Transmission Support on N3 interface refers to the case where two GTP – U tunnels set-up between the gNB and the UPF (GTP – U Tunnel endpoints) within a PDU Session to improve the reliability in the 5G CN. Moreover, the technique requires UL and DL packets carried between the UE and the DN to be duplicated and eliminated at the GTP – U endpoints. Due to the limitations of the software tools, a workaround is proposed to implement Redundant Transmission Support on the N3 interface technique. The workaround scenario involves two UE served with separate PDU sessions within the 5G same cell. There is one GTP – U tunnel for each UE, and the tunnel endpoints are the same for each GTP – U tunnel, the gNB, and the UPF. In addition, two different PDU Sessions end in the same DN. The following figure depicts such a scenario.

The presented model in the 3GPP's Release 16 and the implementation differ in the number of PDU Sessions. Two PDU Session includes the redundancy support covering CN and AN. However, UERANSIM does not affect 5G AN in terms of reliability. Packet loss probability of the user plane transport path between the gNB and the UE(s) equals zero by default. And this is not modified for the implementation; in fact, UERANSIM has no such functionality. To explain the concept better, since the one PDU session and two PDU sessions both refer to the case where packet loss probability is zero at AN, the varying number of the PDU session does not affect the reliability of the user plane path. On the other hand, the redundant GTP – U tunnel may improve the reliability of the user plane path at the CN in case

of imperfection³⁹ on the N3 interface. A possible improvement can be foreseen because imperfection on the N3 interface applies to the GTP – U Tunnels discretely.

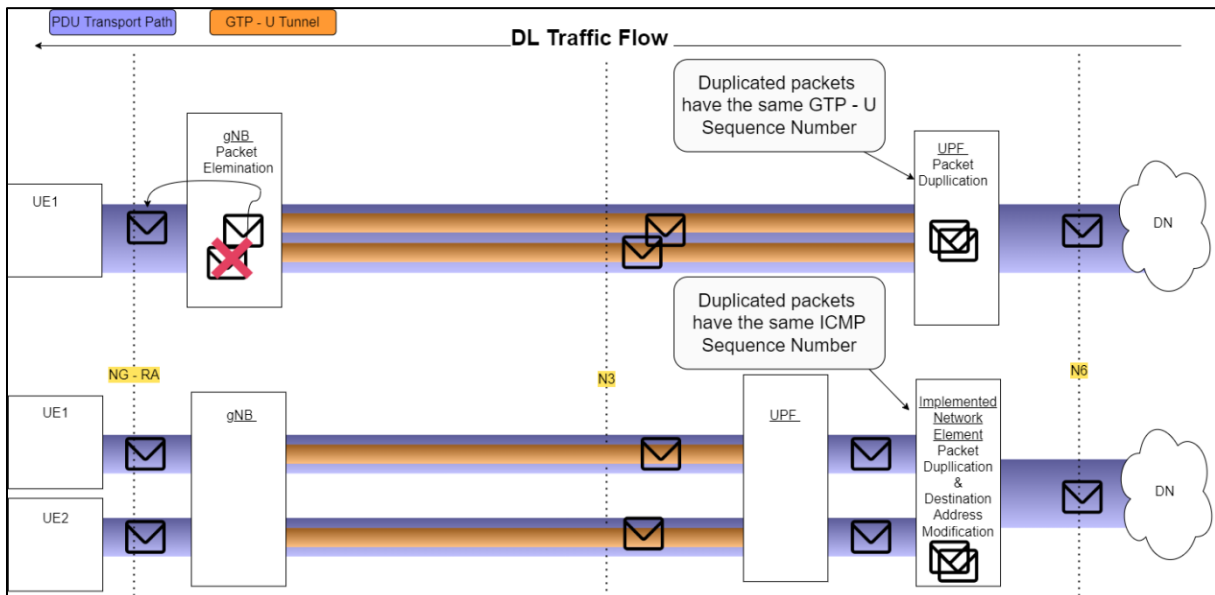


Figure 5-1 The presented model in the 3GPP's Release 16 and the implementation for Redundant Transmission Support on the N3 interface technique

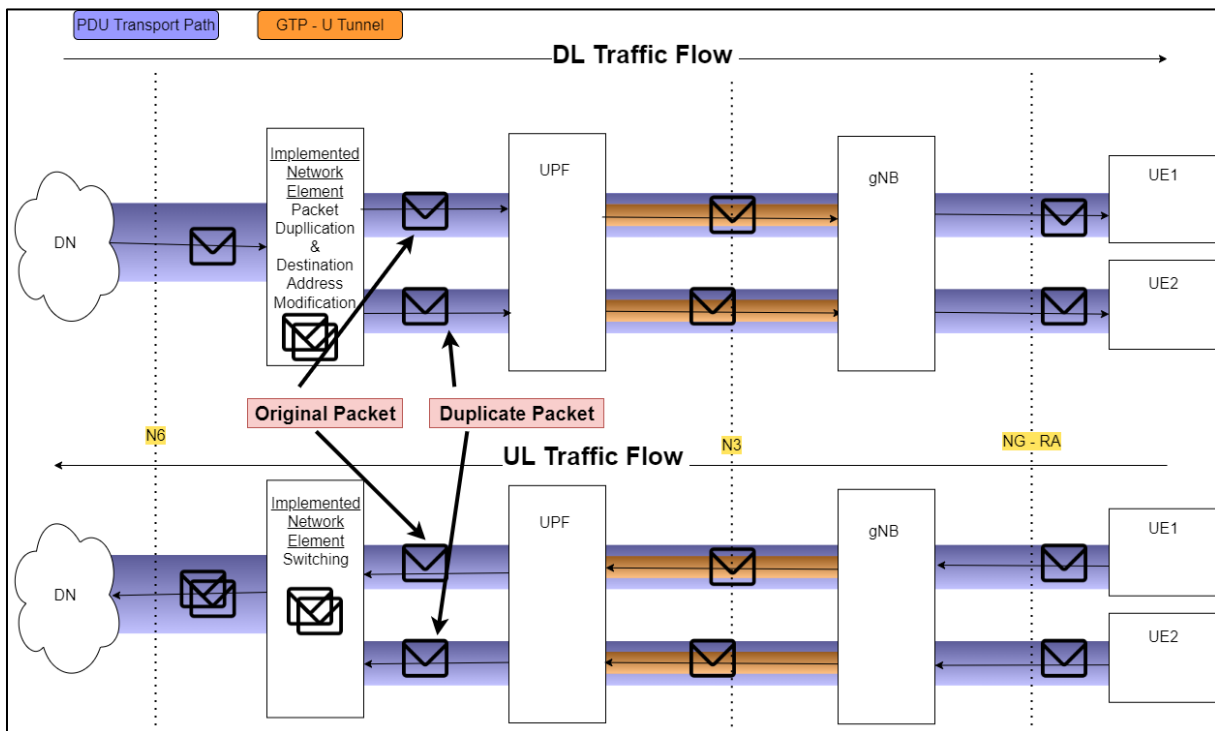


Figure 5-2 DL and UL Traffic Flow in the Implementation

A ping session from the perspective of DN is depicted in Figure 5 - 2; the operation can be summarized in the steps below.

³⁹ Imperfection refers to packet loss.

For the DL traffic flow (from DN to UE(s)),

1. DN sends one packet (ICMP ECHO request) towards one UE (UE1).
2. In Implemented Network instance, first, the received packet is duplicated. Secondly, the duplicate packet's destination address is modified to distinguish the transport path from the original one. Two identical packets, the original and duplicate packets with the same sequence number and destination address, are obtained as an operation outcome.
3. The UPF performs Destination IP Address and TEID mapping for each packet to determine the tunnel belongs to PDU Session. The UPF forwards the original and duplicate packets over two distinct user plane transport paths (over two distinct GTP – U Tunnel).
4. UE1 and UE2 receive the original and duplicate ICMP ECHO request with the same sequence number, respectively.

For the UL traffic flow (from UE(s) to DN),

1. Each UE responds to the ICMP ECHO request with an ICMP ECHO response packet.
2. The gNB performs Destination IP Address and TEID mapping for each packet.
3. Those packets are forwarded to the Implemented Network Instance. The gNB forwards the ICMP ECHO response packets with the same sequence number over two distinct user plane transport paths (over two distinct GTP – U Tunnel).
4. Implemented Network Instance does not affect the UL traffic flow, so it simply forwards the packets.
5. DN receives the two identical ICMP ECHO Response packets from different sources, therefore for the ping session, the redundancy support in the CN is satisfied for the DN.

When imperfection is a matter of fact on the N3 interface, the DN may not receive one of the duplicate packets or both. The imperfection can be applied to the N3 interface with a simple command on the VM2. When packet loss probability increases on the N3 interface, each GTP – U Tunnel suffers from packet losses separately. However, packet loss event occurs randomly, and packet loss on separate GTP – U packets are discrete events. In the case of redundant support in the CN, the overall system reliability improves. In other words, which packet will not survive on the transport path is undetermined. The sequence number of the lost packets possibly varies; therefore, the DN possibly receives one of the duplicate packets respecting the packet loss probability on each path. Hence, the overall data flow will not be affected due to the redundant support in this case (or affected lesser). The system's overall reliability can be evaluated as follows.

$$\text{System Reliability without redundancy} = 1 - P_{Tunnel}^{Loss}$$

$$\text{System Reliability with redundancy} = 1 - P_{Tunnel1}^{Loss} \times P_{Tunnel2}^{Loss}$$

where P_{Tunnel}^{Loss} is Packet Loss Probability on a GTP – U Tunnel

5.2. Implementation

The first step of the implementation is to add the UE2 to the virtualized 5GS. It can be done with the same operation performed for the UE1 in Chapter 4. Then the network block which performs the duplication for the DL packets and the destination address modification for duplicate packets needs to be designed. For such operations, a virtualized network model⁴⁰ is introduced. The following figure shows the proposed model.

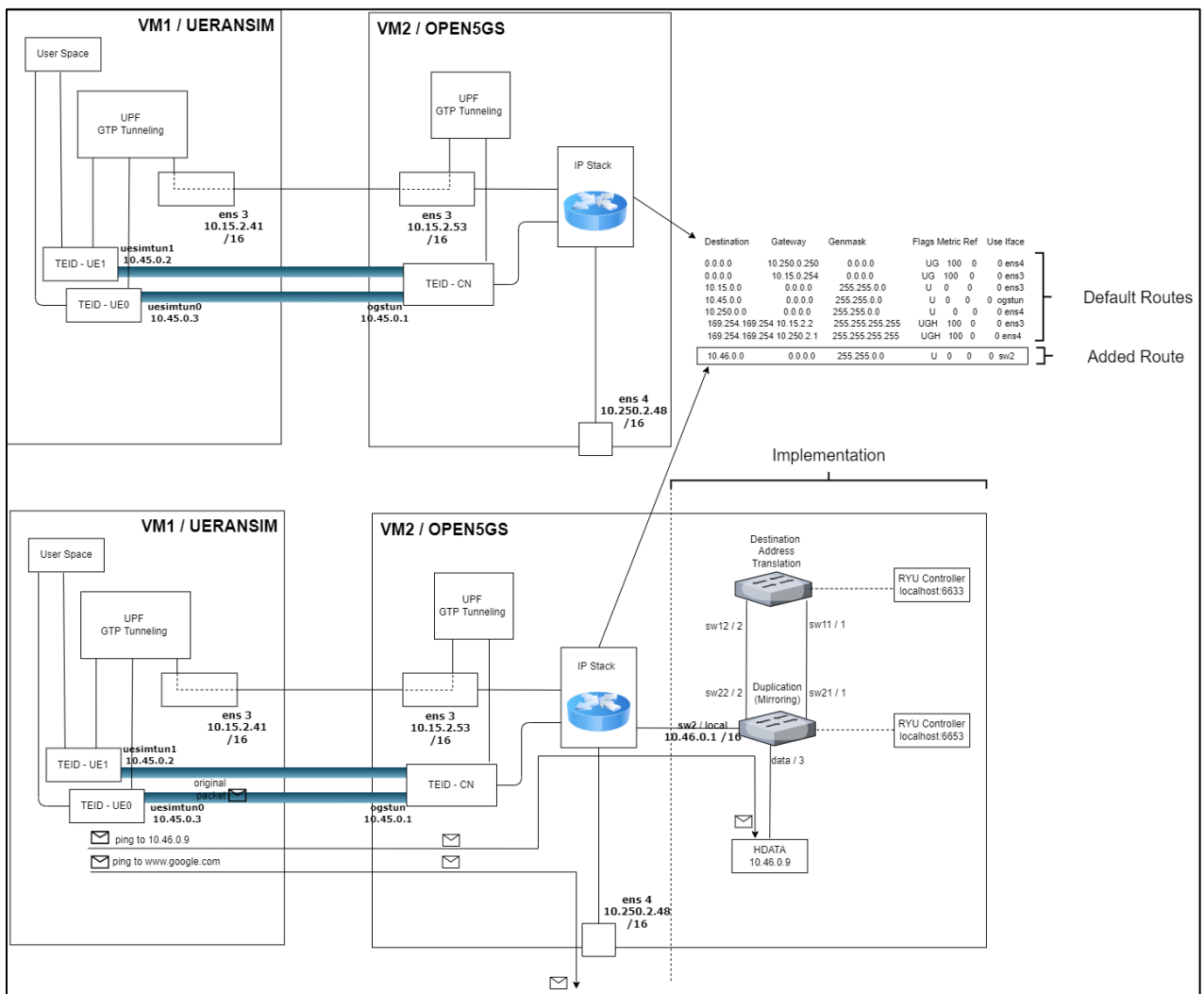


Figure 5-3 Network Model for the implementation

The implementation consists of several steps listed below.

⁴⁰ This is referred as Implemented Network Element in the previous section.

Step – 1: Two instances of OvSwitch are deployed to the VM2, where Ryu Controller instances manage the functionalities of the OvSwitch instances.

Step – 2: The Ryu controllers are deployed to the VM2. One is programmed to provide the packet duplication functionality, which applies to the DL packets. Duplicated packets are identical after the operation in Duplication Block. The other instance of the RYU controller is programmed to provide the destination address modification functionality, which applies duplicated packets. The modification is required to use different GTP – U Tunnels due to the destination IP address – TEID mapping algorithm in gNB and UPF, as explained previously.

Step – 3: A Linux namespace instance is created to model the DN (HDATA). DN implementation is required not to affect the default functionalities of the virtualized 5GS and not to affect the traffic for access and managing the VMs.

Step – 4: The interfaces of the instances are defined.

Step – 5: The route for HDATA is inserted in the routing table of HDATA.

Step – 1

Sw2 is created for the duplication (mirroring) operation, and Sw1 is created to modify the destination IP address of the incoming packets.

```

ovs-vsctl add-br sw1
ovs-vsctl set-controller sw1 tcp:127.0.0.1:6633
ovs-vsctl add-br sw2
ovs-vsctl set-controller sw2 tcp:127.0.0.1:6653

```

ovs-vsctl	add-br	sw1
The parameter indicates Linux Kernel the command applies OvS instance	Command to add OvS instance	Parameter for naming the OvS instance

Table 5-1 OpenvSwitch instance creation

ovs-vsctl	set-controller	sw1	tcp:	tcp:127.0.0.1:6633
The parameter indicates Linux Kernel the command applies OvS instance	Command to set controller for this OvS instance	To which switch that the command applies	Parameter to define connection type between the OvS instance and the controller	the IP address and port number of the Controller

Table 5-2 OpenvSwitch instance and Ryu Controller instance binding

Step – 2

Here the related part of the code that implements the required functionality for the duplication block and the destination address modification block is presented. Complete code for each block can be found in Appendix A.

Ryu Controller Instance for Packet Duplication

```
mirror_port = 1
if int(in_port) == 3:
    out_port = 4294967294
    self.logger.info("Outport %s is mirrored to port %s", out_port, mirror_port)
    actions = [parser.OFPACTIONOutput(out_port), parser.OFPACTIONOutput(mirror_port)]
else:
    actions = [parser.OFPACTIONOutput(out_port)]
```

Figure 5-4 Packet Duplication operation

This piece code implements the following for the packet duplication operation

- Set the duplication(mirroring) port as port – 1.
- For any packet coming from port – 3, set the output as a port – 4294967294 (sw2 link).
- Forward the same packet to port – 1 and port – 4294967294.

The behavior is the same as a regular switch block for any packet coming from the other ports.

Ryu Controller Instance for Destination Address Modification

```
if int(in_port) == 1 and eth.ethertype == ether_types.ETH_TYPE_IP:
    ip_pkt = pkt.get_protocol(ipv4.ipv4)
    self.logger.info("DAT")
    ether_hd = ethernet.ethernet(src = src, dst = dst, ethertype = eth.ethertype)
    ipv4_hd = ipv4.ipv4(4,5,0,0,0,2,0,ip_pkt.ttl,1,0,src = ip_pkt.src, dst = '10.45.0.3')
    icmp_pkt = pkt.get_protocol(icmp.icmp)
    icmp_type = icmp_pkt.type
    icmp_data = icmp_pkt.data
    icmp_hd = icmp.icmp(icmp_type,0,0,icmp_data)
    modified_packet = packet.Packet()
    modified_packet.add_protocol(ether_hd)
    modified_packet.add_protocol(ipv4_hd)
    modified_packet.add_protocol(icmp_hd)
    modified_packet.serialize()
    out_port = 2
    self.logger.info("Packet is modified as ip_pkt.dst %s, eth.dst %s, output %s", ipv4_hd.dst, ether_hd.dst, out_port)
    actions = [parser.OFPACTIONOutput(out_port)]
    data = modified_packet.data
else:
    actions = []
    self.logger.info("Packet is dropped")
    data = None
```

Figure 5-5 Destination IP Address Modification operation

This piece code implements the following for destination address modification

- Parse the packet to modify any Ethertype packets coming from port – 1. The parameters of the Ethernet header, the IP header, and the ICMP header of the incoming packet are obtained by parsing operation. This is required not to change the incoming properties other than the destination IP address.
- Insert the IP address for the new destination while keeping the rest unchanged.
- Create a new packet with the name modified packet.

- Add the Ethernet, Ipv4, and ICMP headers to the created one.
- Set the output port as port – 2. And send the packet.
- If the incoming packet is not from port – 1 (port – 2), do nothing except dropping it.

Step – 3

```
ip netns add hdata
```

This command creates a namespace in the Linux Kernel with the name hdata

```
ip netns exec hdata sysctl -w net.ipv6.conf.all.disable_ipv6=1
ip netns exec hdata sysctl -w net.ipv6.conf.default.disable_ipv6=1
ip netns exec hdata sysctl -w net.ipv6.conf.lo.disable_ipv6=1
```

These commands are for disabling Ipv6 packet traffic.

Step – 4

HDATA interface

```
ip link add veth0 type veth peer name data
```

This command creates a point-to-point link (a virtualized ethernet cable) where the name of one end is veth0, and the other is data.

```
ip link set veth0 netns hdata
```

This command connects the veth0 end of the link to the hdata while the other end (data) is connected to nowhere yet.

```
ip netns exec hdata ip link set veth0 up
```

The link needs to be activated.

```
ip netns exec hdata ip addr add 10.46.0.9/16 dev veth0
```

This command assigns an IP address to the veth0 interface placed in hdata.

SW1 Interfaces

```
ovs-vsctl add-port sw1 sw11
ovs-vsctl add-port sw1 sw12
```

These commands add ports to Sw1

ovs-vsctl	add-port	sw1	sw11
The parameter indicates Linux Kernel the command applies Ovs instance	Command to add a port to the Ovs instance	Parameter for naming the Ovs instance	The name of the interface

Table 5-3 Adding ports to the OpenvSwitch instance

SW2 Interfaces

```
ovs-vsctl add-port sw2 sw21
ovs-vsctl add-port sw2 sw22
```

These commands add ports to the Sw2.

```
ovs-vsctl add-port sw2 data
```

This command connects the virtual link's other end (data) to the Sw2.

```
ip addr add 10.46.0.1/16 dev sw2
```

This command assigns an IP address to SW2. This operation is required to route the traffic towards hdata (so towards sw2) in Linux Router. The route for the 10.46.0.0/16 network via sw2 port is automatically added to the routing table.

```
ip link set sw2 up
```

The link needs to be activated.

SW1 – SW2 connections

```
ip link add sw11 type veth peer name sw21
ip link add sw12 type veth peer name sw22
```

These commands create a point to point to link between the ports of the switches.

```
ip link set sw11 up
ip link set sw12 up
ip link set sw21 up
ip link set sw22 up
```

The links need to be activated.

Hdata – SW2 connection

```
ovs-vsctl add-port sw2 data
ip link set data up
ovs-vsctl add-port sw1 sw11
ovs-vsctl add-port sw1 sw12
```

Step – 5

```
ip netns exec hdata route add default gw 10.46.0.1 veth0
```

This command sets the veth0 port as a default gateway in hdata.

5.3. Proof of Concept

For the proof of concept, the steps below are followed.

VM2 – Step 1: Prove active services (only 5GS NF daemons are active)

VM2 – Step 2: Prove added route for the network 10.46.0.0/16.

VM2 – Step 3: Show the ports of the Packet Duplication and the Destination Address Modification blocks.

VM2 – Step 4: Show ip address of sw2.

VM1 – Step 1: Prove the gNB is active

VM1 – Step 2: Prove the UEs are active

VM1 – Step 3: Show the ip addresses of UEs

VM1 – Step 4: Prove with a ping session that system functions and start capturing on ogstun and uesimtun1 interfaces

VM1 – Step 5: Prove without controllers; the blocks do not function; therefore, namespace hdata (implementation of DN) is unreachable.

VM2 – Step 5: Start controllers and prove implemented functionalities.

VM2 – Step 1

```
ps aux | grep open5gs
```

This command shows the running daemons of open5gs software.

VM2 – Step 2

```
route -n
```

This command shows the routing table in VM2.

```

root@onur-vm:/home/ubuntu# ps aux | grep open5gs
root      3069  0.0  0.0 14864 1124 pts/0    S+   19:07   0:00 grep --color=auto open5gs
open5gs  12396  0.0  0.9 261848 39372 ?        Ss1  Jan18   1:37 /usr/bin/open5gs-nrfd -c /etc/open5gs/nrf.yaml
open5gs  13193  0.0  0.9 264188 38848 ?        Ss1  Jan18   0:21 /usr/bin/open5gs-udrd -c /etc/open5gs/udr.yaml
open5gs  13217  0.0  0.8 238208 35664 ?        Ss1  Jan18   0:21 /usr/bin/open5gs-udmd -c /etc/open5gs/udm.yaml
open5gs  13236  0.0  0.8 237456 35584 ?        Ss1  Jan18   0:20 /usr/bin/open5gs-nssf -c /etc/open5gs/nssf.yaml
open5gs  13255  0.0  0.8 240948 34104 ?        Ss1  Jan18   0:20 /usr/bin/open5gs-bsfd -c /etc/open5gs/bsf.yaml
open5gs  13272  0.0  0.9 238212 36628 ?        Ss1  Jan18   0:21 /usr/bin/open5gs-ausfd -c /etc/open5gs/ausf.yaml
root      13304  0.0  0.0 68308 3908 ?        S    Jan18   0:00 sudo open5gs-pcfd
root      13305  0.0  0.9 267932 38452 ?        S1   Jan18   0:22 open5gs-pcfd
open5gs  13325  0.0  1.1 456680 47232 ?        Ss1  Jan18   0:10 /usr/bin/open5gs-upfd -c /etc/open5gs/upf.yaml
open5gs  14102  0.0  0.9 268168 36760 ?        Ss1  Jan18   0:21 /usr/bin/open5gs-amfd -c /etc/open5gs/amf.yaml
open5gs  14120  0.0  1.4 1733236 59264 ?        Ss1  Jan18   1:23 /usr/bin/open5gs-smfd -c /etc/open5gs/smf.yaml

```

Figure 5-6 Running Open5GS Daemons – 5G CN NFs

```

root@onur-vm:/home/ubuntu# route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          10.250.0.250  0.0.0.0         UG    100    0      0 ens4
0.0.0.0          10.15.0.254   0.0.0.0         UG    100    0      0 ens3
10.15.0.0        0.0.0.0       255.255.0.0     U     0      0      0 ens3
10.45.0.0        0.0.0.0       255.255.0.0     U     0      0      0 ogstun
10.46.0.0        0.0.0.0       255.255.0.0     U     0      0      0 sw2
10.250.0.0       0.0.0.0       255.255.0.0     U     0      0      0 ens4
169.254.169.254 10.15.2.2     255.255.255.255 UGH   100    0      0 ens3
169.254.169.254 10.250.2.1    255.255.255.255 UGH   100    0      0 ens4

```

Figure 5-7 Routing Table of VM2

VM2 – Step 3

```
ovs-vsctl show
```

This command shows the properties of the switches.

```

root@onur-vm:/home/ubuntu# ovs-vsctl show
247cb610-1325-46b9-b297-ff07ebb9fd00
  Bridge "sw1"
    Controller "tcp:127.0.0.1:6633"
    Port "sw12"
      Interface "sw12"
    Port "sw11"
      Interface "sw11"
    Port "sw1"
      Interface "sw1"
      type: internal
  Bridge "sw2"
    Controller "tcp:127.0.0.1:6653"
    Port data
      Interface data
    Port "sw22"
      Interface "sw22"
    Port "sw2"
      Interface "sw2"
      type: internal
    Port "sw21"
      Interface "sw21"
  ovs_version: "2.9.8"

```

Figure 5-8 Designed Blocks Interfaces

VM2 – Step 4

```
ifconfig
```

```

sw2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 10.46.0.1 netmask 255.255.0.0 broadcast 0.0.0.0
  ether 5a:4a:fb:99:59:43 txqueuelen 1000 (Ethernet)
  RX packets 0 bytes 0 (0.0 B)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 0 bytes 0 (0.0 B)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figure 5-9 Sw2 IP Address

VM1 – Step 1

```
ps aux | grep gnb
```

```

ubuntu@onur-access:~/UERANSIM/config$ ps aux | grep gnb
ubuntu  1732  0.0  0.2 768448 8692 pts/23    Sl+  Jan02  15:51 build/nr-gnb -

```

Figure 5-10 Running UERANSIM Daemons – gNB

VM1 – Step 2

```
ps aux | grep ue
```

```
ubuntu@onur-access:~/UERANSIM/config$ ps aux | grep ue
root      3192  0.0  0.1 68308 4260 pts/31  S+   Jan03   0:00 sudo build/nr-
root      3193  0.1  0.1 625924 5240 pts/31  Sl+  Jan03  21:53 build/nr-ue -c
root      3225  0.0  0.1 68308 4368 pts/32  S+   Jan03   0:00 sudo build/nr-
root      3226  0.1  0.1 625924 5280 pts/32  Sl+  Jan03  21:37 build/nr-ue -c
```

Figure 5-11 Running UERANSIM Daemons – UEs

VM1 – Step 3

```
ifconfig
```

```
uesimtun0: flags=369<UP,POINTOPOINT,NOTRAILERS,RUNNING,PROMISC> mtu 1400
inet 10.45.0.2 netmask 255.255.255.255 destination 10.45.0.2
inet6 fe80::fb1e:5555:e0ec:abc0 prefixlen 64 scopeid 0x20<link>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 8 bytes 496 (496.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

uesimtun1: flags=369<UP,POINTOPOINT,NOTRAILERS,RUNNING,PROMISC> mtu 1400
inet 10.45.0.3 netmask 255.255.255.255 destination 10.45.0.3
inet6 fe80::84e5:7a22:1da7:4edd prefixlen 64 scopeid 0x20<link>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 8 bytes 496 (496.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 5-12 IP Addresses of UEs

VM1 – Step 4

```
tcpdump -i ogstun
tcpdump -i uesimtun0
ping -I uesimtun0 www.google.com
```

```
ubuntu@onur-access:~$ ping -I uesimtun0 www.google.com -c 3
PING www.google.com (142.250.180.68) from 10.45.0.2 uesimtun0: 56(84) bytes of data.
64 bytes from mi104s41-in-f4.1e100.net (142.250.180.68): icmp_seq=1 ttl=110 time=15.3 ms
64 bytes from mi104s41-in-f4.1e100.net (142.250.180.68): icmp_seq=2 ttl=110 time=15.2 ms
64 bytes from mi104s41-in-f4.1e100.net (142.250.180.68): icmp_seq=3 ttl=110 time=13.7 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 13.716/14.758/15.326/0.737 ms
ubuntu@onur-access:~$ sudo tcpdump -i uesimtun0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on uesimtun0, link-type RAW (Raw IP), capture size 262144 bytes
20:06:20.204950 IP6 onur-access > ip6-allrouters: ICMP6, router solicitation, length 8
20:06:23.050071 IP onur-access > mi104s41-in-f4.1e100.net: ICMP echo request, id 7794, seq 1, length 64
20:06:23.079367 IP mi104s41-in-f4.1e100.net > onur-access: ICMP echo reply, id 7794, seq 1, length 64
20:06:24.060950 IP onur-access > mi104s41-in-f4.1e100.net: ICMP echo request, id 7794, seq 2, length 64
20:06:24.075230 IP mi104s41-in-f4.1e100.net > onur-access: ICMP echo reply, id 7794, seq 2, length 64
20:06:25.061420 IP onur-access > mi104s41-in-f4.1e100.net: ICMP echo request, id 7794, seq 3, length 64
20:06:25.075120 IP mi104s41-in-f4.1e100.net > onur-access: ICMP echo reply, id 7794, seq 3, length 64
20:07:27.768072 IP6 onur-access > ip6-allrouters: ICMP6, router solicitation, length 8
^C
root@onur-vmi:/home/ubuntu# tcpdump -i ogstun
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ogstun, link-type RAW (Raw IP), capture size 262144 bytes
20:06:23.060621 IP 10.45.0.2 > mi104s41-in-f4.1e100.net: ICMP echo request, id 7794, seq 1, length 64
20:06:23.074930 IP mi104s41-in-f4.1e100.net > 10.45.0.2: ICMP echo reply, id 7794, seq 1, length 64
20:06:24.062382 IP 10.45.0.2 > mi104s41-in-f4.1e100.net: ICMP echo request, id 7794, seq 2, length 64
20:06:24.075862 IP mi104s41-in-f4.1e100.net > 10.45.0.2: ICMP echo reply, id 7794, seq 2, length 64
20:06:25.062451 IP 10.45.0.2 > mi104s41-in-f4.1e100.net: ICMP echo request, id 7794, seq 3, length 64
20:06:25.076048 IP mi104s41-in-f4.1e100.net > 10.45.0.2: ICMP echo reply, id 7794, seq 3, length 64
```

Figure 5-13 Captures on ogstun and uesimtun0 interfaces

VM1 – Step 5

```
ping -I uesimtun0 10.46.0.9
```

```
ubuntu@onur-access:~$ ping -I uesimtun0 10.46.0.9 -c 3
PING 10.46.0.9 (10.46.0.9) from 10.45.0.2 uesimtun0: 56(84) bytes of data.

--- 10.46.0.9 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2047ms

ubuntu@onur-access:~$
20:09:45.996073 IP onur-access > 10.46.0.9: ICMP echo request, id 7796, seq 9, length 64
^C
20 packets captured
20 packets received by filter
0 packets dropped by kernel
ubuntu@onur-access:~/UERANSIM$ sudo tcpdump -i uesimtun0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on uesimtun0, link-type RAW (Raw IP), capture size 262144 bytes
20:10:11.372926 IP onur-access > 10.46.0.9: ICMP echo request, id 7800, seq 1, length 64
20:10:12.396143 IP onur-access > 10.46.0.9: ICMP echo request, id 7800, seq 2, length 64
20:10:13.420058 IP onur-access > 10.46.0.9: ICMP echo request, id 7800, seq 3, length 64
|
^C
19 packets captured
19 packets received by filter
0 packets dropped by kernel
root@onur-vm:/home/ubuntu# tcpdump -i ogstun
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ogstun, link-type RAW (Raw IP), capture size 262144 bytes
20:10:11.375530 IP 10.45.0.2 > 10.46.0.9: ICMP echo request, id 7800, seq 1, length 64
20:10:12.399245 IP 10.45.0.2 > 10.46.0.9: ICMP echo request, id 7800, seq 2, length 64
20:10:13.422883 IP 10.45.0.2 > 10.46.0.9: ICMP echo request, id 7800, seq 3, length 64
```

Figure 5-14 Hdata is not reachable when controllers are not active

VM2 – Step 5

```
:~/ryu/ryu/app$ sudo ryu-manager --ofp-tcp-listen-port 6633 dat.py ofctl_rest.py
```

```
ubuntu@onur-vm:~/ryu/ryu/app$ sudo ryu-manager --ofp-tcp-listen-port 6633 dat.py ofctl_rest.py
loading app dat.py
loading app ofctl_rest.py
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app dat.py of SimpleSwitch13
instantiating app ofctl_rest.py of RestStatsApi
instantiating app ryu.controller.ofp_handler of OFPHandler
(13943) wsgi starting up on http://0.0.0.0:8080
```

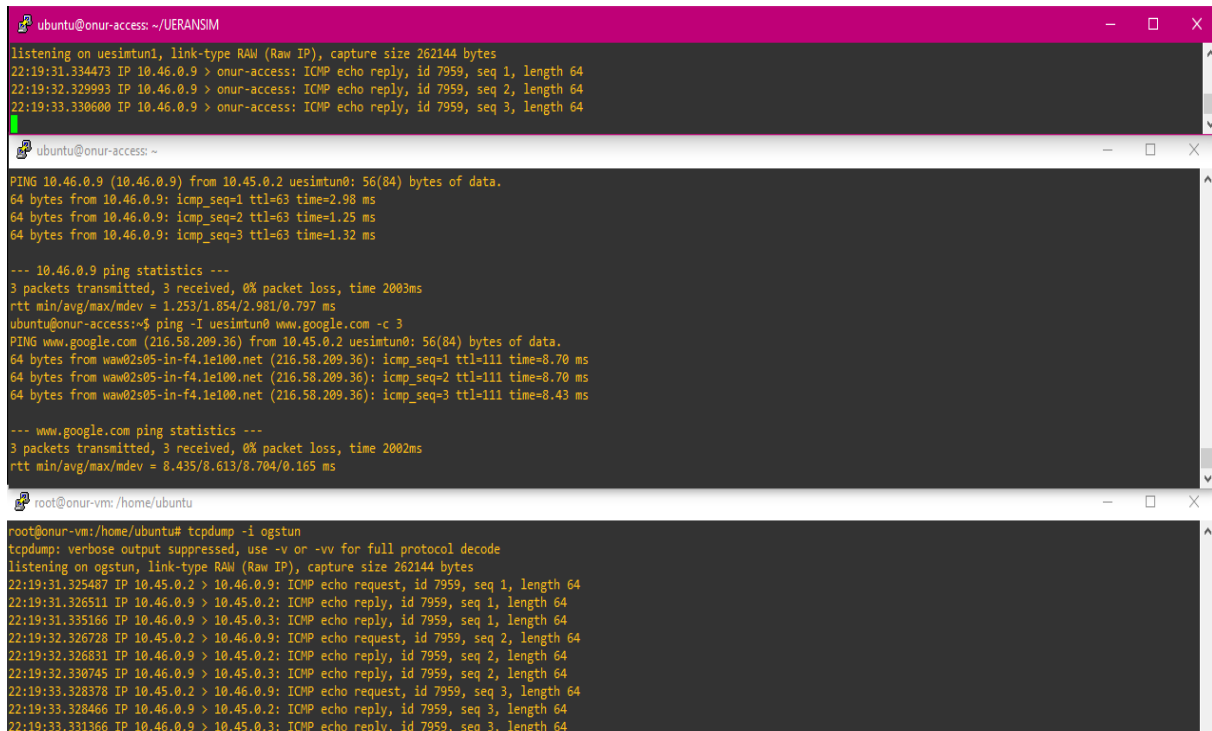
Figure 5-15 Initialization of the controller for destination address modification operation

```
:~/ryu/ryu/app$ sudo ryu-manager --ofp-tcp-listen-port 6653 mirror.py ofctl_rest.py
```

```
ubuntu@onur-vm:~/ryu/ryu/app$ sudo ryu-manager --ofp-tcp-listen-port 6653 mirror.py ofctl_rest.py
loading app mirror.py
loading app ofctl_rest.py
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app mirror.py of SimpleSwitch13
instantiating app ofctl_rest.py of RestStatsApi
instantiating app ryu.controller.ofp_handler of OFPHandler
(21238) wsgi starting up on http://0.0.0.0:8080
In port 4294967294
packet in 0178731550190919 a2:8e:2e:8f:2d:47 e6:05:18:24:eb:fb 4294967294
In port 3
packet in 0178731550190919 e6:05:18:24:eb:fb a2:8e:2e:8f:2d:47 3
Outport 4294967294 is mirrored to port 1
In port 4294967294
packet in 0178731550190919 a2:8e:2e:8f:2d:47 e6:05:18:24:eb:fb 4294967294
```

Figure 5-16 Initialization of the controller for packet duplication operation

```
ping -I uesimtun0 10.46.0.9
```



The image shows three terminal windows. The top window shows a listener on uesimtun1 capturing ICMP echo replies from 10.46.0.9. The middle window shows the execution of 'ping -I uesimtun0 10.46.0.9' and 'ping -I uesimtun0 www.google.com -c 3', displaying packet statistics and RTT values. The bottom window shows a listener on ogstun capturing the same ICMP traffic, demonstrating duplication of packets.

```
ubuntu@onur-access: ~/UERANSIM
listening on uesimtun1, link-type RAW (Raw IP), capture size 262144 bytes
22:19:31.334473 IP 10.46.0.9 > onur-access: ICMP echo reply, id 7959, seq 1, length 64
22:19:32.329993 IP 10.46.0.9 > onur-access: ICMP echo reply, id 7959, seq 2, length 64
22:19:33.330600 IP 10.46.0.9 > onur-access: ICMP echo reply, id 7959, seq 3, length 64

ubuntu@onur-access: ~
PING 10.46.0.9 (10.46.0.9) from 10.45.0.2 uesimtun0: 56(84) bytes of data.
64 bytes from 10.46.0.9: icmp_seq=1 ttl=63 time=2.98 ms
64 bytes from 10.46.0.9: icmp_seq=2 ttl=63 time=1.25 ms
64 bytes from 10.46.0.9: icmp_seq=3 ttl=63 time=1.32 ms

--- 10.46.0.9 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.253/1.854/2.981/0.797 ms
ubuntu@onur-access:~$ ping -I uesimtun0 www.google.com -c 3
PING www.google.com (216.58.209.36) from 10.45.0.2 uesimtun0: 56(84) bytes of data.
64 bytes from waw02s05-in-f4.1e100.net (216.58.209.36): icmp_seq=1 ttl=111 time=8.70 ms
64 bytes from waw02s05-in-f4.1e100.net (216.58.209.36): icmp_seq=2 ttl=111 time=8.70 ms
64 bytes from waw02s05-in-f4.1e100.net (216.58.209.36): icmp_seq=3 ttl=111 time=8.43 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 8.435/8.613/8.704/0.165 ms

root@onur-vm: /home/ubuntu
root@onur-vm: /home/ubuntu# tcpdump -i ogstun
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ogstun, link-type RAW (Raw IP), capture size 262144 bytes
22:19:31.325487 IP 10.45.0.2 > 10.46.0.9: ICMP echo request, id 7959, seq 1, length 64
22:19:31.326511 IP 10.46.0.9 > 10.45.0.2: ICMP echo reply, id 7959, seq 1, length 64
22:19:31.335166 IP 10.46.0.9 > 10.45.0.3: ICMP echo reply, id 7959, seq 1, length 64
22:19:32.326728 IP 10.45.0.2 > 10.46.0.9: ICMP echo request, id 7959, seq 2, length 64
22:19:32.326831 IP 10.46.0.9 > 10.45.0.2: ICMP echo reply, id 7959, seq 2, length 64
22:19:32.330745 IP 10.46.0.9 > 10.45.0.3: ICMP echo reply, id 7959, seq 2, length 64
22:19:33.328378 IP 10.45.0.2 > 10.46.0.9: ICMP echo request, id 7959, seq 3, length 64
22:19:33.328466 IP 10.46.0.9 > 10.45.0.2: ICMP echo reply, id 7959, seq 3, length 64
22:19:33.331366 IP 10.46.0.9 > 10.45.0.3: ICMP echo reply, id 7959, seq 3, length 64
```

Figure 5-17 Hdata is reachable when controllers are active

5.4. Testing and Measurements

In the last step of the Proof-of-Concept Section, a ping session is initialized from the UE1 to HDATA to see the functionality of the Implemented Network Instance.

- The behavior of the network for the ICMP ECHO Request packets is that there will be no effect on those packets.
- The behavior of the network for the ICMP ECHO Response packets is that each packet from the DN will be duplicated, and the destination address of the duplicate one will be changed for UE1 IP to UE2 IP.
- Both UEs receive the ICMP ECHO Response packets.
- The UL (ICMP ECHO Request) and DL (ICMP ECHO Response) packets are transported between the UE1 and DN if the ping session is not interrupted. On the other hand, UE2 only receives DL packets (ICMP ECHO Response) along with the duration of the process.

Testing and Measurements involve the following steps.

Step – 1: The ping session is initialized from the HDATA to UE1 to observe the implementation functions for the DL and UL packets.

- The behavior of the network for ICMP ECHO Request packets is that each packet from the DN will be duplicated, and the destination address of the duplicate one will be changed for UE1 IP to UE2 IP.

- The behavior of the network for the ICMP ECHO Request packets is that there will be no effect on those packets.
- Both UEs receive the ICMP ECHO Request packets with the same sequence number.
- Each UEs responds to the request with an ICMP ECHO Response message.
- The DN receives both ICMP ECHO Response packets⁴¹ with the same sequence number coming from both UEs over a separate user plane path.
- If the ping session is not interrupted, the DN receives two (original and duplicate) ICMP ECHO Response packets corresponding to the one ICMP ECHO Request message.

Step – 2: Increase the packet loss probability to %30 on the ens3 interface⁴² of VM2.

- Prove that the Open5GS data connectivity service is affected due to packet loss probability on the N3 interface
- Prove that redundant transmission support on the N3 interface improves the system reliability.

Step – 3: Repeat the test when packet loss probability equals %10.

Step – 1:

```
ip netns exec hdata ping 10.45.0.2/16
```

```

ubuntu@onur-access: ~
0 packets captured
0 packets received by filter
0 packets dropped by kernel
ubuntu@onur-access:~$ sudo tcpdump -i uesimtun0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on uesimtun0, link-type RAW (Raw IP), capture size 262144 bytes
22:31:24.690494 IP 10.46.0.9 > onur-access: ICMP echo request, id 25390, seq 1, length 64
22:31:24.690564 IP onur-access > 10.46.0.9: ICMP echo reply, id 25390, seq 1, length 64
22:31:25.691780 IP 10.46.0.9 > onur-access: ICMP echo request, id 25390, seq 2, length 64
22:31:25.691832 IP onur-access > 10.46.0.9: ICMP echo reply, id 25390, seq 2, length 64
22:31:26.693128 IP 10.46.0.9 > onur-access: ICMP echo request, id 25390, seq 3, length 64
22:31:26.693167 IP onur-access > 10.46.0.9: ICMP echo reply, id 25390, seq 3, length 64

ubuntu@onur-access: ~/UERANSIM
0 packets received by filter
0 packets dropped by kernel
ubuntu@onur-access:~/UERANSIM$ sudo tcpdump -i uesimtun1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on uesimtun1, link-type RAW (Raw IP), capture size 262144 bytes
22:31:24.694751 IP 10.46.0.9 > onur-access: ICMP echo request, id 25390, seq 1, length 64
22:31:24.694839 IP onur-access > 10.46.0.9: ICMP echo reply, id 25390, seq 1, length 64
22:31:25.695230 IP 10.46.0.9 > onur-access: ICMP echo request, id 25390, seq 2, length 64
22:31:25.695265 IP onur-access > 10.46.0.9: ICMP echo reply, id 25390, seq 2, length 64
22:31:26.696011 IP 10.46.0.9 > onur-access: ICMP echo request, id 25390, seq 3, length 64
22:31:26.696042 IP onur-access > 10.46.0.9: ICMP echo reply, id 25390, seq 3, length 64

root@onur-vm: /home/ubuntu
2:23:13.997659 IP waw02s05-in-f36.1e100.net > 10.45.0.2: ICMP echo reply, id 7962, seq 3, length 64
C
5 packets captured
5 packets received by filter
0 packets dropped by kernel
root@onur-vm:/home/ubuntu# ip netns exec hdata ping 10.45.0.2
PING 10.45.0.2 (10.45.0.2) 56(84) bytes of data:
4 bytes from 10.45.0.2: icmp_seq=1 ttl=63 time=3.88 ms
4 bytes from 10.45.0.3: icmp_seq=1 ttl=63 time=7.68 ms (DUP!)
4 bytes from 10.45.0.2: icmp_seq=2 ttl=63 time=1.76 ms
4 bytes from 10.45.0.3: icmp_seq=2 ttl=63 time=4.96 ms (DUP!)
4 bytes from 10.45.0.2: icmp_seq=3 ttl=63 time=1.76 ms
4 bytes from 10.45.0.3: icmp_seq=3 ttl=63 time=4.53 ms (DUP!)
C
-- 10.45.0.2 ping statistics --
packets transmitted, 3 received, +3 duplicates, 0% packet loss, time 2003ms

```

Figure 5-18 Ping from Hdata towards the UE – 1

⁴¹ Received duplicated packets are marked with DUP! on the log of ping session.

⁴² This interface corresponds to the N3 interface of the 5GS.

Step – 2:

```
tc qdisc add dev ens3 root netem loss 30%
ping -I uesimtun0 www.google.com -c 1000
```

```
ubuntu@onur-access:~$ ping -I uesimtun0 www.google.com -c 1000
PING www.google.com (172.217.21.68) from 10.45.0.2 uesimtun0: 56(84) bytes of data.
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=1 ttl=111 time=8.57 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=2 ttl=111 time=14.4 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=3 ttl=111 time=14.2 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=4 ttl=111 time=14.0 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=7 ttl=111 time=14.2 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=8 ttl=111 time=14.4 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=12 ttl=111 time=13.8 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=13 ttl=111 time=14.0 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=14 ttl=111 time=13.9 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=15 ttl=111 time=14.1 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=16 ttl=111 time=14.0 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=17 ttl=111 time=14.1 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=20 ttl=111 time=13.9 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=22 ttl=111 time=14.1 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=995 ttl=111 time=14.3 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=996 ttl=111 time=14.0 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=997 ttl=111 time=14.3 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=998 ttl=111 time=14.2 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=999 ttl=111 time=14.0 ms
64 bytes from fra07s31-in-f68.1e100.net (172.217.21.68): icmp_seq=1000 ttl=111 time=14.2 ms

--- www.google.com ping statistics ---
1000 packets transmitted, 675 received, 32% packet loss, time 1005931ms
rtt min/avg/max/mdev = 7.381/14.766/39.444/2.773 ms
ubuntu@onur-access:~$
```

Figure 5-19 Ping statistics when CN is lossy. The packet loss rate is slightly more than %32 as expected.

```
ip netns exec hdata ping 10.45.0.2/16 -c 1000
```

```
ubuntu@onur-access: -
22:58:28.178703 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 744, length 64
22:58:28.178733 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 744, length 64
22:58:30.182154 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 746, length 64
22:58:30.182185 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 746, length 64
22:58:31.183739 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 747, length 64
22:58:31.183770 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 747, length 64
22:58:32.185310 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 748, length 64
22:58:32.185340 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 748, length 64
22:58:33.187408 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 749, length 64
22:58:33.187440 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 749, length 64
22:58:35.190864 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 751, length 64
22:58:35.190893 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 751, length 64

ubuntu@onur-access: ~/UERANSIM
22:58:29.184825 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 745, length 64
22:58:30.184796 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 746, length 64
22:58:30.184824 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 746, length 64
22:58:31.186363 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 747, length 64
22:58:31.186393 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 747, length 64
22:58:32.188362 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 748, length 64
22:58:32.188391 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 748, length 64
22:58:34.191792 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 750, length 64
22:58:34.191848 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 750, length 64
22:58:35.194179 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 751, length 64
22:58:35.194205 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 751, length 64

64 bytes from 10.45.0.2: icmp_seq=740 ttl=63 time=1.66 ms
64 bytes from 10.45.0.3: icmp_seq=741 ttl=63 time=4.71 ms
64 bytes from 10.45.0.2: icmp_seq=742 ttl=63 time=3.46 ms
64 bytes from 10.45.0.3: icmp_seq=743 ttl=63 time=4.64 ms
64 bytes from 10.45.0.2: icmp_seq=744 ttl=63 time=1.64 ms
64 bytes from 10.45.0.3: icmp_seq=745 ttl=63 time=5.22 ms
64 bytes from 10.45.0.2: icmp_seq=746 ttl=63 time=1.84 ms
64 bytes from 10.45.0.3: icmp_seq=746 ttl=63 time=4.50 ms (DUPL!)
64 bytes from 10.45.0.2: icmp_seq=747 ttl=63 time=1.85 ms
64 bytes from 10.45.0.3: icmp_seq=747 ttl=63 time=4.39 ms (DUPL!)
64 bytes from 10.45.0.2: icmp_seq=748 ttl=63 time=3.14 ms
64 bytes from 10.45.0.3: icmp_seq=748 ttl=63 time=4.88 ms (DUPL!)
64 bytes from 10.45.0.2: icmp_seq=749 ttl=63 time=2.42 ms
64 bytes from 10.45.0.3: icmp_seq=750 ttl=63 time=4.90 ms
64 bytes from 10.45.0.2: icmp_seq=751 ttl=63 time=1.51 ms
64 bytes from 10.45.0.3: icmp_seq=751 ttl=63 time=5.03 ms (DUPL!)
```

Figure 5-20 Packets with sequence number 748 are delivered by both UP paths, so hdata receives both (DUP!). However, packets sequence numbers 749 and 750 are delivered by only one of the paths.


```

23:02:37.045259 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 992, length 64
23:02:37.045297 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 992, length 64
23:02:38.046305 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 993, length 64
23:02:38.046336 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 993, length 64
23:02:39.047672 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 994, length 64
23:02:39.047702 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 994, length 64
23:02:40.049244 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 995, length 64
23:02:40.049270 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 995, length 64
23:02:41.050729 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 996, length 64
23:02:41.050763 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 996, length 64
23:02:44.079617 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 999, length 64
23:02:44.079653 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 999, length 64

23:02:43.080330 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 998, length 64
23:02:43.080412 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 998, length 64
23:02:44.082697 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 999, length 64
23:02:44.082724 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 999, length 64
23:02:45.083256 IP 10.46.0.9 > onur-access: ICMP echo request, id 3835, seq 1000, length 64
23:02:45.083322 IP onur-access > 10.46.0.9: ICMP echo reply, id 3835, seq 1000, length 64
23:08:32.428035 IP6 onur-access > ip6-allrouters: ICMP6, router solicitation, length 8

64 bytes from 10.45.0.2: icmp_seq=996 ttl=63 time=1.45 ms
64 bytes from 10.45.0.3: icmp_seq=996 ttl=63 time=3.99 ms (DUP!)
64 bytes from 10.45.0.3: icmp_seq=998 ttl=63 time=4.78 ms
64 bytes from 10.45.0.2: icmp_seq=999 ttl=63 time=1.79 ms
64 bytes from 10.45.0.3: icmp_seq=999 ttl=63 time=4.70 ms (DUP!)
64 bytes from 10.45.0.3: icmp_seq=1000 ttl=63 time=4.49 ms

--- 10.45.0.2 ping statistics ---
1000 packets transmitted, 908 received, +491 duplicates, 9% packet loss, time 1001836ms
rtt min/avg/max/mdev = 0.949/3.124/12.813/1.623 ms
root@onur-vm:/home/ubuntu#

```

Figure 5-21 Ping statistics when deployed redundant transmission support on the N3 interface. The improvement in packet loss rate can be seen clearly. Packets with sequence number 997 are lost in both UP paths.

Step – 3:

```

tc qdisc del dev ens3 root netem loss 30%
tc qdisc add dev ens3 root netem loss 10%
ping -I uesimtun0 www.google.com -c 1000
ip netns exec hdata ping 10.45.0.2/16 -c 1000

```

```

64 bytes from mil04s44-in-f4.1e100.net (142.250.180.164): icmp_seq=997 ttl=110 time=7.84 ms
64 bytes from mil04s44-in-f4.1e100.net (142.250.180.164): icmp_seq=998 ttl=110 time=8.08 ms
64 bytes from mil04s44-in-f4.1e100.net (142.250.180.164): icmp_seq=999 ttl=110 time=7.88 ms
64 bytes from mil04s44-in-f4.1e100.net (142.250.180.164): icmp_seq=1000 ttl=110 time=7.96 ms

--- www.google.com ping statistics ---
1000 packets transmitted, 897 received, 10% packet loss, time 1002390ms
rtt min/avg/max/mdev = 7.242/9.335/362.663/17.348 ms
ubuntu@onur-access:~$

64 bytes from 10.45.0.3: icmp_seq=998 ttl=63 time=4.28 ms (DUP!)
64 bytes from 10.45.0.2: icmp_seq=999 ttl=63 time=1.48 ms
64 bytes from 10.45.0.3: icmp_seq=999 ttl=63 time=4.53 ms (DUP!)
64 bytes from 10.45.0.2: icmp_seq=1000 ttl=63 time=1.27 ms

--- 10.45.0.2 ping statistics ---
1000 packets transmitted, 993 received, +822 duplicates, 0% packet loss, time 1000637m
rtt min/avg/max/mdev = 0.619/2.896/13.519/1.698 ms

```

Figure 5-22 Ping test is repeated while packet loss probability is %10 instead of %30.

6 Conclusion

This thesis examines redundant transmission support on the 5G CN to enhance Ultra-Reliable Low Latency Communication services. In this context, three redundancy techniques presented in 3GPP Release 16 are analyzed: Dual Connectivity, support on N3/N9 interface, and support on Transport Layer. All techniques are based on packet duplication and elimination at the user plane path. The redundancy can be provided at the level of PDU Session, GTP Tunnel, or Transport layer.

To prove the concept, an implementation of redundant transmission support on the N3 interface with networking virtualization instruments is presented in Chapter 5. As expected, the redundant path reduced the packet loss rate along the user plane path and significantly improved system reliability. In numerical terms, a user path with a %1 packet loss rate improves to %0.01 with the redundancy support, and the system reliability reaches %99.99. The obtained reliability value already satisfies such scenarios introduced in [13] as Discrete Automation, Process automation monitoring, and Electricity distribution – medium voltage. For the others where the reliability requirement is higher, the combination of introduced support techniques could be utilized. For example, redundant transmission support on the N3 interface technique where a redundant GTP Tunnel is set up can be boosted by implementing redundancy support on the transport layer on each GTP Tunnel.

On the other hand, the inherent contradiction between reliability and low latency complicates the design of URLLC services. As the replication and elimination operations, and possibly the exchange of control messages corresponding to the process, contribute to the overall packet processing time within the system. These additional operations inevitably increase the end-to-end latency in the 5GS. The use of layer 2 (e.g., MPLS) protocols for backhaul/underlying networks could be a solution to overcome the problem of increased processing time contrary to the implementation where IPv4 is used as an underlying network. Such protocols eliminate GTP/UDP/IP header encapsulating the PDUs and reduce the packet header length. [17] proves that direct Ethernet transport instead of GTP – U transport provides a %31 reduction in overhead of the small packets and a %17 in mixed traffic. Besides, for the overlay network, Ethernet-based or Unstructured PDU Sessions may provide further improvement in end-to-end delay latency.

Appendix A - Ryu Controllers Code Files

A.1 Packet Duplication Block Controller

```
# Copyright (C) 2011 Nippon Telegraph and Telephone Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# Packet Duplication Block
# by Onur Özenir
#
# This code is a modified version of the original code. The original
# code is created by Franco Callegati and Chiara Contoli for the
# lecture "LABORATORY OF NETWORK PROGRAMMABILITY AND AUTOMATION 2021".
# The code implements packet duplication for any type of incoming packets from # port 3. The original
# packet is sent from port 4294967294 (this is the port connected to the Linux Router, and duplicate
# packet is sent # to port 1. Unless the input port is not 3, the block function as a regular switch except arp
# floods to avoid cycle between this block and destination address translation block.
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import arp
from ryu.lib.packet import ether_types
from ryu.lib.packet import ipv4
from ryu.lib.packet import icmp
from ryu.ofproto import inet
from ryu.lib.packet import tcp

class SimpleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.mac_to_port = {}
        self.arp_table_gw={ }
        @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
```

```

# install table-miss flow entry
#
# We specify NO BUFFER to max_len of the output action due to
# OVS bug. At this moment, if we specify a lesser number, e.g.,
# 128, OVS will send Packet-In with invalid buffer_id and
# truncated packet data. In that case, we cannot output packets
# correctly. The bug has been fixed in OVS v2.1.0.
#self.logger.info("SWITCH FEATURES - CONFIG_DISPATCHER PHASE")
match = parser.OFPMatch()
actions =
[parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,ofproto.OFPCML_NO_BUFFER)]
self.add_flow(datapath, 0, match, actions)
#match = parser.OFPMatch(eth_dst='ff:ff:ff:ff:ff:ff', arp_op='1')
#actions = [parser.OFPActionOutput(ofproto.OFPP_FLOOD)]
#self.add_flow(datapath, 0, match, actions)
match = parser.OFPMatch(eth_type=0x0806,eth_dst='ff:ff:ff:ff:ff:ff')
actions = [parser.OFPActionOutput(ofproto.OFPP_FLOOD)]
self.add_flow(datapath, 1, match, actions)
match = parser.OFPMatch(in_port = 2 )
out_port = 4294967294
actions = [parser.OFPActionOutput(out_port)]
self.add_flow(datapath, 1, match, actions)
def add_flow(self, datapath, priority, match, actions, buffer_id=None):
ofproto = datapath.ofproto
parser = datapath.ofproto_parser

inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,actions)]
if buffer_id:
    mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                            priority=priority, match=match,
                            instructions=inst)
else:
    mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                            match=match, instructions=inst)
datapath.send_msg(mod)

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
# If you hit this you might want to increase
# the "miss_send_length" of your switch
if ev.msg.msg_len < ev.msg.total_len:
    self.logger.debug("packet truncated: only %s of %s bytes",
                    ev.msg.msg_len, ev.msg.total_len)
msg = ev.msg
datapath = msg.datapath
ofproto = datapath.ofproto
parser = datapath.ofproto_parser
in_port = msg.match['in_port']
self.logger.info("In port %d", in_port)
pkt = packet.Packet(msg.data)
eth = pkt.get_protocols(ethernet.ethernet)[0]

if eth.ethertype == ether_types.ETH_TYPE_LLDP:
    # ignore lldp packet
    return
dst = eth.dst
src = eth.src
dest= 0

```

```

mirror_port = 1
dpid = format(datapath.id, "d").zfill(16)
self.mac_to_port.setdefault(dpid, {})
self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)
# learn a mac address to avoid FLOOD next time.
self.mac_to_port[dpid][src] = in_port
if dst in self.mac_to_port[dpid]:
    out_port = self.mac_to_port[dpid][dst]
else:
    out_port = ofproto.OFPP_FLOOD
if int(in_port) == 3 :
    out_port = 4294967294
    self.logger.info("Outport %s is mirrored to port %s", out_port, mirror_port)
    actions = [parser.OFPActionOutput(out_port), parser.OFPActionOutput(mirror_port)]
else:
    actions = [parser.OFPActionOutput(out_port)]
# install a flow to avoid packet_in next time
if out_port != ofproto.OFPP_FLOOD:
    match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
    #verify if we have a valid buffer_id, if yes avoid to send both
    #flow_mod & packet_out
    if msg.buffer_id != ofproto.OFP_NO_BUFFER:
        self.add_flow(datapath, 1, match, actions, msg.buffer_id)
        return
    else:
        self.add_flow(datapath, 1, match, actions)
data = None
if msg.buffer_id == ofproto.OFP_NO_BUFFER:
    data = msg.data

out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                        in_port=in_port, actions=actions, data=data)
datapath.send_msg(out)

```

A.2 Destination Address Modification Block Controller

```
# Copyright (C) 2011 Nippon Telegraph and Telephone Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
# Destination Address Modification Block
# by Onur Özenir
#
# This code is a modified version of the original code. The original
# code is created by Franco Callegati and Chiara Contoli for the
# lecture "LABORATORY OF NETWORK PROGRAMMABILITY AND AUTOMATION 2021".
# The code implements destination address modification for incoming
# Ethernet packets where input port is 1. If the incoming packet is
# not an Ethernet packet or the input port is not 1, then the packet
# is dropped.
```

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER
from ryu.controller.handler import set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import arp
from ryu.lib.packet import ether_types
from ryu.lib.packet import ipv4
from ryu.lib.packet import icmp
from ryu.ofproto import inet
from ryu.lib.packet import tcp
```

```
class SimpleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

    @set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
    def switch_features_handler(self, ev):
        datapath = ev.msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        match = parser.OFPMatch()
        actions = [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER,
```



```

        ofproto.OFPCML_NO_BUFFER)]
self.add_flow(datapath, 0, match, actions)

def add_flow(self, datapath, priority, match, actions, buffer_id=None):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,actions)]
    if buffer_id:
        mod = parser.OFPFlowMod(datapath=datapath, buffer_id=buffer_id,
                                priority=priority, match=match,
                                instructions=inst)
    else:
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                match=match, instructions=inst)
    datapath.send_msg(mod)

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    if ev.msg.msg_len < ev.msg.total_len:
        self.logger.debug("packet truncated: only %s of %s bytes",
                          ev.msg.msg_len, ev.msg.total_len)
    msg = ev.msg
    datapath = msg.datapath
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']
    self.logger.info("In port %d", in_port)
    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    if eth.ethertype == ether_types.ETH_TYPE_LLDP:
        # ignore lldp packet
        return
    dst = eth.dst
    src = eth.src
    destination_ip = 0
    source_ip = 0
    self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)
    if int(in_port) == 1 and eth.ethertype == ether_types.ETH_TYPE_IP: #and str(src) == '10.46.0.9':
        ip_pkt = pkt.get_protocol(ipv4.ipv4)
        self.logger.info("Destination address is modified")
        ether_hd = ethernet.ethernet(src = src,dst = dst, ethertype = eth.ethertype)
        ipv4_hd = ipv4.ipv4(4,5,0,0,0,2,0,ip_pkt.ttl,1,0,src = ip_pkt.src, dst = '10.45.0.3')
        icmp_pkt = pkt.get_protocol(icmp.icmp)
        icmp_type = icmp_pkt.type
        icmp_data = icmp_pkt.data
        icmp_hd = icmp.icmp(icmp_type,0,0,icmp_data)
        modified_packet = packet.Packet()
        modified_packet.add_protocol(ether_hd)
        modified_packet.add_protocol(ipv4_hd)
        modified_packet.add_protocol(icmp_hd)
        modified_packet.serialize()
        out_port = 2
        self.logger.info("Packet is modified as ip_pkt.dst %s, eth.dst %s, output
%s",ipv4_hd.dst,ether_hd.dst,out_port)
        actions = [parser.OFPActionOutput(out_port)]
        data = modified_packet.data

```

```
else:
    actions = []
    self.logger.info("Packet is dropped")
    data = None
out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                          in_port=in_port, actions=actions, data=data)
datapath.send_msg(out)
```

Bibliography

- [1] Oumer Teyeb, Gustav Wikström, Magnus Stattin, Thomas Cheng, Sebastian Faxér, Hieu Do, "EVOLVING LTE TO FIT THE 5G FUTURE," ERICSSON, 2017. [Online]. Available: https://www.ericsson.com/4ae09e/assets/local/reports-papers/ericsson-technology-review/docs/2017/etr_evolving_lte_to_fit_the_5g_future.pdf.
- [2] A. Detti, "Functional Architecture," *5G Italy White Book: From Research to Market*, CNIT, pp. 59-68.
- [3] Nurul Huda Mahmood, Melisa Lopez, Daniela Laselva, Klaus Pedersen, Gilberto Berardinelli, "Reliability Oriented Dual Connectivity for URLLC services in 5G New Radio," in *15th International Symposium on Wireless Communication Systems (ISWCS 2018)*, Lisbon, 2018.
- [4] A. Aijaz, "Duplication in Dual Connectivity Enabled 5G Wireless Networks: Overview and Challenges," *IEEE Communications Standards Magazine*, pp. 20-28, September 2019.
- [5] 3GPP TS 23.501: "5G; System architecture for the 5G System (5GS)", V16.6.0, 2020-10.
- [6] Jaya Rao, Sophie Vrzic, "Packet Duplication for URLLC in 5G: Architectural Enhancements and Performance Analysis," in *IEEE Network*, pp. 20-28, April/March 2018.
- [7] Stefan Rommer, Peter Hedman, Magnus Olsson, Lars Frid, Shabnam Sultana, Catherine Mulligan, *5G CORE NETWORKS: Powering Digitalization*, Academic Press, 2020.
- [8] 3GPP TS 29.244: "LTE; 5G; Interface between the Control Plane and the User Plane nodes", V16.6.0, 2021-01.
- [9] 3GPP TS 23.502: "5G; Procedures for the 5G System (5GS)", V16.8.0, 2021-04.
- [10] 3GPP TS 38.415: "5G; NG-RAN; PDU Session User Plane protocol", V15.1.0, 2018-09.
- [11] 3GPP TS 29.281: "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", V17.0.0, 2021-03.

- [12] Jaya Rao, Sophie Vrzic, "Packet Duplication for URLLC in 5G Dual Connectivity Architecture," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, Ottawa, Ontario, Canada, 2018.
- [13] 3GPP TS 22.261: "5G; Service requirements for next generation new services and markets", V15.5.0 Release, 2018-07.
- [14] "UERANSIM," [Online]. Available: <https://github.com/aligungr/UERANSIM>.
- [15] "Open5GS," [Online]. Available: <https://open5gs.org/>.
- [16] "Universal TUN/TAP device driver," [Online]. Available: <https://www.kernel.org/doc/html/latest/networking/tuntap.html>.
- [17] John Kaippallimalil, H Anthony Chan, *Network Virtualization and Direct Ethernet Transport for Packet Data Network Connections in 5G Wireless*, Cesena, 2014.
- [18] "MongoDB," [Online]. Available: <https://www.mongodb.com/>.
- [19] I. M. González, "Virtualized Cellular Networks with Native Cloud Functions", Vigo: M.S. thesis, Dept. Tel. Eng., Univ. of Vigo, 2021.
- [20] C. Grasselli, "Multi-Domain Orchestration of Virtualized Mobile Core Networks", Bologna: M.S. thesis, Dept. Tel. Eng., Univ. of Bologna, 2019.
- [21] S. D. Santi, "5G Network Architecture", Cesena: M.S. thesis, Dept. Informatics, Univ. of Bologna, 2019.
- [22] "Open vSwitch Manual," [Online]. Available: <https://docs.openvswitch.org/en/latest/ref/index.html>.

List of Figures

Figure 1-1 Service Categories in 5GS [1]	2
Figure 1-2 5G URLLC Requirements [1]	3
Figure 2-1 5G Reference Point System Architecture (Simplified Version) [5]	7
Figure 2-2 5G Service-Based System Architecture (Simplified Version) [5]	8
Figure 2-3: 5GS Architecture for Non-roaming case (Simplified Version) [5]	9
Figure 2-4 NG – RAN in 5GS	10
Figure 2-5 User Plane and Control Plane Protocol Stacks in NG – RAN [4]	11
Figure 2-6 5G CN Service-Based Interfaces and services consumed by the AMF	14
Figure 2-7: SMF Services and Service Consumers	15
Figure 2-8 User Plane Path for one UPF and two UPFs cases	19
Figure 2-9 PFCP Protocol Stack [8]	20
Figure 2-10 N4 Association Setup Procedure [9]	21
Figure 2-11 N4 Association Release Procedure [9]	22
Figure 2-12 N4 Association Release Procedure [9]	22
Figure 2-13 N4 Report Procedure [9]	22
Figure 2-15 PDU Session Establishment Procedure [9]	28
Figure 2-16 PDU Session Modification Procedure [9]	29
Figure 2-17 PDU Session Release Procedure [9]	29
Figure 2-18 G – PDU Protocol Stack	31
Figure 2-19 GTP v1 – U Header [11]	31
Figure 2-20 GTP -U Tunnel and PDU Session	33
Figure 2-21 GTP – U Tunnel Setup Procedure	34
Figure 2-22 GTP – U Echo Request / Response Messages	35
Figure 2-23 Packet Processing in the UPF [7]	35
Figure 2-24 The call flow of a UE-triggered registration procedure	40
Figure 2-25 The call flow of a UE-triggered Service Request procedure	41
Figure 2-26 The call flow of a UE-triggered PDU Session Establishment Procedure	45
Figure 3-1 5G CN deployment for Dual Connectivity including Protocol Stack	49
Figure 3-2 System Procedure for Dual Connectivity	51
Figure 3-3 Redundant GTP – U Tunnel Setup Procedure	53
Figure 3-4 The call flow for Redundancy support on N3 Interface	54
Figure 3-5 Redundancy support on N3/N9 Interfaces	55
Figure 3-6 The call flow for Redundancy support on N3/N9 Interfaces	56
Figure 3-7 Redundancy support on the Transport Layer	56
Figure 3-8 The call flow for Redundancy support on the Transport Layer	57
Figure 4-1 Open5GS Architecture [15]	65
Figure 4-2 Test Environment deployment	65
Figure 4-3 The gNB configuration	66
Figure 4-4 The UE configuration	66

<i>Figure 4-5 VMI</i>	67
<i>Figure 4-6 The AMF Configuration</i>	68
<i>Figure 4-7 The SMF Configuration</i>	69
<i>Figure 4-8 The UPF Configuration</i>	69
<i>Figure 4-9 VM2</i>	69
<i>Figure 4-10 Log output of Nrfd (after all NFs are activated)</i>	71
<i>Figure 4-11 Log output of Bsfd</i>	71
<i>Figure 4-12 Log output of Nssfd</i>	72
<i>Figure 4-13 Log output of Udrd</i>	72
<i>Figure 4-14 Log output of Udmd</i>	72
<i>Figure 4-15 Log output of Ausfd</i>	72
<i>Figure 4-16 Log output of Pcf</i>	73
<i>Figure 4-17 Log output of Smfd</i>	73
<i>Figure 4-18 Log output of Upfd</i>	73
<i>Figure 4-19 Log output of Amfd (after gNB is activated)</i>	74
<i>Figure 4-20 Log output of gNB (after UE is activated)</i>	74
<i>Figure 4-21 Log output of UE including the Registration and PDU Session Establishment Procedures</i>	75
<i>Figure 4-22 Log output of Amfd (after UE is activated)</i>	75
<i>Figure 4-23 Capture on ens3 interfaces of VMI and VM2</i>	76
<i>Figure 4-24 N4 Node Level Messages – PFCP Association</i>	76
<i>Figure 4-25 PFCP Session Establishment Request (PDRs)</i>	77
<i>Figure 4-26 PFCP Session Establishment Request (FARs)</i>	78
<i>Figure 4-27 PFCP Session Establishment Response</i>	79
<i>Figure 4-28 PFCP Session Modification Request</i>	80
<i>Figure 4-29 PFCP Session Modification Response</i>	80
<i>Figure 4-30 Packet Processing in the UPF</i>	80
<i>Figure 4-31 Ping Session</i>	81
<i>Figure 4-32 Echo-Request – Overlay and Underlay Network</i>	81
<i>Figure 4-33 Echo Reply – Overlay and Underlay Network</i>	81
<i>Figure 5-1 The presented model in the 3GPP's Release 16 and the implementation for Redundant Transmission Support on the N3 interface technique</i>	84
<i>Figure 5-2 DL and UL Traffic Flow in the Implementation</i>	84
<i>Figure 5-3 Network Model for the implementation</i>	86
<i>Figure 5-4 Packet Duplication operation</i>	88
<i>Figure 5-5 Destination IP Address Modification operation</i>	88
<i>Figure 5-6 Running Open5GS Daemons – 5G CN NFs</i>	92
<i>Figure 5-7 Routing Table of VM2</i>	92
<i>Figure 5-8 Designed Blocks Interfaces</i>	92
<i>Figure 5-9 Sw2 IP Address</i>	92
<i>Figure 5-10 Running UERANSIM Daemons – gNB</i>	92
<i>Figure 5-11 Running UERANSIM Daemons – UEs</i>	93

<i>Figure 5-12 IP Addresses of UEs</i>	93
<i>Figure 5-13 Captures on ogstun and uesimtun0 interfaces</i>	93
<i>Figure 5-14 Hdata is not reachable when controllers are not active</i>	94
<i>Figure 5-15 Initialization of the controller for destination address modification operation</i>	94
<i>Figure 5-16 Initialization of the controller for packet duplication operation</i>	94
<i>Figure 5-17 Hdata is reachable when controllers are active</i>	95
<i>Figure 5-18 Ping from Hdata towards the UE – 1</i>	96
<i>Figure 5-19 Ping statistics when CN is lossy. The packet loss rate is slightly more than %32 as expected.</i>	97
<i>Figure 5-20 Packets with sequence number 748 are delivered by both UP paths, so hdata receives both (DUP!). However, packets sequence numbers 749 and 750 are delivered by only one of the paths.</i>	97
<i>Figure 5-21 Ping statistics when deployed redundant transmission support on the N3 interface. The improvement in packet loss rate can be seen clearly. Packets with sequence number 997 are lost in both UP paths.</i>	98
<i>Figure 5-22 Ping test is repeated while packet loss probability is %10 instead of %30.</i>	98