A UPF MODULE FOR 5G NETWORKS WITH QUALITY OF SERVICE SUPPORT: SOFTWARE IMPLEMENTATION AND REALISTIC EVALUATION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUSTAFA FATİH ÇEMEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2023

Approval of the thesis:

A UPF MODULE FOR 5G NETWORKS WITH QUALITY OF SERVICE SUPPORT: SOFTWARE IMPLEMENTATION AND REALISTIC EVALUATION

submitted by **MUSTAFA FATİH ÇEMEN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Ece Güran Schmidt
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Orhan Gazi
Electrical and Electronics Engineering, Medipol University _____

Prof. Dr. Ece Güran Schmidt
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Serkan Sarıtaş
Electrical and Electronics Engineering, METU _____

Date: 05.09.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: MUSTAFA FATİH ÇEMEN

Signature :

ABSTRACT

A UPF MODULE FOR 5G NETWORKS WITH QUALITY OF SERVICE SUPPORT: SOFTWARE IMPLEMENTATION AND REALISTIC EVALUATION

ÇEMEN, MUSTAFA FATİH
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ece Güran Schmidt

September 2023, 61 pages

5G Network provides higher reliability, lower latency, and increased bandwidth rate compared to 4G Network. Similar to the Software Defined Networking (SDN) Paradigm, 5G networks promote the separation of the control and data plane together with flow-based services. 5G Network consists of the radio access network and the core network which is essentially the second part of the access network that is not with radio technology. The modules of the 5G network are called functions.

This thesis focuses on the User Plane Function (UPF) which is the essential component of the 5G Network Core. UPF provides packet forwarding, General Packet Radio Service (GPRS) Tunnelling Protocol (GTP) functions, and port translation. UPF is a data plane component that matches each incoming packet against a Packet Detection Rule (PDR) to define its flow and then executes the corresponding per-flow actions.

This thesis implements a selection of UPF functions towards Quality of Service (QoS) enforcement. To this end, we implement the Quality of Service Flow Identifier (QFI) processing, the forwarding or dropping of the packet (gating) and the enforcement of Maximum Bit Rate (MBR) per flow.

The implementation is in software using DPDK (Data Plane Development Kit) library. We evaluate the performance using a realistic set-up and a packet generator. Our results show that the implementation can support a large number of flows at high bit rates.

Keywords: 5G network core, user plane function, 5G quality of service, gate status control, bitrate enforcement

ÖZ

5G AĞLARI İÇİN SERVİS KALİTESİ DESTEKLEYEN UPF MODÜLÜ: YAZILIM GERÇEKLEŞTİRİMİ VE GERÇEKÇİ DEĞERLENDİRME

ÇEMEN, MUSTAFA FATİH
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ece Güran Schmidt

Eylül 2023 , 61 sayfa

5G Ağı, 4G Ağına kıyasla daha yüksek güvenilirlik, daha düşük gecikme süresi ve daha yüksek bant genişliği oranı sağlar. Yazılım Tanımlı Ağ (SDN) Paradigmasına benzer şekilde, 5G ağları akış tabanlı hizmetlerle birlikte kontrol ve veri düzleminin ayrılmasını teşvik etmektedir. 5G Ağı, radyo erişim ağı ve esasen erişim ağının radyo teknolojisine sahip olmayan ikinci kısmı olan çekirdek ağdan oluşmaktadır. 5G ağının modülleri fonksiyon olarak adlandırılmaktadır.

Bu tez, 5G Ağ Çekirdeğinin temel bileşeni olan Kullanıcı Düzlemi İşlevine (UPF) odaklanmaktadır. UPF paket iletimi, Genel Paket Radyo Hizmeti (GPRS) Tünel Protokolü (GTP) işlevleri ve port çevirisi sağlar. UPF, gelen her paketi, akışını tanımlamak için bir Paket Algılama Kuralı (PDR) ile eşleştiren ve ardından ilgili akış başına eylemleri yürüten bir veri düzlemi bileşenidir.

Bu tez, Hizmet Kalitesi (QoS) uygulamasına yönelik bir dizi UPF işlevini hayata geçirmektedir. Bu amaçla, Hizmet Kalitesi Akış Tanımlayıcısı (QFI) işleme, paketin iletilmesi veya düşürülmesi ve akış başına Maksimum Bit Hızı (MBR) uygulaması

gerçekleştirilmektedir.

Uygulama Veri Düzlemi Geliştirme Kiti (DPDK) kütüphanesi kullanılarak yazılımda gerçekleştirilmiştir. Gerçekçi bir kurulumda bir paket üreteci kullanarak performansı değerlendiriyoruz. Sonuçlarımız, uygulamanın yüksek bit hızlarında çok sayıda akışı destekleyebileceğini göstermektedir.

Anahtar Kelimeler: 5G ağ çekirdeği, kullanıcı düzlemi işlevi, 5G hizmet kalitesi, kapı durumu kontrolü, bit hızı uygulaması

To my family

ACKNOWLEDGMENTS

I would like to start by expressing my gratitude to my very valuable advisor Prof. Dr. Ece Güran Schmidt, who has supported me throughout this thesis period and the courses, projects and other academic studies I have taken since my undergraduate education and who has answered my questions promptly and diligently.

I would like to express my sincere gratitude to the Bull Technology family, where I am still working, for providing a friendly work environment with engineers with high technical background. Additionally, my tennis partner Yakup Erdem Yıldız, and my violin maestro Uğur Çit, both of whom have been instrumental in assisting me during challenging times.

I would like to extend my sincere love to my family, who have wholeheartedly supported my academic achievements since day one.

Finally, I would like to thank the Scientific and Research Council of Turkey (TUBITAK) for providing scholarship (2210) support during my graduate education.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xv

LIST OF ALGORITHMS xvii

LIST OF ABBREVIATIONS xviii

CHAPTERS

1 INTRODUCTION 1

2 INTRODUCTION TO 5G CORE NETWORKS AND LITERATURE SURVEY 5

2.1 5G Network Architecture 5

2.1.1 5G and Previous Generation Networks 5

2.1.2 Overview of 5G Networks 6

2.1.3 Data (User) Plane in 5G Networks 8

2.1.4 Control Plane - User Plane Seperation (CUPS) 9

2.1.5 Network Slicing - SDN Relation 10

2.1.6 OpenFlow - PFCP Relation 11
2.1.7 Protocol Data Unit (PDU) Sessions 11
2.1.8 The GPRS Tunneling Protocol (GTP) 12
2.1.9 QoS Management 13
2.1.10 QoS Flows 14
2.1.11 5G Current Development Status 15
2.2 UPF(User Plane Function) in 5G Core Networks 16
2.2.1 UPF Functionality and Typical Implementation 16
2.2.1.1 Intel DPDK (Data Plane Development Kit) 17
2.2.1.2 OvS (Open vSwitch) 18
2.2.1.3 ovs-vswitchd 19
2.2.1.4 Datapath 19
2.2.1.5 5G Compliant Switch based on OvS (Open vSwitch) . . 19
2.2.2 UPF Performance Metrics 21
2.2.3 Literature on UPF Implementation 22
3 THE UPF DESIGN 27
3.1 5G Architecture Implementation 27
3.1.1 UPF Implementation 28
3.1.2 Access and Control Plane Interface 28
3.1.3 Data Network (Internet) Interface: 29
3.2 GTP Implementation 29
3.3 Software Architecture for User Plane 30
3.4 Ingress - Egress Policing and OpenFlow Meter 36

3.5 QER Action Implementation 37
3.6 Quality of Service Actions 38
3.7 Token-Bucket Algorithm for Maximum Bitrate Limiting 38
4 EVALUATION 45
4.1 UPF Test Environment 45
5 CONCLUSION 55
REFERENCES 57

LIST OF TABLES

TABLES

Table 4.1 Core Allocation 48

Table 4.2 3 UE, Total 3 flows 50

Table 4.3 3 UE, Total 3*1275 flows 50

Table 4.4 32 UE, Total 32*1275 flows 50

LIST OF FIGURES

FIGURES

Figure 2.1 Core architecture comparison between 4G and 5G 7

Figure 2.2 5G Network Architecture with Control and User Plane 7

Figure 2.3 5G User Plane Functionalities 9

Figure 2.4 Packet Data Unit (PDU) segments with GTP explanation 13

Figure 2.5 A typical UPF Implementation 16

Figure 2.6 The components and interfaces of OvS [1] 18

Figure 2.7 Packet processing flow in the UPF 21

Figure 2.8 QoS flow to DRB mapping [2] 26

Figure 3.1 Overall 5G architecture implementation visualization with MAC and IP address 28

Figure 3.2 GTP Headers 30

Figure 3.3 Miniflow Struct 30

Figure 3.4 Miniflow Map Example for Packets from UE to DN 32

Figure 3.5 5G User Plane Architecture 32

Figure 3.6 Flow Table Construction 35

Figure 3.7 A Match-Action Table Entry 35

Figure 3.8 PPCP QER Struct 37

Figure 3.9 Token Bucket Algorithm [3] 39

Figure 4.1 Test configuration of UPF 46

Figure 4.2 ICMP Echo Request Message from UE to DN 47

Figure 4.3 3 UE, Total 3 flows 51

Figure 4.4 3 UE, Total 3 flows with TRex Real-Time Update 51

Figure 4.5 3 UE, Total 3*1275 flows with destination IP Range 52

Figure 4.6 32 UE, Total 32*1275 flows 52

Figure 4.7 32 UE, Total 32*1275 flows with TRex Real Time Update . . . 53

LIST OF ALGORITHMS

ALGORITHMS

Algorithm 1 Modified Token Bucket Algorithm Pseudocode 42

LIST OF ABBREVIATIONS

3GPP	3rd Generation Partnership Project
5GC	5G Core
5GS	5G System
AMF	Access and Mobility Management Function
BAR	Buffer Action Rule
CN	Core Network
CUPS	Control and User Plane Separation
DL	Downlink
DN	Data Network
DPDK	Data Plane Development Kit
DSCP	Differential Services Code Point
DRB	Data Radio Bearer
EPC	Evolved Packet Core
EMC	Exact Match Cache
FAR	Forwarding Action Rule
GBR	Guaranteed Bit Rate
gNB	gNodeB
GPS	Global Positioning System
GSM	The Global System for Mobile Communications
GTP	GPRS Tunnelling Protocol
GTP-U	GTP User Data Tunnelling.
ICMP	Internet Control Message Protocol
IE	Information Element
IoT	Internet of Things

LBA	Leaky Bucket Algorithm
LTE	Long-Term Evolution
MAR	Multi-Access Rule
MBR	Maximum Bit Rate
NR	New Radio
OS	Operating System
OSI	The Open Systems Interconnection
OvS	Open vSwitch
OVSDB	Open vSwitch Database
PDU	Protocol Data Unit
PDR	Packet Detection Rule
PFCP	Packet Forwarding Control Protocol
PMD	Poll Mode Driver
QER	QoS Enforcement Rule
QFI	QoS Flow Identifier
QoS	Quality of Service
RAN	Radio Access Network
RF	Radio Frequency
RTT	Round Trip Time
SDN	Software-Defined Networking
SMF	Session Management Function
TBA	Token Bucket Algorithm
TCP	Transmission Control Protocol
TE	Traffic Engineering
TEID	Tunnel Identifier
TS	Technical Specification
UDP	User Datagram Protocol

UL	Uplink
UPF	User Plane Function
UE	User Equipment
URR	Usage Reporting Rule
VLAN	Virtual Local Area Network

CHAPTER 1

INTRODUCTION

The telecommunications industry is undergoing a major transformation. 5G, which is often seen as a technology that will enable faster speeds and support the development of Industry 4.0, is actually a fundamental shift in wireless communications. It will place wireless communications at the center of the digital economy. This transformation is not just a small improvement over previous generations of technology, but a major step change that the industry may not see again for a long time.

The 5G architecture consists of two parts: the new Radio Network (NG-RAN) supporting the New Radio (NR), and the 5G Core Network (5GC). Both parts have changed significantly from previous generations of technology.

The 5GC is designed to be a modular architecture, where each network function is implemented as a separate service. This makes it easier to add new features and services and to scale the network as needed. It uses a common set of standards for identity, authentication, quality of service (QoS), policy, and charging. This will make it easier for operators to manage their networks and provide consistent services to their customers.

It is important to note that 5G has a similar design philosophy to Software Defined Networking (SDN). To this end, the control and data planes are separate and the services are flow-based. The communication between the control and data plane is by Packet Forwarding Control Protocol (PFCP) which is similar to OpenFlow protocol. The SMF, or Session Management Function, manages the sessions of end users (or devices). This includes establishing, modifying, and releasing sessions, as well as allocating IP addresses for each session. The SMF does not communicate directly

with end-user devices but instead communicates with them indirectly through the AMF, or Access and Mobility Management Function. The AMF forwards session-related messages between the devices and the SMFs.

The main focus of this thesis is the User Plane Function (UPF) module of the 5G Core Network (5GC). The UPF is responsible for processing and forwarding user data. It connects to external IP networks and acts as a stable IP anchor point for devices, hiding their mobility. This means that IP packets destined for a specific device can always be routed from the Internet to the UPF that is serving that device, even if the device is moving around in the network.

An incoming packet is assigned to a flow by the UPF by matching the packet headers to a Packet Detection Rule (PDR). PDRs are installed by the SMF in a table and each PDR defines a series of packet processing rules for the packets of the flow.

The UPF can execute various network or user policies for each flow, such as:

- Gating: This is the process of blocking or restricting traffic. For example, the UPF could be configured to block traffic from a specific website or application.
- Redirection: This is the process of sending traffic to a different destination. For example, the UPF could be configured to redirect traffic from a specific website to a content filter.
- Data rate limitations: This is the process of limiting the amount of data that can be transferred over a network. For example, the UPF could be configured to limit the data rate for a specific user or application.

The main contribution of this thesis is the software implementation of the UPF components that realize QoS management. The implementation includes gating and data rate limitation to the Maximum Bit Rate (MBR) that is defined per flow. These features can be used by mobile network operators to facilitate customized billing or service provisioning based on customer payment arrangements. We further implement packet counting, and usage statistics found in UPF solutions by industry leaders like Napatech, Ericsson, and Nokia.

We adopt the OpenFlow protocol to implement PFCP protocol. The interface of UPF

to the Radio network requires encapsulation of GPRS Tunnelling Protocol (GTP). To this end, we implement GTP encapsulation and decapsulation.

We employ the DPDK software library and OvS architecture which is an open virtual SDN switch. Our tests with a packet generator framework confirm the correct operation of the UPF implementation, and its capability to support high data rates and a large number of flows.

The remainder of this thesis is organized as follows. Chapter 2 gives background information for general 5G network architecture and draws parallels to the Software Defined Network and Network Slicing. Moreover, the current 5G development status and quality of service management are also addressed. Chapter 3 elaborates on the software architecture between the user and control plane and overall network core architecture. Moreover, the software implementation of Quality of Service Information elements is explained in detail. In Chapter 4, we show a realistic performance evaluation of the Maximum Bitrate limiting under different traffic scenarios. Chapter 5 summarizes the conclusions of this thesis and the additional QoS information element software that we plan to implement in the future.

CHAPTER 2

INTRODUCTION TO 5G CORE NETWORKS AND LITERATURE SURVEY

This section aims to comprehensively analyze the fundamental components and advancements in 5G networks. It encompasses a discussion on the distinctions between 5G and previous generations, emphasizing the user plane's critical role and the Quality of Service (QoS) management. Additionally, the current status of 5G deployment will be examined. Furthermore, the significance of the User Plane Function (UPF) within the 5G Core (5GC)Network will be explored, along with an investigation into its performance metrics. Finally, a comprehensive literature review will be presented focusing on Intel DPDK (Data Plane Development Kit) and Open vSwitch (OvS). This complete exploration aims to provide valuable insights into the evolution and potential impact of 5G networks on the future of communication technology.

2.1 5G Network Architecture

2.1.1 5G and Previous Generation Networks

The 3rd Generation Partnership Project (3GPP) 5G Core architecture was designed with the fundamental principle of excluding backward compatibility with preceding radio access networks such as GSM and LTE. With the progression of new access network generations, distinct methodologies emerged concerning the division of functionalities between the core network and the radio network, along with implementing novel protocols to establish their interconnection. For instance, during the development of LTE (4G) around 2007-2008, the introduction of the IP-based S1 interface facilitated the linkage between the radio and core networks. Consequently, over time, the complexity of network architecture increased, resulting in service providers

deploying a combination of 2G, 3G, and 4G technologies across various frequency bands to ensure comprehensive coverage.

The 5G Core architecture represents a significant shift in perspective as it seeks to create an "access independent" interface that can be applied universally to various relevant access technologies, including those not explicitly stated by the 3GPP, such as fixed access. This forward-looking approach aims to maximize future adaptability and longevity. As a result, the 5G Core architecture is designed to accommodate potential developments and innovations in the telecommunications environment [4].

A comparison between the 5G Core (5GC) architecture and the current Evolved Packet Core (EPC) architecture reveals notable similarities and differences. User data processing components and their integration with 3GPP radio access networks show significant similarities between the new 5GC architecture and the traditional EPC network architecture originally designed for 4G/LTE. However, there is a distinct difference in the part of the network that is only responsible for signaling-related functions.

An additional difference lies in the way the 5G Core architecture is visualized and described in two different ways. The first visualization depicts the interconnection of various network functions. In this visualization, "Service Based interfaces" are used as an essential difference from previous 3GPP architectures. This concept requires network functions encompassing logic and functionality to handle signal streams that are not interconnected via traditional point-to-point interfaces. Instead, they serve and provide other network functions. In every interaction between network functions, one entity serves as the "Service Consumer." In contrast, the other acts as the "Service Producer" [5]. A graphical representation of this architecture and its comparison to 4G networks is shown in Fig. 2.1.

2.1.2 Overview of 5G Networks

5G networks have two main functionalities, namely, control plane and data plane functionalities. These functions are visualized in Fig. 2.2, and the main components can be summarized as follows:

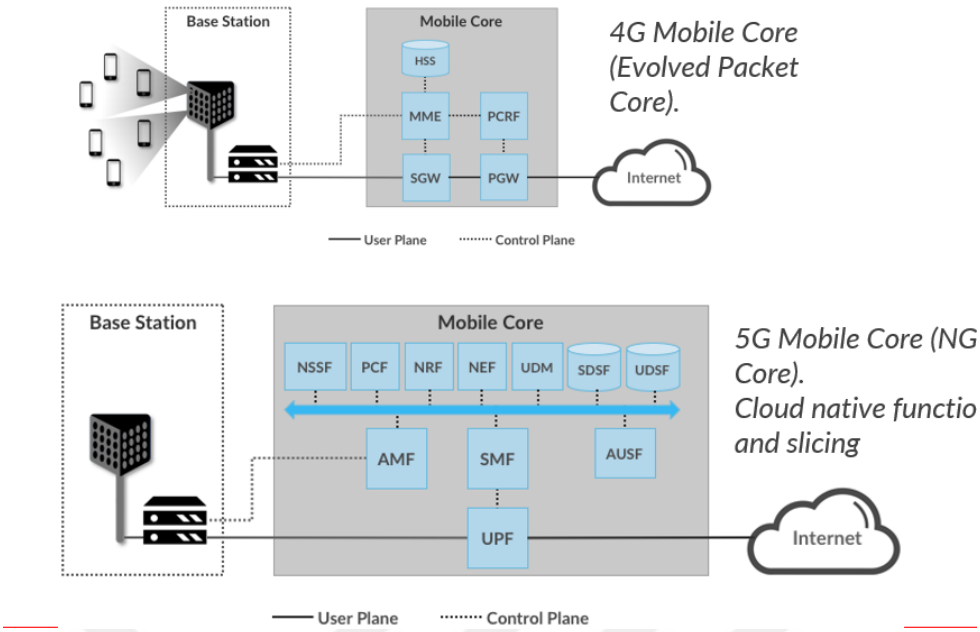

Figure 2.1: Core architecture comparison between 4G and 5G

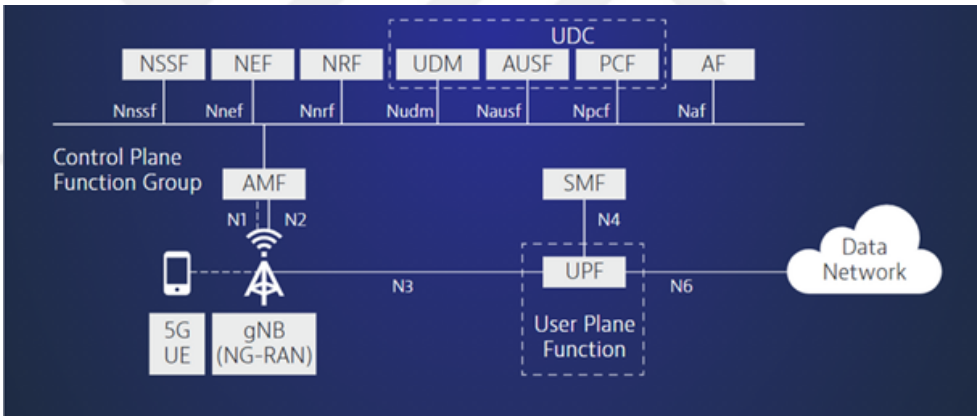

Figure 2.2: 5G Network Architecture with Control and User Plane

- Control Plane Functions can be further partitioned for easy management.
- User Equipment (UE) is a 5G mobile phone, and gNB is the mobile base station.
- For Radio Technology, 5G New Radio (NR) implements various advanced 5G RF technologies. For instance, LTE is used for 4G Networks.
- Data Network is the Internet.

To ensure reliable communication among these functions, 5G networks rely on the definition of Information Elements (IEs) within the specifications and standards, as outlined by the 3GPP. These IEs serve as structured data units, and these elements are essential for facilitating signaling and communication between different network elements. Adopting diverse protocols and interfaces in 5G necessitates using specific IEs, enabling the exchange of relevant information. Ultimately, integrating IEs contributes to the effective functioning and operational efficiency of the entire 5G network.

2.1.3 Data (User) Plane in 5G Networks

The UPF is a hardware/software module that forwards UE traffic between the access networks, such as the 5G New Radio and the Data Networks. The main functionality of the UPF is connecting the 5G Access Network, which is composed of 5G mobile phones and base stations, to the Internet, and this functionality is illustrated in Fig. 2.3. The principal role of the data plane lies in the necessity for the UPF implementation to handle each packet individually while also managing per-flow information [6]. Additionally, the UPF encompasses various other significant functionalities, which are outlined below:

- UPF is the interconnect point between the mobile infrastructure and the Data Network (Internet) for encapsulation and decapsulation of GPRS Tunnelling Protocol for the user plane functionalities.
- UPF is responsible for searching the IP lookup table, calculating checksum, and port translation.
- Packet routing and forwarding tasks also belong to UPF.
- Handling Protocol Data Unit (PDU) sessions for differentiated QoS.
- Sending reporting triggers to control plane such as traffic usage report.
- Applying QoS metrics such as rate limiting.
- Packet marking such as transport level marking (DSCP).

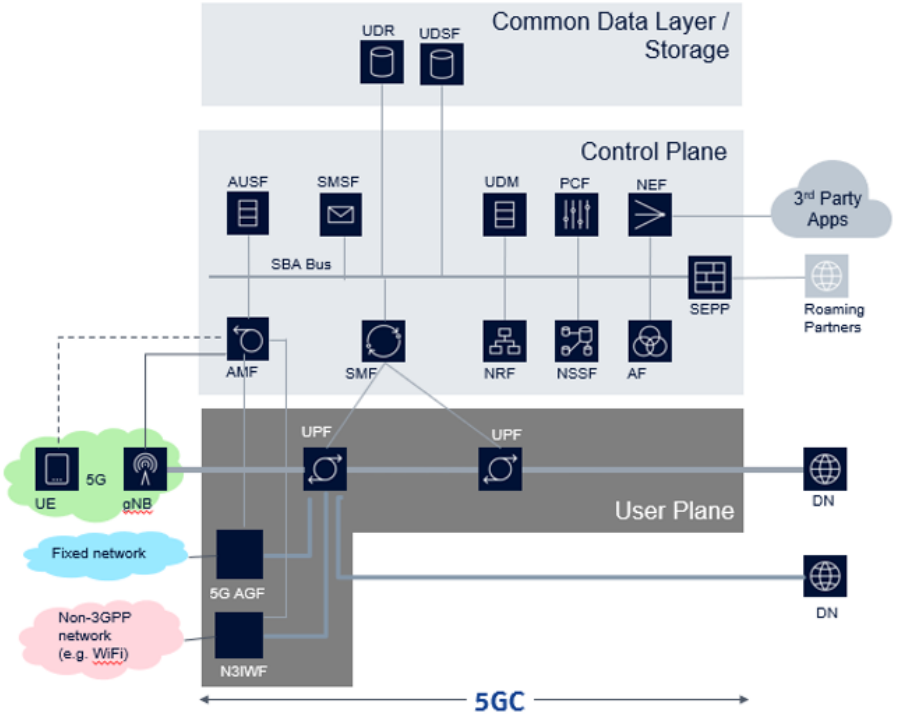

Figure 2.3: 5G User Plane Functionalities

2.1.4 Control Plane - User Plane Seperation (CUPS)

The User Plane architecture in 5th Generation Systems (5GS) incorporates a fundamental aspect known as CP-UP separation, which is considered an obligatory component of the 5G architecture. In contrast to the Evolved Packet Core (EPC), where the CP-UP split, commonly referred to as "CUPS" (Control and User Plane Separation of EPC nodes), was introduced as an optional addition in Rel-14, the 5G Core includes it from its beginning.

Several reasons drive the integration of CP-UP separation into 5GC. First, it enables networks' flexible deployment and operation, offering options for distributed and centralized configurations. Moreover, this separation allows independent scaling of the control and user plane functions, facilitating efficient resource management. As mobile operator networks face escalating traffic volumes, the demand for cost-effective User Plane solutions becomes paramount. These solutions must meet end-user requirements for high bitrates and low latency and be sustainable for the mobile opera-

tors in the long term [7].

Software Defined Network (SDN) is a network architecture paradigm that aims to enhance traditional networks' flexibility, manageability, and programmability through CP-UP separation. In conventional network design, the control plane is responsible for making routing decisions; on the other hand, the data plane is responsible for forwarding data packets. SDN achieves this separation by decoupling the network's control plane from the data plane. This separation allows network administrators and operators to centrally manage and configure network resources through a software-based controller, enabling dynamic and automated network provisioning. SDN introduces a more abstract and programmable view of the network, allowing for a more straightforward implementation of innovative applications and services and enabling efficient resource utilization. By separating the control logic from the underlying hardware, SDN provides optimization of network performance and security enhancement.

2.1.5 Network Slicing - SDN Relation

Network slicing is another essential component of 5G architecture. Network slicing architecture is similar to the Software Defined Network (SDN) idea, a common concept nowadays due to its strong relationship with cloud applications. In traditional network design, the networking device hardware router has two essential functions. The first one is control plane functionalities in which the router Determines how and where packets are forwarded according to the algorithms like shortest path. The second is data plane functions in which the router forwards packets from incoming to outgoing links. In SDN, control plane functionalities like making decisions are removed from the router, and this path decision problem is assigned to a centralized Network Operating System (OS) that has a global view of the overall network. As a result, the router is converted into a simple packet-forwarding device. Moreover, OpenFlow protocol is introduced for communication between newly introduced Network OS and routers. SDN relation to 5G can be summarized as network slicing is the combination of SDN and network virtualization. The virtualized network allows to create subnets to provide connectivity more adjusted to specific needs. The creation

of subnetworks will give particular characteristics to a part of the network, being a programmable network, and will allow prioritizing connections; for example, public safety emergencies could be put in front of other users. This application can be realized by applying different latencies or prioritizing them in the connection to the network so that they can't be affected by possible overloads of the mobile network.

2.1.6 OpenFlow - PFCP Relation

OpenFlow, an integral component of Software-Defined Networking (SDN), is a protocol that facilitates dynamic network management by decoupling the control plane from the data plane, enabling centralized network control. Network administrators can employ a programmable and flexible approach through the OpenFlow framework, directing network traffic flow and configuring network devices in real-time. By providing a standardized interface between the control and data planes, OpenFlow enables the implementation of diverse applications, such as traffic engineering, load balancing, and security management. This paradigm shift in networking empowers researchers and practitioners to innovate and optimize network functionalities effectively, leading to increased scalability and adaptability in modern communication infrastructures.

In the context of 5G networks, PFCP is the corresponding CUPS separation protocol, similar to OpenFlow in SDN.

The following section will discuss another important part of the 5G Network Core: Protocol Data Unit sessions and QoS management.

2.1.7 Protocol Data Unit (PDU) Sessions

A Protocol Data Unit (PDU) session between a specific UE and the UPF is established before data transmission in 5G networks. UPF can be implemented in software, and it can be executed inside either a virtual machine or containers (termed as a UPF instance). Each UPF instance may handle several concurrent PDU sessions from various users. However, operators should control the number of UPF instances to

save resources in the provision of QoS.

Initially, UE, which is a 5G mobile phone, initiates Registration Procedures for the network. After authentication steps are completed successfully, a UE requests a PDU session to a specific data network, and the Session Management Function (SMF) decides the acceptance based on the system's actual conditions and the PDU session's QoS requirements. Upon acceptance, the PDU session is tunneled through the transport network to the UPF, and the communication between the UE and the data network can start.

2.1.8 The GPRS Tunneling Protocol (GTP)

The GTP is a crucial component of mobile networks, employed not just in 5G but also in 3G and 4G (LTE) networks. It enables the smooth transmission of user data and signaling messages between different network elements. Operating at the application layer of the OSI model, it plays a vital role in supporting mobility, session management, and data forwarding. In 5G, GTP is used to establish and maintain tunnels between various network elements, enabling efficient transportation of user data between the UE and the core network. For 5G networks, the UDP source and destination port number is 2152 as specified in [8]. The QFI (QoS Flow Identifier) and TEID (tunnel id) are the most important elements of GTP in terms of QoS flow management. The QFI value is a 6-bit field that helps the network differentiate between different QoS flows in a PDU session. Each QoS flow may have unique properties such as priority, loss tolerance, latency requirements, and data rate limits. The use of QFI ensures that the network follows the appropriate policies and QoS profiles, providing the necessary resources for each QoS flow. The TEID is a 32-bit identifier used for the endpoints of GTP tunnels. Each endpoint in the UE or core network has a unique TEID, which helps differentiate it from other tunnels. This ensures that the receiving end can accurately identify the associated tunnel and deliver data packets to the correct destination. The PDU packet, which includes GTP with QFI and TEID, is depicted in Fig. 2.4.

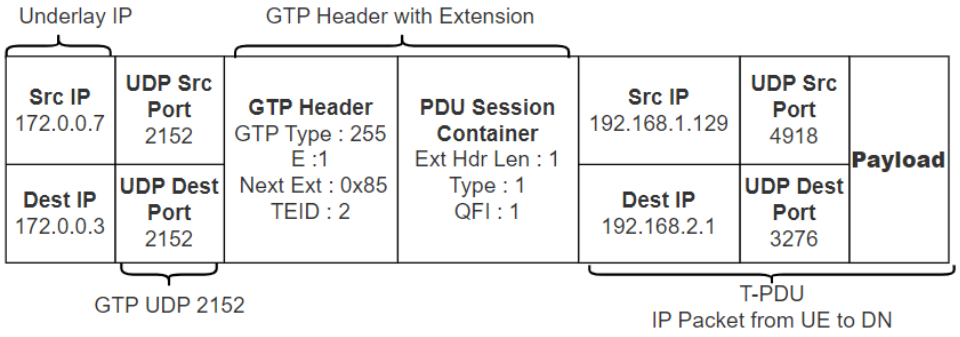

Figure 2.4: Packet Data Unit (PDU) segments with GTP explanation

2.1.9 QoS Management

In 5G networks, subscriber equipment generates highly changing dynamics of PDU session requests. To guarantee QoS, each UPF instance should handle a limited number of concurrent PDU sessions. Therefore contention for resources is expected. Operators may launch new UPF instances when there are more requests for PDU sessions and terminate idle ones when few customers need PDU sessions. Therefore, operators should apply some algorithms for managing UPF instances efficiently.

The 5G standards support the connectivity of UE with various types (IP, Ethernet, unstructured) of external data networks. 3GPP designed the 5G core based on the Service Based Architecture and the total control and user plane separation. SDN architecture for separating these two functionalities, namely the control and data planes, is mentioned in Section 2.1.4. The control plane of 5G Systems (5GS) includes the Access and Mobility Management Function (AMF) and the Session Management Function (SMF). 5G base stations and the UPF perform the tasks of the 5G data plane; that is, they provide necessary procedures to convey data flows between end devices and data networks. Before the communication of a specific UE and a data network, a PDU session should be started and handled by UPF. The Session Management Function is responsible for managing user sessions and assigns a PDU session to an appropriate UPF instance.

2.1.10 QoS Flows

In the context of 5G, the QoS framework relies on QoS Flows, which represent the most detailed level of QoS differentiation. QoS parameters and characteristics are directly linked to each QoS Flow. These flows are distinguished by a unique identifier known as the QoS Flow ID (QFI), which is specific to a Packet Data Unit (PDU) Session. The Next Generation Radio Access Network (NG-RAN) can establish a (Data) Radio Bearer for each QoS Flow or combine multiple QoS Flows into a single (Data) Radio Bearer based on NG-RAN's internal logic. This means there is no strict one-to-one mapping between Data Radio Bearers and QoS Flows. Instead, the NG-RAN has the flexibility to handle its resources in a way that it deems most suitable while ensuring that the QoS requirements for each QoS Flow are met. As this study mainly focuses on the Core Network, the Radio Access Network will not be extensively analyzed. For a more detailed analysis of NG-RAN, [9] can be examined.

Regarding the transmission of QFI information, it is conveyed through an encapsulation header called GTP-U on N3 (and N9) interfaces without modifying the end-to-end packet header. Data packets marked with the same QFI receive identical traffic forwarding treatment, including scheduling and admission threshold mechanisms. The QoS Flows can be categorized as GBR QoS Flows, which necessitate a guaranteed flow bit rate, or Non-GBR QoS Flows, which do not have such strict requirements.

Access Network (NG-RAN) classifies and differentiates the forwarding of data packets in both the Downlink (DL) and Uplink (UL) directions. DL packets, arriving at the UPF and heading toward the UE, are compared against Packet Detection Rules (PDRs) established by the Session Management Function (SMF). These PDRs employ IP 5-tuple filters for classification purposes. Each PDR is associated with one or more QoS Enforcement Rules (QERs) that contain information on how to enforce specific parameters, such as bitrates. Additionally, the QERs include the QFI value, which is added to the GTP-U header (N3 encapsulation header). QoS Flows to DRB mapping is illustrated in Fig. 2.8; however, since the main focus of this thesis is on QoS management at the network core side, detailed information on the access network side is not described, interested readers can refer to [2] for the detailed analysis.

As outlined in Section 2.1.7, the primary goal of establishing the UE's PDU Session is to create a Default QoS Flow connecting the UE and the Data Network (DN) through the gNB and the 5GC. Subsequently, this established Default QoS Flow enables the UE to conduct data exchange with the DN. Within the 5G context, the QoS Flow represents the most refined level of a traffic flow, allowing telecommunication companies to apply charging mechanisms based on specific QoS metrics.

The Default QoS Flow is classified as a non-Guaranteed Bit Rate QoS Flow, lacking both uplink (UL) and downlink (DL) Packet Filters. It possesses the lowest priority in terms of traffic mapping. In simpler terms, if UL or DL traffic fails to match any Packet Filters in the other Dedicated QoS Flows within a UE's PDU Session for the DN, the Default QoS Flow will be utilized to forward the UE traffic. Additionally, the Default QoS Flow can be employed by a UE to signal an Application Function for the establishment of Guaranteed Bit Rate QoS Flows, particularly for applications that demand high bandwidth capabilities [10].

In the final stage, DSCP (Differential Services Code Point) plays a crucial role in establishing the end-to-end QoS Stream for uplink (UL) traffic. The SMF (Session Management Function) is responsible for configuring the DSCP marking to the UPF over the N4 PFCP interface using the Forwarding Action Rule (FAR) structure during the QoS Flow setup. In the context of 5G networks, DSCP marking means the classification and prioritization of data packets based on specific code points within their headers. These code points are strategically assigned to different service types in accordance with specific QoS requirements. By implementing DSCP marking, network operators can efficiently manage and prioritize traffic flow, thus ensuring that essential data such as real-time communication and high-priority applications receive preferential treatment and appropriate QoS processing. This mechanism is vital in maintaining a consistent and efficient data flow throughout the 5G network, ultimately increasing overall network performance and improving user experience.

2.1.11 5G Current Development Status

In 2019, numerous companies made announcements regarding their 5G network deployments, primarily based on the 3GPP Release 15. However, these initial 5G im-

plementations lacked various crucial features and service specifications, including the new 5G Core (5GC), support for network latency as low as one millisecond, and the primary performance indicator for 5G user data transmission known as UPF performance.

The projected adoption rate for 5G is significantly different from that of previous generation networks (3G and 4G). This is because while previous technologies were driven by mobile internet usage, 5G is expected to be mainly driven by new IoT applications such as connected and self-driving cars [11].

2.2 UPF(User Plane Function) in 5G Core Networks

2.2.1 UPF Functionality and Typical Implementation

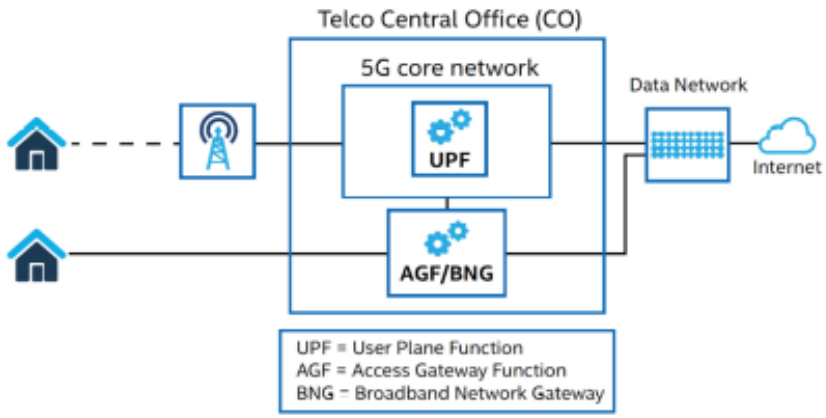

Figure 2.5: A typical UPF Implementation

From an architectural perspective, the UPF module exhibits numerous resemblances to a router. Nevertheless, it distinguishes itself from a conventional router by functioning as a multi-layer switch responsible for user plane data forwarding in both access and data networks. The implementation of UPF has been demonstrated in two different studies. One study utilizes the Click modular router [12] to achieve flexible

packet processing [13]. The other study employs an SDN switch based on OvS [14] for UPF implementation. The following example can be examined to understand better the UPF design proposed in these studies. As illustrated in Fig. 2.5, the UPF is located within the Telco Central Office (CO). As a flow-based packet processor, the UPF is responsible for the processing and forwarding data packets between the UE and the Internet (data network). Through its Match Action capabilities, the UPF efficiently manages packet routing between the DN and the Radio Network, effectively serving as a gateway connecting the access network with the Internet. In addition to its routing capabilities, the UPF also performs critical functions like encapsulating and decapsulating GPRS tunneling and applying QoS metrics. This makes it an essential component in maintaining reliable data transmission and connectivity in the 5G network architecture. However, the conventional forwarding scheme based on the Linux kernel often performs poorly when executing these vital tasks. To enhance packet processing, software-based solutions have been explored. Two notable solutions are Intel DPDK and OvS, which will be further elaborated on in the following section.

2.2.1.1 Intel DPDK (Data Plane Development Kit)

The DPDK operates on Linux operating systems and serves as a replacement for the conventional network stack, adopting a run-to-completion model. This approach necessitates the allocation of all resources before data plane applications run on logical processing cores. DPDK employs PMD (Poll Mode Driver) for device access to minimize overhead in high-speed scenarios, thereby reducing the impact of interrupt processing. An essential component of DPDK is the Environment Abstraction Layer (EAL), which conceals hardware and software specifics while providing a library function interface that binds the DPDK to applications. All DPDK-utilizing applications must include the EAL's header files, offering functionality such as Memory Management, Buffer Management, and processor core and socket management. DPDK libraries execute entirely in userspace and are optimized for high performance, carrying out fundamental tasks similar to those performed by the Linux network stack.

2.2.1.2 OvS (Open vSwitch)

OvS is a well-established virtual switch that operates at multiple layers and is released under the open-source Apache 2.0 license. It facilitates Software-Defined Networking (SDN) control semantics through the OpenFlow protocol, along with its OVSDB management interface. The software can be obtained from [15] and is accessible through various Linux distributions. OVS serves as an OpenFlow-capable software switch, where an OpenFlow controller provides instructions to the datapath on how to handle different packet types, known as flows, which define actions such as forwarding, output, or modifying VLAN tags. The process of matching a packet with a flow is referred to as flow matching. Specific flows are cached in the datapath to optimize performance, while others are stored in the userspace components.

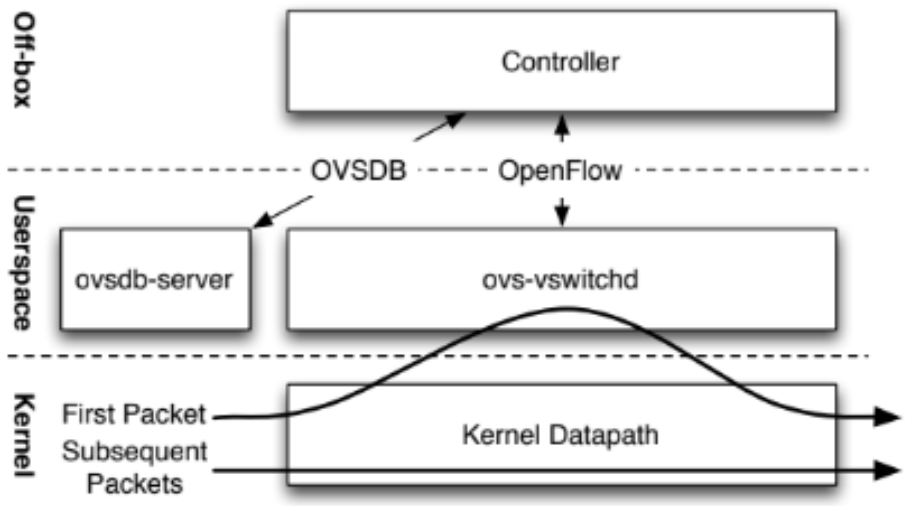

Figure 2.6: The components and interfaces of OvS [1]

In Fig. 2.6, the packet forwarding process of OVS is depicted. When a packet arrives at the NIC, it enters the OVS datapath. If a flow matching the packet is found in the datapath, the actions specified in the flow are executed accordingly. If a flow is not matched (flow missing), the packet is forwarded to ovs-vswitchd, where another flow-matching process is performed. After ovs-vswitchd determines the appropriate handling for the packet, it is returned to the datapath with the desired actions. The

datapath is often instructed to cache the flow for efficient handling of similar packets in the future.

2.2.1.3 ovs-vswitchd

OVS-vswitchd serves as the primary user space program that functions as a critical daemon in the OvS infrastructure. Its primary responsibility includes the management of flow matching and packet routing tasks. As a core component of the OVS system, ovs-vswitchd is responsible for obtaining the desired configuration settings for OvS via an Inter-Process Communication (IPC) channel from ovsdb-server, a key database server component. By obtaining configuration details from ovsdb-server, ovs-vswitchd ensures the smooth operation of the OVS data plane in compliance with specified network policies and configurations.

2.2.1.4 Datapath

The Datapath represents the primary packet forwarding module of OvS, strategically implemented in the kernel space to ensure high performance. Its primary function involves caching OpenFlow flows and executing actions on received packets that match specific flow(s). In cases where no flow is matched for a particular packet, the packet is forwarded to the user space program, ovs-vswitchd. Typically, ovs-vswitchd will generate a new flow for the datapath, which will subsequently handle packets of this type. The exceptional performance of the Datapath stems from the fact that the majority of packets successfully match flows within the datapath, thereby allowing them to be directly processed in the kernel space. This streamlined process contributes to the overall efficiency and effectiveness of packet handling within the OvS architecture.

2.2.1.5 5G Compliant Switch based on OvS (Open vSwitch)

The use of DPDK in the data plane of the 5G compliant switch design can enable high-performance UPF design in the 5G network [16]. The rte-mbuf packet descriptor [17] in the DPDK library is used to move packet data from the kernel level to

the application level without duplication. In addition, the datapath flow cache used by the OVS (Open Virtual Switch) application for cache-defined flow acceleration architecture, which is another technology that can be used in the data plane, has been examined, and the mechanisms for adding, deleting, and managing new cache entries have been studied. From the papers reviewed, [18] reveals that adding and removing rules in OVS is possible. Building upon this concept, to understand the technologies that can be used on the Control Plane, the OpenFlow protocol, which replaces the 5G protocol PFCP and is the protocol used in OvS for communication between the control plane and the user plane, is analyzed. Openflow [19] is similar to PFCP in many respects and was one of the protocols that could be used as a replacement before the PFCP protocol was developed. OpenFlow facilitates enhanced network control and flexibility, aiming to optimize network operations effectively. By offering interfaces for configuring traffic and monitoring the network, OpenFlow enables the assessment of whether end-to-end QoS constraints, such as guaranteed bitrate, are being fulfilled. Additionally, it supplies essential input for Traffic Engineering (TE) approaches to calculate appropriate path [20]. In addition, the OvS application uses OpenFlow as a control plane protocol [21]. The structure of OVS has been analyzed as it receives OpenFlow messages, parses them, and stores the message details along with session information. This examination has led to the realization that the OpenFlow messages used by OVS can be substituted with PFCP (Packet Forwarding Control Protocol) messages that are endorsed by 5G networks [22].

As discussed in Section 2.1.6, PFCP serves as the communication protocol between the control and user planes within 5G networks. This protocol operates similarly to OpenFlow in SDN. [23] provides a comprehensive explanation of all the communication details between the UPF and the control plane, particularly the SMF. The rules of all the tasks of the UPF are communicated to the UPF via the PFCP protocol under the control of the SMF. The most important of these rules, PDR and FAR, are crucial in designing the control and data plane architecture. Apart from these two rules, other rules such as URR, QER, BAR, and MAR are also included in the PFCP protocol. The path that traffic data will follow upon arrival at the UPF is determined by a set of rules. These rules and their relationship to PFCP are illustrated in Fig. 2.7. In 5 G-compliant switch implementation, the OpenFlow protocol is adapted to

accommodate these rules. Consequently, the software architecture between the user and control plane is constructed.

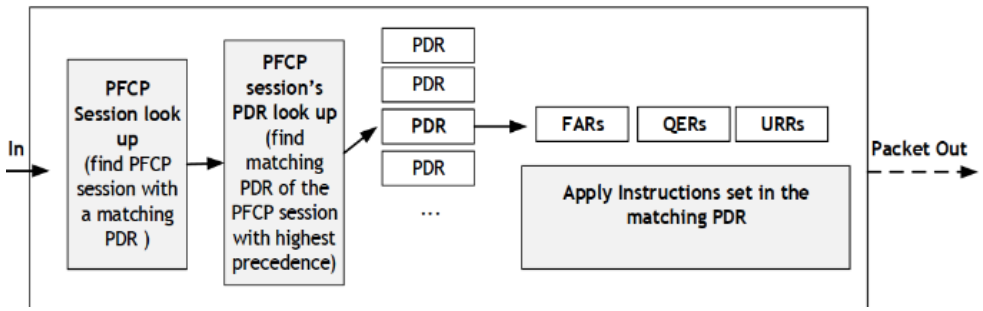

Figure 2.7: Packet processing flow in the UPF

2.2.2 UPF Performance Metrics

The detailed performance measurement specifications for all 5G components of 3GPP can be found in [24]. The key performance indicators of 5G networks, including peak data rate, user plane latency, and connection density, are explained in [25]. As for the UPF performance measurements, the impact of a number of QoS flows [14] to throughput will be investigated in this thesis.

- Latency of the User Plane: In a network, latency measures the time it takes for some data to reach its destination across the network. It is usually measured as a round trip delay - the time taken for information to get to its destination and back again. The round-trip delay is an important measure for the TCP/IP network since the TCP network waits for an acknowledgment to come back before sending any more packets. Therefore, the round-trip delay has a key impact on the network's performance. Latency is usually measured in milliseconds (ms). As explained in [25], the user plane latency for UL and DL should be 0.5ms. Therefore, it is expected that the latency for the 5G network would be one millisecond.

- Connection density: The total number of devices that satisfy QoS metrics per km^2 is one million [25].

- Peak throughput: 5G network supports up to 10Gbit/sec for the data rate experienced by user [25].

2.2.3 Literature on UPF Implementation

The conventional Linux kernel-based forwarding scheme exhibits poor performance. Consequently, three technologies have been developed to accelerate 5G UPF packet processing: eXpress Data Path (XDP), Intel DPDK, and P4 (Programming Protocol-Independent Packet Processors) [26]. Each of these technologies serves distinct purposes and exhibits various strengths. XDP, operating in the Linux kernel space, enables high-performance packet processing at the earliest stage of network stack processing. It is well-suited for low-latency, high-throughput packet filtering, and forwarding, making it ideal for scenarios that demand efficient handling of large packet volumes. On the other hand, DPDK is a collection of libraries and drivers that allow user-space applications to directly access and manipulate network packets, bypassing the kernel networking stack. It is specifically designed for high-performance data plane applications, such as network function virtualization (NFV), software routers, and load balancers, where low latency and high packet throughput are crucial. P4, a domain-specific language, offers a high-level abstraction for defining packet processing in network devices. Users can specify packet processing behavior independently of the underlying hardware, making it suitable for programmable network devices like switches and NICs, commonly used in Software-Defined Networking (SDN) environments. Each technology follows a different programming language, where XDP programs are written in C or eBPF and run in the kernel space, DPDK applications are written in C or C++ and operate in user space, and P4 employs a domain-specific language for expressing packet processing rules and actions, which are then compiled into target-specific instructions. Concerning performance, both XDP and DPDK deliver exceptional results. XDP achieves low latencies and high packet rates due to its early-stage kernel processing, while DPDK applications benefit from bypassing the kernel to maximize packet processing throughput [27, 28]. Regarding flexibility and programmability, XDP has limited capabilities compared to DPDK, primarily focusing on packet filtering and forwarding. In contrast, DPDK provides a high degree of flexibility in user space, allowing developers to implement custom packet processing

logic. In addition to the advantages of the flexibility supplied by DPDK, as demonstrated in reference [26], DPDK also exhibits lower tail latencies compared to XDP at low packet rates due to its poll mode driver. Regarding use cases, DPDK is frequently utilized in networking applications that demand high-speed packet processing, such as NFV, software routers, and load balancers. Meanwhile, P4 is commonly employed for defining the behavior of programmable network devices, such as programmable switches and NICs in SDN environments. It is important to note that P4's performance heavily depends on the underlying hardware platform and the optimization of the P4 program for that specific platform. Additionally, it should be noted that the focus of this thesis is the implementation of a software-based UPF along with the application of QoS rules. Due to the limitation of P4 in expressing packet schedulers, which results in the inability to implement QERs (QoS Enforcement Rules) [29], DPDK emerges as a more suitable solution compared to P4 for this particular implementation.

Companies with advanced technologies like Napatech incorporate the use of UPF solutions to achieve fast data transmission by directing user traffic to an accelerator card alongside a general-purpose processor platform. This allows for a decrease in CPU load due to a greater number of UPF services being offloaded to the accelerator card. In contrast to this approach, this thesis presents the development of a 5G core network UPF using an utterly software-based solution to minimize equipment costs. To overcome performance issues in the software-based solution, a DPDK-based user mode driver model is adopted, and poll mode drivers are utilized to replace the interrupt mode. This modification effectively reduces timing and switching issues caused by interrupts and also prevents the additional overhead associated with extensive memory copies [16].

Last but not least, DPDK stands out for its excellent performance, desired level of programmability, and ability to apply the QoS Enforcement Rule, making it a perfect choice for high-speed user space packet processing in software-based UPF solutions.

The next step after deciding on the technology to be used in the UPF implementation is to proceed with selecting testbed elements. As 5G networks consist of various components, including UE, gNB (RAN), and the control plane, designing the test

environment to encompass 5G networks comprehensively is essential. As per the choices made, outlined in [28], we employed UERANSIM [30] - an open-source simulator for UE and RAN - to simulate UE and gNB. UERANSIM is utilized to conduct end-to-end 5G communication experience. Furthermore, when taking into account other components of the 5G network, such as control plane elements, the literature highlights open-source applications such as free5GC and open5GS to simulate end-to-end 5G communication. Open5GS is more advantageous than free5GC in terms of control plane implementations. These advantages can be summarized as follows [31]:

- Reduced processing delays.
- Enhanced Stability: open5GS demonstrates greater stability, particularly when dealing with burst UE scenarios.
- Improved Memory Pool Allocation: open5GS addresses and mitigates memory pool allocation errors, particularly in the context of the AMF, Unified Data Management (UDM), and Unified Data Repository (UDR) components, when handling burst UEs.
- Reduced SBI Connection Timeout: open5GS minimizes the occurrence of Service-Based Interface (SBI) connection timeouts, leading to more reliable and efficient communication between network elements.

Hence, we select open5GS for control plane function implementation in this work.

After the testbed construction is determined, the last step is to establish the test architecture, select the appropriate software or simulation tools for measuring, and define performance metrics. In [32], the DPDK-based UPF was evaluated by utilizing a high-speed packet generation tool on a separate computer. Two cores were assigned for UPF testing. To assess the performance of the DPDK-based UPF, they employed Pktgen, a testing tool running on another PC. Pktgen generates testing packets and transmits them to UPF through NIC. Upon receiving the packets, the DPDK-based UPF performs various operations like IP lookup, IP/port translation, and adding/removing GTP tunnel headers depending on the type of packet received (with or without

tunneling). Ultimately, performance is evaluated in terms of maximum throughput and stability. However, unlike [32] the TRex Traffic generator proposed in [28], and we employ TRex [33] in the test machine. Furthermore, measurement methods are studied and inspired by observing the impact of the number of QoS Flows on UE throughput in [27]. This study aims to investigate the effect of the number of UE and the number of QoS flows on rate-limited throughput.

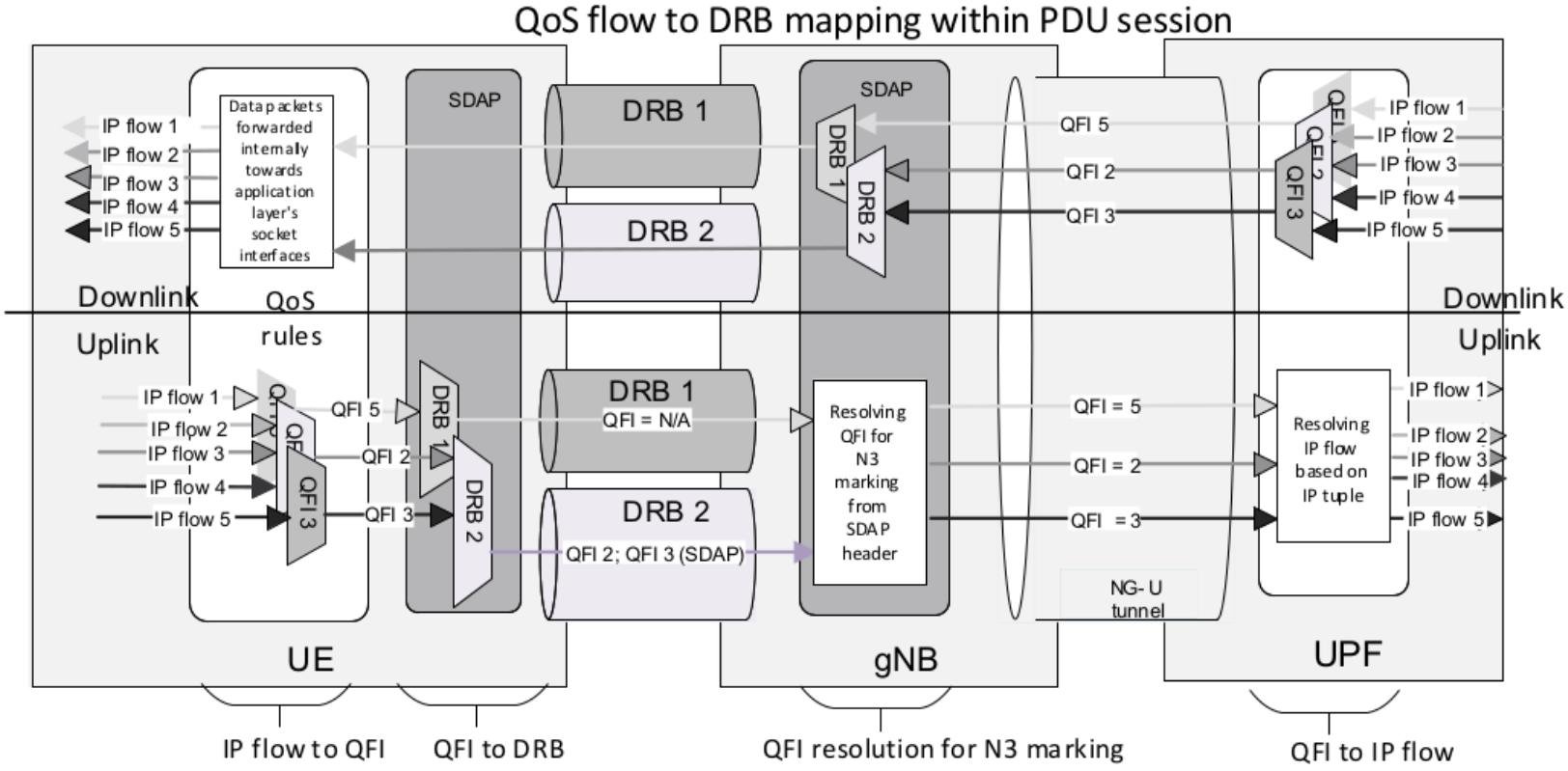

Figure 2.8: QoS flow to DRB mapping [2]

CHAPTER 3

THE UPF DESIGN

The main contribution of this thesis is the software implementation of the UPF components that realize QoS management. To this end, we implement Gate Status and Maximum Bit Rate (MBR) enforcement functionalities which are represented by the respective IE. The Gate Status IE serves to determine whether packets should be forwarded (when the gate is open) or discarded (when the gate is closed) in both the uplink (UL) and downlink (DL) directions. The MBR IE is included when there's a need to apply MBR enforcement action to packets that match a specific PDR. When this IE is present, it specifies the maximum bit rate for both uplink and downlink directions for the packets of the specified flow for the IE. We extend the token-Bucket algorithm according to the network core software architecture to implement MBR IE. In addition to implementing QoS management in the data plane, we also adapt the OpenFlow protocol to implement the PFCP protocol. Furthermore, we implement GTP encapsulation and decapsulation as we explain in Section 2.1.8. We employ the DPDK software library and OvS architecture that we explain in Section 2.2.1.2. The following sections provide more information on these implementation details.

3.1 5G Architecture Implementation

In Fig. 3.1, a general representation of all 5G components is provided. Network interfaces connected to the UPF can be listed as Control Plane (N4), Access Network (N3), and Data Network (N6). The detailed implementations of the UPF and its associated interfaces are explained in the following subsections.

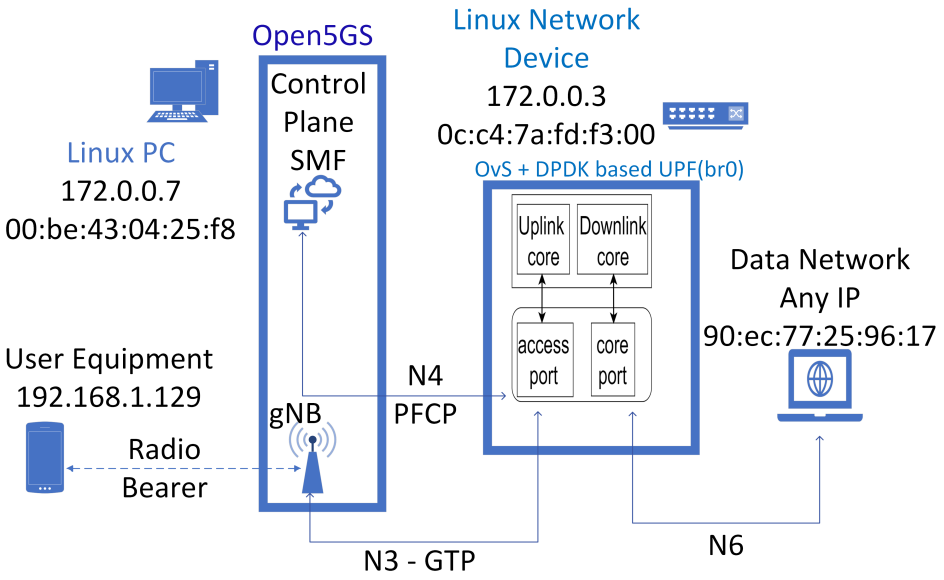

Figure 3.1: Overall 5G architecture implementation visualization with MAC and IP address

3.1.1 UPF Implementation

We develop the UPF software implementation using OvS software switch. OvS is an open-source virtual switch that operates at the data link layer. Its implementation uses Intel's DPDK solution. To this end, our software implementation operates on a network device running Linux with DPDK-enabled ports. Logically, the UPF software implementation functions as a virtual switch and is designated as the bridge (br0) within the OvS implementation. The MAC and IP addresses assigned to the br0 on the network device can be exemplified as follows:

- UPF Source MAC Address for N3, N4 and N6 Interface: "0c:c4:7a:fd:f3:00"
- UPF Source IP Address for N3, N4 and N6 Interface: "172.0.0.3"

3.1.2 Access and Control Plane Interface

In the established setup, both the access and control plane functions run on another Linux PC under the Open5GS application which was discussed in Section 2.2.3.

Since both the access and control plane functions run on the Open5GS application, destination access and destination control MAC address corresponding to the MAC and IP addresses assigned to the Ethernet interface on the Linux PC. These addresses can be exemplified as follows:

- UPF Destination MAC Address for N3 and N4 Interface: "00:be:43:04:25:f8"
- UPF Destination IP Address for N3 and N4 Interface: "172.0.0.7"

3.1.3 Data Network (Internet) Interface:

Here, the MAC address of the modem, which serves as the gateway to the external world (data network or internet), can be entered.

- UPF Destination MAC Address for N6 Interface (Modem): "90:ec:77:25:96:17"
- UPF Destination IP Address for N6 Interface: The IP Address of any website

3.2 GTP Implementation

GTP is the data transport protocol to be used between UPF and gNodeB (Base Station) as we introduce in Section 2.1.8. If the TS 29.281 standard, which describes the GTP protocol in detail, is examined in detail, it is seen that an information header containing tunnel information is added on the TCP/IP protocol stacks. Thanks to this added GTP stack, data can be transported between two endpoints with a single (Tunnel Endpoint Identifier) identifier. With the help of this approach, routing can be accelerated by matching a single identifier regardless of the content of the data being transported.

In GTP Implementation, first, we construct the GTP headers, and these headers are shown in Fig. 3.2. In addition, as illustrated in Fig. 2.4, the PDU Session container with QFI information constitutes the extension part of the GTP header. Hence, the GTP extension header will allow the data to be routed according to different QoS according to the QFI indicator.

```
struct ethernet_hdr*    ethernet_frame;
struct ip*              underlay_ip;
struct udp_hdr*         gtp_udp;
struct gtp_hdr*         gtp;
struct gtp_ext*         gtp_ext;
struct gtp_ext_hdr*     gtp_ext_hdr;
struct ip*              overlay_ip;
struct udp_hdr*         udp;
char                    payload[10];
```

Figure 3.2: GTP Headers

3.3 Software Architecture for User Plane

Here, we would like to define the term *flow* as we use in this thesis as it is used in two different contexts.

A flow is a group of packets that matches a PDR. The packets that belong to the same flow are processed by the UPF by executing the same actions. Furthermore, flow is a 64-bit aligned data structure in the OvS library that stores all header fields of a packet.

We propose and implement a UPF architecture that maximizes the packet processing rate and minimizes the packet latency in the user (data) plane. To prevent the processing speed loss that may occur due to the large number of fields that need to be matched in the packet, we design a "miniflow" structure that encompasses 5G protocol details. Miniflow is a compact representation of packet header fields and actions. DPDK has a detailed flow data structure, and the flow structure is explained in [34].

Here we note that the packet fields that need to be matched are often significantly fewer than the "Don't care" fields.

```
struct miniflow {

    struct flowmap map;
    Followed by:

    uint64_t values[n];   where 'n' is miniflow_n_values(miniflow).
};
```

Figure 3.3: Miniflow Struct

The miniflow struct is shown in Fig. 3.3, in this struct one bit is held for each uint64 value. For each bit that is one, the value that should be the corresponding one is added to the end of the miniflow in sequence. The rule match tables are cached in the UP data path as described in Section 2.2.1.2. The miniflow structure saves memory usage, and the number of cache lines accessed is significantly reduced. A miniflow is created for each PDR added to the tables, and each packet arriving at the UPF is matched against the miniflows in the cache. Miniflow map construction is further elaborated in Fig. 3.4, in the context of the miniflow data structure, the map member plays a crucial role by holding one bit for each uint64 element within the detailed flow struct [34]. These bits serve as indicators, where a 0-bit signifies that the corresponding uint64 element in the flow struct holds a value of zero, while a 1-bit indicates that it may contain a non-zero value. Once the map is constructed, a subsequent step involves examining each corresponding non-zero bit in the map. For every non-zero bit, the original uint64 value from the 64-bit aligned flow struct is appended to the miniflow struct. As an example, the network protocol value that corresponds to the 9th index of the bit map array ([8]) in Fig. 3.4 is appended. The collection of these appended values constitutes the values array, which plays a crucial role in representing and managing non-zero data elements within the miniflow structure.

We present the UPF architecture that is implemented in software with DPDK, Poll Mode Driver (PMD) in Fig. 3.5. In our design, incoming data traffic is processed by PMD threads assigned to the processors and is taken out of the system by taking actions. One PMD thread is assigned to the uplink direction, similarly, one PMD thread is assigned to the downlink direction traffic. According to 3GPP standards, each packet arriving at the UPF can have three different exit paths. These paths are "Access (base station direction)," "CP-Function (SMF direction)," and "Core (data networks direction)." After each PMD receives the incoming traffic, it creates the miniflow structure from its header fields, searches for a match in the flow tables using this structure, and takes the necessary actions (URR, BAR, QER, etc.) as a result of the match found and forwards it to the specified output port in the FAR. Since OvS processes a batch of input packets, the packet can be deleted without being forwarded at all as per the rules, it can be stored to be sent later, or even an existing header can be deleted and replaced with a new one [35]. All the activities that are the responsibilities

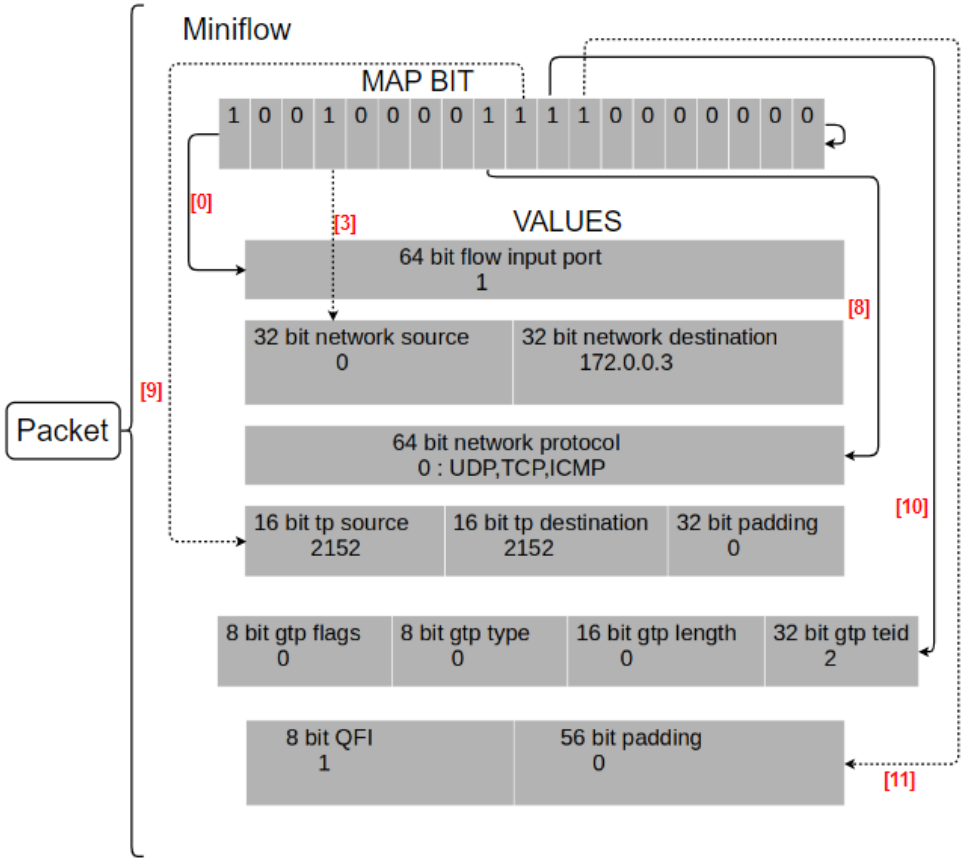

Figure 3.4: Miniflow Map Example for Packets from UE to DN

of the user plane are illustrated in Fig. 3.5.

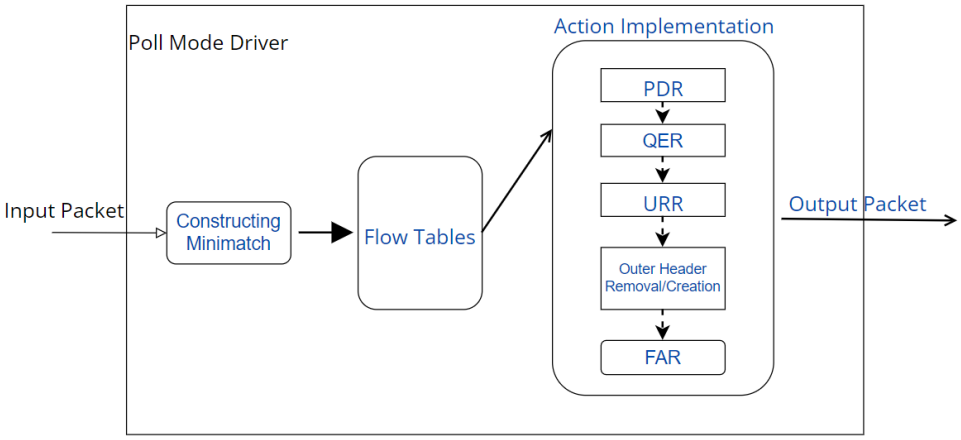

Figure 3.5: 5G User Plane Architecture

The most significant factor affecting the speed of packet data processing in the data plane is the rapid miniflow construction and matching when packets enter the system. Understanding which rule corresponds to each incoming packet in a large flow table (Match-Action Table) can be time-consuming. Even in cases where the flow table is not excessively large but still reasonably small, the multitude of fields that need to be matched within each packet significantly influences the data processing speed. We chose not to design flow tables as a single large rule table to overcome the initial challenge. Instead, they are hierarchically structured to process packet data, and exact match cache (EMC) is the smallest table in Fig. 3.6.

Therefore, the worst-case complexity is O(N) with "N" entries. subtables, the worst-case complexity is O(N) and much of the overhead is in hash computation.

EMC is implemented as a hash table requiring an exact match on the entire miniflow structure. The hash value can be computed in software. After miniflow extraction and hash computation, the arriving packet is first checked in EMC for a match with the help of DPDK `emc_processing` function [36]. If a match is found, an exact match check is performed against the packet's miniflow. Note that as the name implies, no wildcard match is carried out at this stage. If there's no match, subsequent entries are checked. Therefore, the worst-case complexity is O(N) with "N" entries The packets that cannot be matched in the EMC are forwarded to a larger table that we call the Datapath Classifier. If a matching rule is found in the Datapath Classifier, the corresponding actions are executed, and the packet is sent along the appropriate path. The Datapath Classifier performs wildcard matches, and hash tables are used to implement wildcard matching. When a packet that belongs to wildcard matching entries is received, both EMC and Datapath Classifier experience a miss. However, a matching flow is found in the main flow table, and a learning mechanism caches this flow into the Datapath Classifier in a miniflow format by using `miniflow_extract` function of DPDK. Additionally, the cache and datapath classifier flow tables are built to speed up packet processing and reduce latency in an effective way. These tables use a compact version of a packet's header fields, previously mentioned miniflow structure, to index and lookup operations. By extracting the important fields of the packet, the amount of memory used is reduced, which allows for faster access times and more efficient use of resources. The miniflow key helps identify packet attributes quickly

and accurately, which leads to quick decision-making and execution during the packet forwarding process. Overall, the cache and datapath classifier flow tables play a significant role in maximizing packet processing performance and minimizing delays in the network environment. A miniflow struct is constructed for each incoming packet with new map and values fields. This created miniflow is compared with the entries of the cache table by using `netdev_flow_key_equal_mf` function of DPDK [37]. Since both entries and miniflow are generated by using `miniflow_extract` function, the comparison is realized bitwise. Thanks to the miniflow approach, the cache lookup is completed in a single operation, which is beneficial since a "struct flow" is fairly large. After finding the matching entry, all the required actions for the current match are executed as illustrated in Fig. 3.7.

However, if no match is found in both the cache and datapath classifier tables, the matching process will be carried out in the main flow table, where all rules are stored. This table performs wild card matches, and it uses precedence value in PDR to prioritize traffic. If a match is found in the main flow table, the appropriate actions are executed, and the packet data is directed out of the system, meanwhile this flow information is also added to the datapath classifier table. The different sizes of the three tables cause varying packet matching speeds, which necessitates the management of the rules they apply. Ultimately, the controller adds flows to the main flow table when packets are not matched to any of the three tables.

The packet traffic over the same flow must be processed with consistent actions. To achieve this, we dynamically add or remove rules from the EMC and Datapath Classifier tables. Our design adds entries to the datapath classifier table for packets that bypassed it, based on statistical information, periodically replicating the same rule in the cache table. The process of statistical addition and deletion involves keeping a count of accesses to each rule. When this counter reaches zero, the rule is removed from the flow tables, or after a specific period (timeout), these added rules will be removed from the tables. This process ensures efficient packet handling and forwarding based on the information stored in the cached, classified, and main flow tables.

These tables are composed of Match-Action entries, where match entries correspond to specific 5G protocol details to be matched, such as GTP TEID and GTP port num-

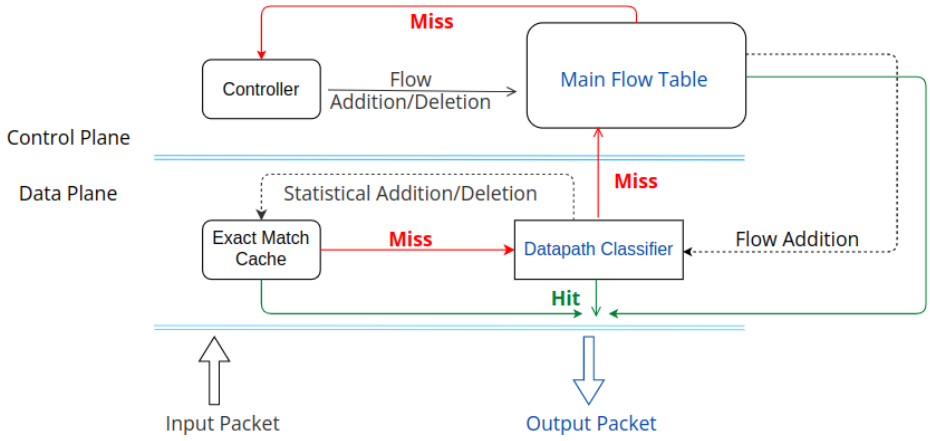

Figure 3.6: Flow Table Construction

ber. Action entries, on the other hand, correspond to 5G rule implementations, such as URR and outer header removal. An example of a match-action table entry for a packet from UE to DN is shown in Fig. 3.7. We extended table entries to meet QoS metrics. We insert QFI to match field, and QER to action field in these tables. Hence, an UL packet with applied QoS metrics is directed to the core network port.

ID	MATCH	ACTION
1 (From UE to DN)	GTP_TEID = 2 GTP_QFI = 1	QER_ID = 1 SEND_CORE_NETWORK_PORT = 2

Figure 3.7: A Match-Action Table Entry

As a result, in our user plane solution, QER is extracted from the PFCP session establishment request message, and the required matching fields such as QFI are added to a main flow table, subsequently, the relevant actions are performed when matching packets enter the system.

Using Ethernet cards that support the DPDK library, incoming traffic is taken into the application by bypassing the Linux kernel so that it can be distributed to the PMD's in the user plane design using different queues based on RSS (Receive Side Scaling). Each PMD starts to process the distributed traffic. The first operation is to create a miniflow from the header fields. Using this miniflow, matching is performed first in

the cache table, and if not found then in the datapath classifier table. Then, the actions for the matched rule are executed on the packet.

Another critical issue to consider in the user plane architecture is the order of prioritization of the actions since the order of the actions to be performed is essential for the possible final state of the packet. The order of these actions is specified in the PFCP protocol and may vary depending on the traffic to be processed. For example, the "Measurement Before QoS Enforcement" flag in URR aims to perform URR measurements before QER rules or IP translation should be performed before applying FAR [23].

3.4 Ingress - Egress Policing and OpenFlow Meter

OvS utilizes Ingress-Egress Policing as a traffic management mechanism to control the rate of incoming and outgoing per-flow traffic on virtual switch ports [38]. Ingress policing regulates the rate of per-flow incoming traffic on a port to prevent excessive traffic from entering the switch, which can help manage network congestion and ensure fair allocation of resources. If the incoming traffic rate exceeds a predefined threshold, excess packets can be dropped, marked, or shaped according to defined policies. Egress policing controls the rate of per-flow outgoing traffic from a port to ensure that the switch does not forward traffic at a rate higher than what the egress interface or network can handle. If the traffic rate exceeds the configured limit, the switch can apply policies to manage the excess traffic, such as dropping or remarking packets.

OpenFlow Meter is a concept within the OpenFlow protocol that measures traffic rates and statistics associated with network flows. OpenFlow Meters monitor and enforce traffic shaping and policing policies and provide the ability to measure traffic rates and statistics to apply traffic policing policies accurately. By configuring OpenFlow Meters, network administrators can define traffic rate limits, burst sizes, and action policies for managing both ingress and egress traffic on OvS ports. This integration allows for dynamic and granular traffic management in virtualized and SDN environments, helping to ensure efficient network utilization and prevent congestion. Hence,

OpenFlow Meter forms the foundation of our QoS management approach.

3.5 QER Action Implementation

As discussed in Section 3.3 OvS batches the packets for each match-action entry in the cache table, and the maximum number of the packets in a batch is defined as 32 [39]. When packet or packet groups hit a QER action, the corresponding function is called with QER id. From this QER id, we can access corresponding IEs from the PFCP QER struct as illustrated in Fig. 3.8.

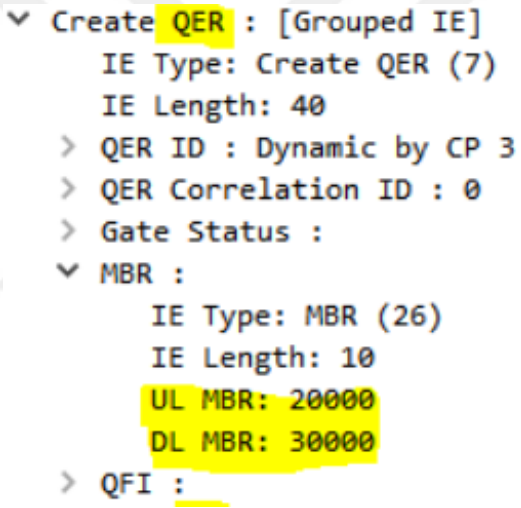

Figure 3.8: PPCP QER Struct

The QFI information in the PFCP QER struct is one of the match entries. The MBR IE in the PFCP QER struct specifies the rate-limiting value to be applied as a result of the matching. The QFI provided by SMF, during the PDU session establishment, and the QFI in the PFCP QER struct have identical values. This QFI mapping allows for rate limiting based on the UL MBR value for packets arriving from the N3 interface and the DL MBR value for packets arriving from the N6 interface. Moreover, units of MBR and GBR fields are specified as kilobits per second in [23].

3.6 Quality of Service Actions

We realize the QoS actions according to the QoS metrics stated by identifying different possible QER messages. According to the QER messages, three different actions must be taken. These are traffic opening/closing (gate control), MBR, and GBR. Traffic opening/closing and maximum bit rate actions are implemented in this thesis. The traffic opening/closing action is more straightforward than the other actions. In this action, the incoming packets are dropped, as in the case where the gate is closed. On the contrary, the traffic is redirected to the corresponding address port if the gate is open. The packet metering principle can be summarized as: following PDR matching, all packets are processed through a single packet processing pipeline without duplication of the packets. Furthermore, a single match-action entry covers all required actions as illustrated in Fig. 3.7. Specific to QER packets can be deleted or modified through outer header creation/removal. Importantly, the entire packet pipeline remains a unified and continuous entity from start to finish. The token-bucket method is chosen and implemented for the maximum data rate action. Token-bucket algorithm is one of the most commonly used methods in rate-limiting design. Another common algorithm in the literature for rate limiters is the Leaky Bucket Algorithm. The main advantage of a token bucket over a leaky bucket is that token bucket can send large burst-size packets more quickly, whereas leaky bucket always sends packets at a constant rate. The implementation of the Token-bucket algorithm in user plane architecture is summarized in the following section.

3.7 Token-Bucket Algorithm for Maximum Bitrate Limiting

The Token Bucket Algorithm (TBA) is illustrated in Fig. 3.9, and it works as follows:

- The bucket is initially full.
- Tokens are generated and added to the bucket at a constant rate of ρ tokens per unit of time.
- When a request comes in, the bucket is checked. If there is enough capacity, the request is allowed to proceed. Otherwise, the request is rejected.

- When a request is allowed, tokens that are proportional to the packet size are removed from the bucket.
- The bucket can hold up to σ tokens. Hence it allows bursts in the shaped traffic.
- If the bucket is empty at the time of the event, the event might be delayed until a token becomes available.

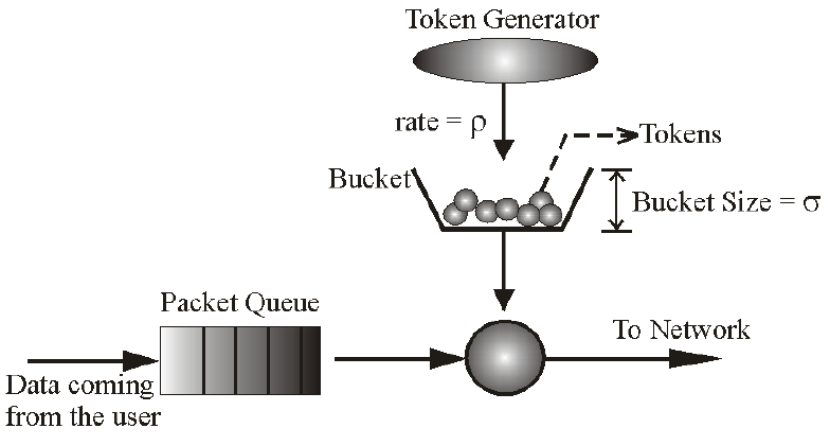

Figure 3.9: Token Bucket Algorithm [3]

We have introduced modifications to the legacy token bucket model. Our primary modification involves moving from a constant token generation rate (ρ) to a time-dependent token generation rate. This adaptation is necessary due to the dedicated core allocated by DPDK for processing incoming packets. As these packets are often batching, we use time difference calculations. Also, to comply with 5G standards, we include QER_MBR in the token generation rate.

The second change concerns the determination of the maximum bucket size (σ). In this calculation, we make use of the OvS parameter $burst\_duration$ (msec) [40] together with the MBR parameter of the QER action corresponding to the flow that we denote with QER_MBR (kbps).

We further define the following parameters, and it is important to note that all parameters that we define and use are per-flow:

- $burst\_size(bits) = burst\_duration(msec) \cdot QER\_MBR(kbps)$

- $bucket\_size(t)(bits)$: The maximum amount of bits of the respective flow that can be allowed to get service at time t.
- $volume$: The number of bits that arrive which are in the flow.
- $volume\_served$: The number of bits that are accepted and served by the token bucket.
- $exceeded\_packet$ is an array that stores index values to indicate the packets that exceed the rate limit.
- t_{now}: The time instant that we calculate the eligible packets for service. The calculation takes place when $volume$ bits of flow packets arrive.
- $t_{MBR_{used}}$: The previous time instant the calculation was carried out.
- t_{max}: Maximum time difference value to make sure that Δt is not too large. It is set 1 second experimentally.

The token bucket update mechanism and marking the exceeding packet are presented in Algorithm 1.

To make the subject more understandable, the token-bucket algorithm can be explained with the following example. In this example, the packet size is kilobits, and `exceeded_packet` array is constructed to store index value to indicate the packets that exceed the rate limit. If there are enough tokens in the bucket to handle one incoming packet, it is necessary to inspect the packets one by one by decreasing the $bucket\_size(t_{now})$ by $volume\_served$. However, if there are not enough tokens even for the first incoming packet, we use the `exceeded_packet` array to mark the subsequent packets arriving in this batch. The maximum number of packets in a burst is defined as 32. The array has a size of 32 to accommodate arriving packet groups of a given flow with a size of 32 packets. Initially:
$\texttt{exceeded\_packet}[0] \cdots \texttt{exceeded\_packet}[31] = FF$.

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & .. & 31 \\ FF & FF & FF & FF & FF & FF & FF \end{bmatrix} \tag{3.1}$$

Initially, `exceeded_packet` array is constructed with negative value as shown in Eq. (3.1). Since the unit of the bucket and volume is bits, meter measurement is realized for bits. As an example, the above condition (if $bucket\_size > volume$) is not satisfied for $bucket\_size$ = 3500 and $volume$ = 5000 such that the bucket can not have enough space for 5000 incoming bits. First, we mark the first exceeding packet as calculated in Eq. from (3.2). In Eq. (3.2), `exceeded_packet` is an index value rather than a packet size information.

$$\texttt{exceeded\_packet} = bucket\_size/1000 \tag{3.2}$$

Additionally, the unit of the rate (MBR and GBR) is kilobits per second, and the unit of the $bucket\_size$ and $volume$ is bits. Hence, $bucket\_size$ is divided by 1000, and `exceeded_packet` is calculated as 3, and the remaining tokens stay in the bucket such that:

$$bucket\_size\% = 1000 \tag{3.3}$$

1. Initialization

$bucket\_size \leftarrow burst\_size$

$N \leftarrow 32$: Maximum number of the packets in a batch

for $i \in 0 \ldots (N-1)$ **do**

  $exceeded\_packet[i] \leftarrow FF$

2. Packet Reception

$volume$ bits arrive for the flow at t_{now}.

$\Delta t \leftarrow t_{now} - t_{MBR_{used}}$ (msec)

Limit Δt, and update $t_{MBR_{used}}$:

$\Delta t \leftarrow max(\Delta t, t_{max})$

$t_{MBR_{used}} \leftarrow t_{now}$

Update the $bucket\_size$ using Eq. 3.4.

$$bucket\_size(t_{now}) = bucket\_size(t_{MBR_{used}}) + \Delta t(msec) \cdot QER\_MBR(kbps) \tag{3.4}$$

Limit the $bucket\_size$:

if $bucket\_size(t_{now}) > burst\_size$ **then**

  $bucket\_size(t_{now}) \leftarrow burst\_size$

3. Packet Accept or Drop

if $bucket\_size(t_{now}) > volume$ **then**

  $volume\_served \leftarrow volume$

else

  for $i \in 0 \ldots (N-1)$ **do**

    if $volume\_served > bucket\_size$ **then**

      $exceeded\_packet[i] \leftarrow 0$

  for $i \in 0 \ldots (N-1)$ **do**

    if $exceeded\_packet[i] = 0$ **then**

      Drop the remaining packets with a total number of bits

      $volume - volume\_served$

Accept the $volume\_served$ bits of packets

Update the $bucket\_size$

$bucket\_size(t_{now}) \leftarrow bucket\_size(t_{now}) - volume\_served$

Algorithm 1: Modified Token Bucket Algorithm Pseudocode

And $bucket\_size$ is calculated as 500 from (3.3).

Calculated `exceeded_packet` = 3.
This value is used for the aforementioned `exceeded_packet`[32] array such that starting from index =3, all negative values are replaced by 0, and it is illustrated in (3.5).

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & .. & 31 \\ FF & FF & FF & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.5}$$

It means that `exceeded_packet` = 3 value is used as an index for this array. These packets are accepted for the index from 0 to 2, index = 3 is marked, and from index = 3 to 31 packets are dropped. In (3.6), accepted packets are illustrated as green; on the other hand, dropped packets are shown as red.

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & .. & 31 \\ FF & FF & FF & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.6}$$

The index value approach is found to be consistent with the initial scenario. It should be noted that the incoming packet size is equal to 5000 bits, and the $bucket\_size$ is 3500 bits. In this approach, 3000 bits are decreased from the bucket, while the remaining 500 bits stay in the bucket. In the index value approach, the first three array elements (ranging from index 0 to 2) are visually represented as green, signifying their acceptance. On the other hand, starting from the 4th array element (from index 3 to 31), subsequent elements are designated for deletion.

In summary, a rate limiter is a counter used to keep track of how many requests are sent from the same user or IP address or to set a maximum speed based on the application. The request is not allowed if the counter exceeds the limit. There are different implementations of how these counters are stored. In-memory caches are preferred due to their fast access; REDIS is an example of a cache implementation [41].

CHAPTER 4

EVALUATION

This section describes the evaluation of UPF rate limiting performance by using the TRex traffic generator. Forwarded packets consist of ICMP Echo Request messages sent from a User Equipment (UE) towards a Data Network (DN). Dropped and processed packet counting is performed through OvS Bridge Statistics. The configuration of UE's in this configuration includes different International Mobile Subscriber Identity (IMSI) numbers, thus facilitating simultaneous performance evaluation across multiple users [14]. This approach consequently allows the evaluation of multiple UE's with different Quality of Service (QoS) configurations. The experiment demonstrates precise rate-limiting capabilities and enables real-time observation of throughput under the conditions of rate-limiting. Real-time statistics are obtained from the TRex tui window that is continuously updated [42].

4.1 UPF Test Environment

Linux network device 2 is equipped with four cores, with two cores allocated to the operating system and the remaining two cores reserved for UPF processing—one for UPF-DL and the other for UPF-UL. The model of each CPU is Intel(R) Atom(TM) CPU C3558, and it contains 8 GB RAM. The working principle of the test scenario is visualized in Fig. 4.1. Overall architecture can be summarized as follows: In this configuration, full-duplex 1Gbps port 0 of NIC 1 and NIC 2 is connected through 10 Gbps ethernet cable. Packets are generated from TRex's port 0 at speeds up to 1Gbps. These generated packets are ICMP Echo Request messages sent from a UE to the DN. ICMP ping messages facilitate the diagnosis of network connectivity issues involving

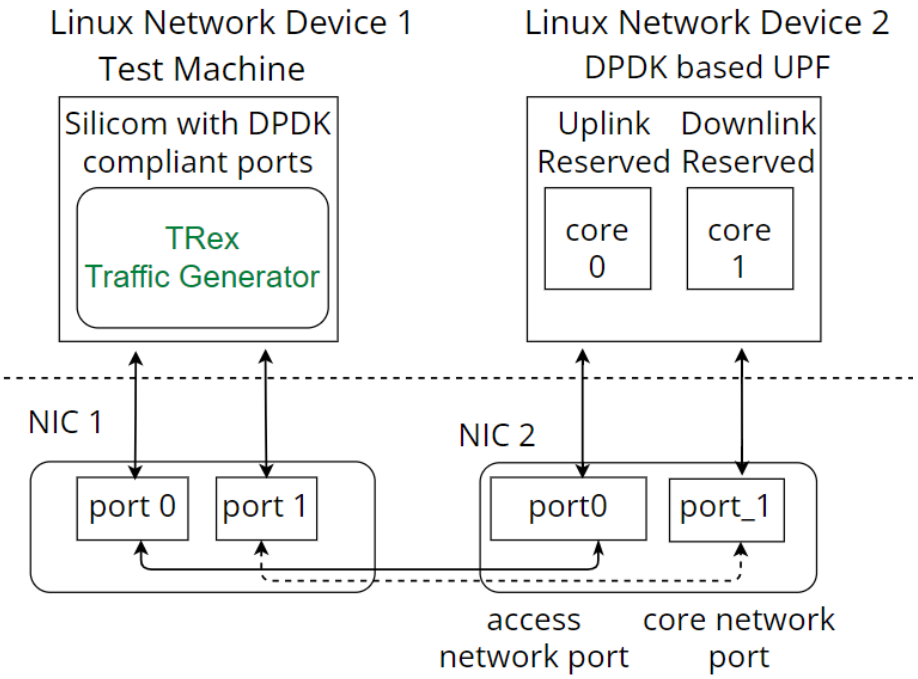

Figure 4.1: Test configuration of UPF

host or network reachability and responsiveness assessments. This process requires transmitting ICMP Echo Request messages to the targeted host and then waiting for an Echo Reply in response. In addition, ICMP ping messages have utility in the area of Packet Loss Detection and Round Trip Time (RTT) measurements. ICMP header has a fixed size of 8 bytes.

Our tests include ICMP Ping request-reply messages. For example, if a 4-byte payload is used in ICMP, the resulting IP packet consists of 76-byte packets. The components of this packet are shown in Fig. 4.2. Due to the 24-byte overhead in the Ethernet frame (7-byte preamble, 1-byte SFD, 4-byte CRC, and 12-byte IPG), generating ICMP packets with only a 4-byte payload results in packet generation rates close to 1 Gbps not being observable in a TRex environment. As a result, tests were performed using ICMP packets with a payload of 910 bytes to achieve a 1 Gbps traffic generation in Silicom Customer Premises Equipment (CPE) hardware [43]. Silicom is a company specialized in providing advanced network and data infrastructure solutions for a wide range of industries. Silicom's solutions integrate DPDK to deliver efficient management of resources and high-performance results. Silicom device is equipped

with eight cores, and the model of each CPU is Intel(R) Atom(TM) CPU C3758, and it contains 16 GB RAM. Since, DPDK enabled ports, in this device, support up to 1 Gbps, rate limiting measurement is conducted by 1 Gbps packet generation rate.

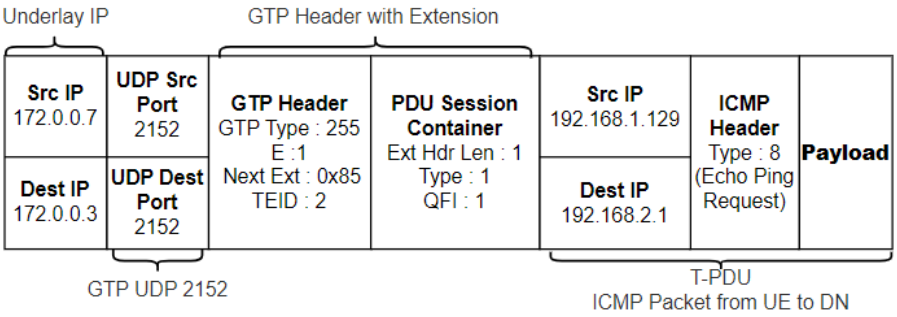

Figure 4.2: ICMP Echo Request Message from UE to DN

The UPF processes these ICMP Echo Request messages in the UL direction. The packet and byte counts of the processed packets in the UL direction can be observed using OvS Bridge Statistics commands. After the outer header removal action is applied, the ICMP packet should be forwarded to the DN. Instead of sending the packets to the modem (DN), we reserved a DPDK port on the Silicom device to respond to ICMP pings. Hence, ICMP ping requests arrive at the Silicom port, and their responses are also forwarded from this port to the UPF. The UPF processes these ICMP Reply responses in the DL direction. Finally, after the UPF processes the ICMP Reply responses, Rx byte measurements are taken from TRex port 0. Therefore, measurements are performed for the DL direction.

The first observation is focused on evaluating the impact of core allocation on packet processing efficiency. For this purpose, a 1 Gbps traffic generation scenario was initially evaluated using a single core for UPF only, and the resulting impact of core allocation on OvS packet processing efficiency is detailed in Table Tab. 4.1. Since a one-core configuration can only handle 63% of the incoming traffic, a two-core configuration is studied, where one core is designated for UPF-DL and the other for UPF-UL processing. Given that the two-core configuration for UPF is capable of processing 99% of the incoming traffic, no further investigation of core allocation was necessary. DPDK isolates CPU cores and optimizes network performance by mitigating the interrupts, as a result, it can handle all incoming traffic. However, the

underlying hardware and operating system still play a role. As an example, some NICs and hardware features may delay or batch interrupts. This can affect the timing and predictability of packet processing. As a result, the end-to-end 5G network experience was conducted using this two-core setup for UPF, covering the evaluation of the following three different scenarios:

Table 4.1: Core Allocation

Number of Cores	Total Generated Packet Number	Processed Packet Number
1	1.200.000	761.721 (63%)
2	1.200.000	1.199.716 (99%)

The initial test was conducted employing TRex to generate 1 Gbps traffic using 3 UE's with 3 distinct flows. A total of 1.200.000 packets were generated per UE, implying a packet generation rate of 300 Mbps per UE. For the 3 UE's, the MBR (Maximum Bit Rate) values were set at 100, 100, and 50 Mbps, respectively, and packets were processed through 3 distinct flows associated with different QER (QoS Enforcement Rule) IDs. Table 4.2 presents the resultant dropped packet counts. For QER IDs 1 and 2, where the rate-limiting was set to 100 Mbps and the generation rate was 300 Mbps, approximately 67% of the packets were dropped, while 33% were processed. Consequently, an effective throughput of 99 Mbps was achieved for QER IDs 1 and 2. Similarly, for QER ID 3, with a rate-limiting of 50 Mbps and a generation rate of 300 Mbps, approximately 83% of the packets were dropped, while 17% were processed. This led to an effective throughput of 51 Mbps for QER ID 3. A comparison between the measured throughput and the configured MBR values is depicted in Fig. 4.3. These measurements affirm the accurate functionality of the rate-limiting algorithm. In addition to these measurements, a real-time evaluation of the aggregated rate-limited throughput was performed for overall flows by monitoring the instantaneous bit rate on the receiver (Rx) side within TRex's real-time update window. Fig. 4.4 illustrates that the rate-limited throughput varies between 240 and 260 Mbps. Given the total rate-limited speed of 250 Mbps, the proper operation of real-time rate limiting is thereby demonstrated.

Secondly, the initial test condition is extended with multiple flows for per UE. For each UE, the destination IP address ranges between 192.168.2.1 - 192.168.6.255. Hence 1275 different IP flows are generated per UE. As a result, a total number of 3.825 (3*1275) flow is tested. Multiple flows per UE configuration are illustrated in Fig. 4.5. Additionally, the resultant dropped packet counts are presented in Table 4.3, and we obtained similar results to Table 4.2.

The final test was conducted using 32 UE's with 40.800 (32*1275) distinct flows. A total of 105.000 packets were generated per UE, implying a packet generation rate of 26 Mbps per UE. For the 32 UE's, the MBR (Maximum Bit Rate) values were set at 10 Mbps for QER IDs 1 - 16, and 20 Mbps for QER IDs 17 - 32. The resultant dropped packet counts are presented in Table 4.4. For QER IDs 1 and 8, where the rate-limiting was set to 10 Mbps and the generation rate was 26 Mbps, approximately 62% of the packets were dropped, while 38% were processed. Consequently, an effective throughput of 9.8 Mbps was achieved for QER IDs 1 and 8. For QER IDs 17 and 32, where the rate-limiting was set to 20 Mbps and the generation rate was 26 Mbps, approximately 24% of the packets were dropped, while 76% were processed This led to an effective throughput of 19.7 Mbps for QER IDs 17 and 32. A comparison between the measured throughput and the configured MBR values is depicted in Fig. 4.6. These measurements confirm the correct functionality of the rate-limiting algorithm for multiple flows. Additionally, for aggregated rate-limited throughput Fig. 4.7 illustrates that the rate-limited throughput varies between 460 and 500 Mbps. Since the aggregated rate-limited speed for 32 UE is 480 Mbps, real-time rate limiting is properly performed.

The averaging window is specified as 2 seconds in 3GPP specifications [44]. Therefore, we performed TRex measurements every 2 seconds. During these measurements, we observed that the aggregated MBR value was occasionally exceeded. We infer that the reason for these exceedances is that the token generation rate is not constant and is calculated proportionally to the MBR and packet arrival times. Similarly, some data points are below the MBR because packets arrive frequently and one at a time in some cases. As a result of frequent packet arrivals, the packet arrival time is small and measurements are below the MBR value. To overcome these deviations, a fixed token generation rate could be used, as in the legacy token bucket. This fixed

rate can be defined by taking into account the averaging window time.

Last but not least, our two-core UPF solution is expected to have a maximum capacity of over 1 Gbps. This expectation is based on measurements made when the Silicom device responded to ping requests with a maximum throughput of 1 Gbps. If the tests are repeated using a product capable of responding at 10 Gbps, the two-core UPF solution is expected to handle even higher speeds. Furthermore, the packet processing efficiency can be improved by approximately 40% by increasing the number of cores, as shown in Tab. 4.1. This suggests that an 8-core UPF solution can handle a throughput of 10 Gbps.

Table 4.2: 3 UE, Total 3 flows

QER ID	MBR(Mbps)	Total Generated Packet Number	Dropped Packet Number
1	100 Mbps	1.200.000	803.146 (67%)
2	100 Mbps	1.200.000	803.244 (67%)
3	50 Mbps	1.200.000	1.001.583 (83%)

Table 4.3: 3 UE, Total 3*1275 flows

QER ID	MBR(Mbps)	Total Generated Packet Number	Dropped Packet Number
1	100 Mbps	1.200.000	803.387 (67%)
2	100 Mbps	1.200.000	803.249 (67%)
3	50 Mbps	1.200.000	1.001.611 (83%)

Table 4.4: 32 UE, Total 32*1275 flows

QER ID	MBR(Mbps)	Total Generated Packet Number	Dropped Packet Number
1	10 Mbps	105.000	65.339 (62%)
8	10 Mbps	105.000	65.338 (62%)
17	20 Mbps	105.000	25.678 (24%)
32	20 Mbps	105.000	25.677 (24%)

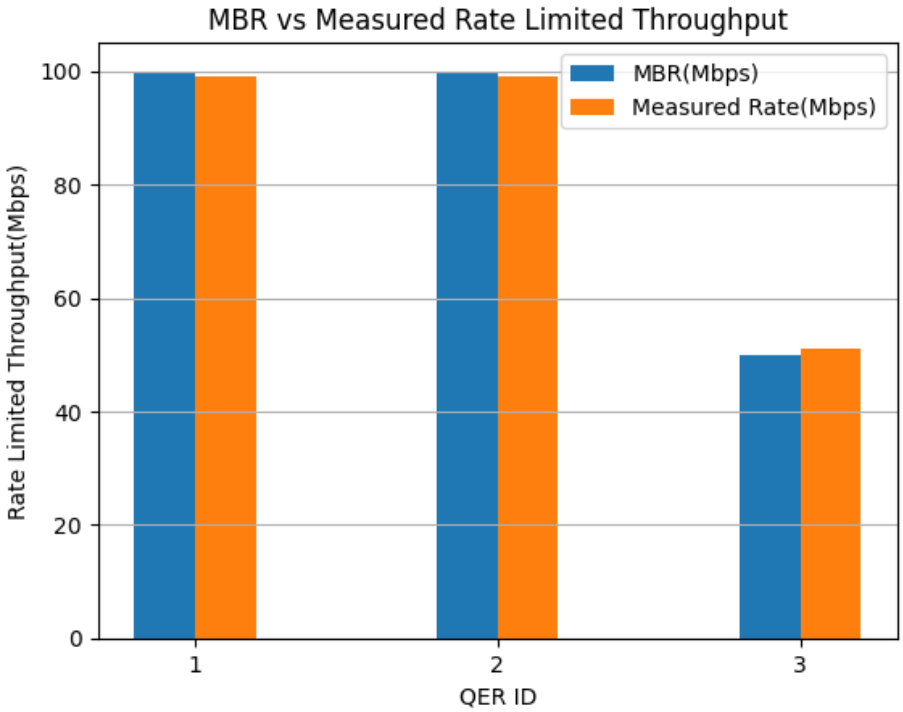

Figure 4.3: 3 UE, Total 3 flows

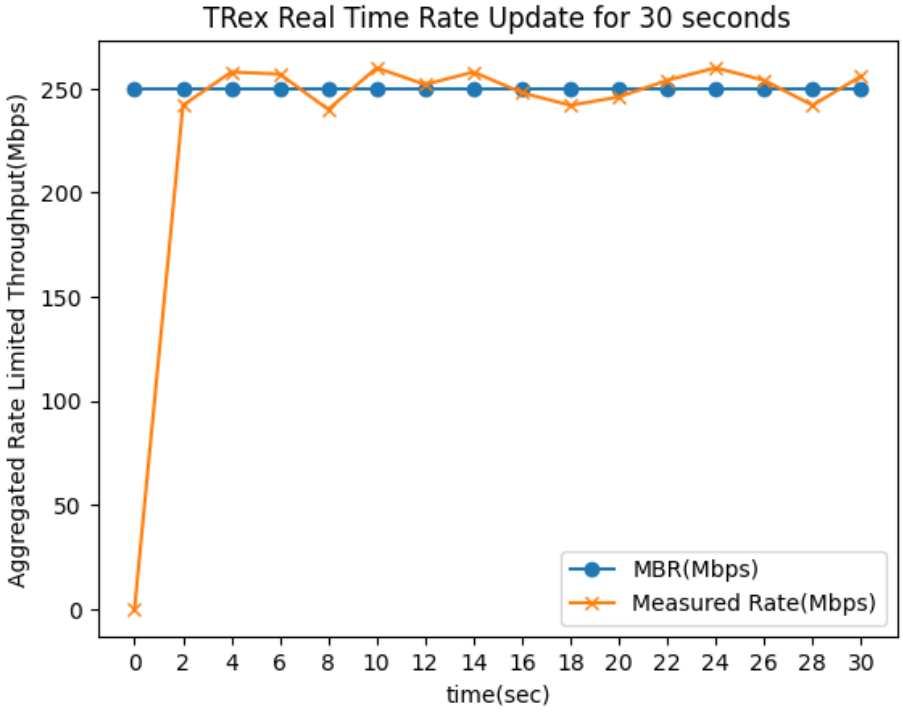

Figure 4.4: 3 UE, Total 3 flows with TRex Real-Time Update

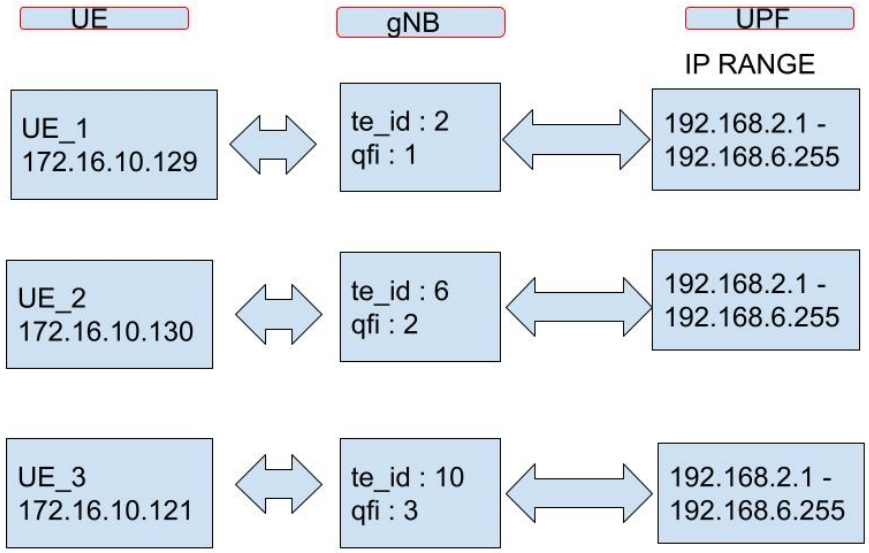

Figure 4.5: 3 UE, Total 3*1275 flows with destination IP Range

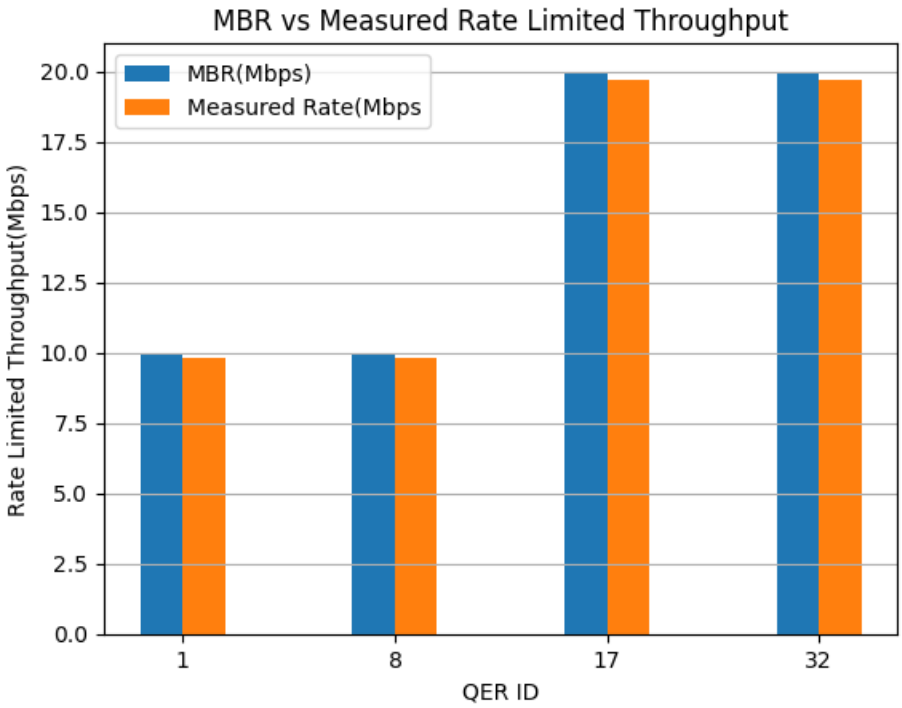

Figure 4.6: 32 UE, Total 32*1275 flows

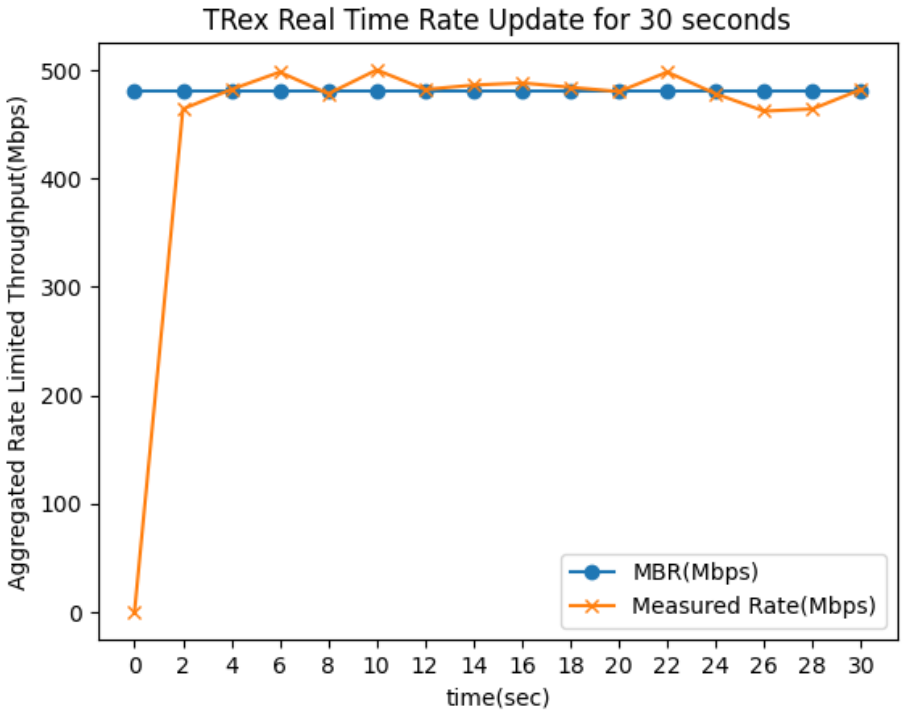

Figure 4.7: 32 UE, Total 32*1275 flows with TRex Real Time Update

CHAPTER 5

CONCLUSION

This thesis proposes implementing and evaluating the User Plane Function module that supports quality of service metrics in 5G networks. The UPF module connects 5G mobile devices and base stations to the Internet and also supports packet counting, usage statistics, and QoS. We have included the QFI field in the GTP tunnels established between gNB and UPF to uniquely identify the appropriate QoS Flow for a specific UE. We also extended the PDR functionality to ensure that UL traffic originating from the UE is correctly mapped to the corresponding QoS Flow by pointing it to the appropriate QER rule.

Our study integrates gate status control and maximum bitrate enforcement features in the QER into UPF's software implementation. For example, we configured the DL Maximum Bit Rate for UEs to be limited to 100 Mbps. As a result, any data traffic coming from the DN that exceeds this threshold is automatically dropped by UPF. This application empowers mobile network operators with the necessary tools to facilitate billing and service delivery based on specific customer payment agreements.

The software implementation of UPF uses OvS, an open-source virtual switch at the data link layer, powered by Intel's DPDK solution. In this integration, we extended flow table entries to accommodate QoS metrics, including adding QFI to the match field and configuring QER to the action field. The token-bucket method is adopted and applied to comply with 3GPP specifications for rate-limiting design. TRex tui window updates the statistics in real-time. Hence, precise rate-limiting capabilities and real-time throughput monitoring under rate-limiting conditions are evaluated using both OvS Bridge Statistics and the TRex traffic generator. Our results demonstrate that the UPF software solution can effectively process 40,800 distinct flows with QoS

support, maintaining a throughput of 1 Gbps. Furthermore, the UPF solution is expected to support a throughput of 10 Gbps with an 8-core device.

Our future work focuses on fulfilling all QoS metrics in 5G networks. In this work, we utilized the ingress policing feature supported by OvS for traffic policing. To realize the software implementation of the GBR information element, it is essential to use the principles of egress policing for traffic shaping. Ensuring a minimum guaranteed bandwidth requires that packets are kept in queuing structures and processed according to the configured rate. Similarly, the correct functionality of the GBR implementation can be demonstrated using OvS Bridge Statistics and the TRex traffic generator.

REFERENCES

[1] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, *et al.*, "The design and implementation of Open vSwitch," in *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, pp. 117–130, 2015.

[2] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan, "Chapter 9 - quality-of-service," in *5G Core Networks*, pp. 204–206, Academic Press, 2020.

[3] F. H. Faheem and A. Mehmood, "Performance analysis of a token bucket shaper for different communication streams,"

[4] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan, "Chapter 3 - architecture overview," in *5G Core Networks*, p. 15, Academic Press, 2020.

[5] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan, "Chapter 3 - architecture overview," in *5G Core Networks*, pp. 19–21, Academic Press, 2020.

[6] R. Botez, J. Costa-Requena, I.-A. Ivanciu, V. Strautiu, and V. Dobrota, "SDN-based network slicing mechanism for a scalable 4G/5G core network: A kubernetes approach," *Sensors*, vol. 21, no. 11, p. 3773, 2021.

[7] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan, "Chapter 6 - session management," in *5G Core Networks*, pp. 121–122, Academic Press, 2020.

[8] 3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane," Tech. Rep. 29.281, 3rd Generation Partnership Project (3GPP), 2022. Version 17.2.0.

[9] E. Dahlman, S. Parkvall, and J. Sköld, "Chapter 20 - beyond the first release of 5G," in *5G NR: the Next Generation Wireless Access Technology* (E. Dahlman, S. Parkvall, and J. Sköld, eds.), pp. 407–414, Academic Press, 2018.

[10] "5G core part 1 — architecture overview." https://derekcheung.medium.com/5g-core-part-1-architecture-overview-6d29160c5c0e, Derek Cheung, Accessed: 2020-07-22.

[11] S. S. Alshamrani, N. Jha, and D. Prashar, "B5G ultrareliable low latency networks for efficient secure autonomous and smart internet of vehicles," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–15, 2021.

[12] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems (TOCS)*, vol. 18, no. 3, pp. 263–297, 2000.

[13] B. Pinczel, D. Géhberger, Z. Turányi, and B. Formanek, "Towards high performance packet processing for 5G," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network*, pp. 67–73, IEEE, 2015.

[14] J. Costa-Requena, A. Poutanen, S. Vural, G. Kamel, C. Clark, and S. K. Roy, "SDN-based UPF for mobile backhaul network slicing," in *2018 European Conference on Networks and Communications (EuCNC)*, pp. 48–53, 2018.

[15] "Ovs - Open vSwitch." `http://www.openvswitch.org/`. (Accessed on 23/07/2023).

[16] H. Zhang, Z. Chen, and Y. Yuan, "High-performance UPF design based on DPDK," in *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, pp. 349–354, IEEE, 2021.

[17] R. Rajesh, K. B. Ramia, and M. Kulkarni, "Integration of LwIP stack over Intel(r) DPDK for high throughput packet delivery to applications," in *2014 Fifth International Symposium on Electronic System Design*, pp. 130–134, 2014.

[18] P. Krongbaramee and Y. Somchit, "Implementation of SDN stateful firewall on data plane using Open vSwitch," in *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pp. 1–5, 2018.

[19] ONF, "OpenFlow switch specification," tech. rep., Open Networking Foundation, 2012. Version 1.3.1.

[20] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "Opennetmon: Network monitoring in OpenFlow software-defined networks," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, pp. 1–8, 2014.

[21] P. R. Srivastava and S. Saurav, "Networking agent for overlay l2 routing and overlay to underlay external networks l3 routing using OpenFlow and Open vSwitch," in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 291–29, 2015.

[22] F. Rezazadeh, H. Chergui, L. Christofi, and C. Verikoukis, "Actor-critic-based learning for zero-touch joint resource and energy control in network slicing," in *ICC 2021 - IEEE International Conference on Communications*, pp. 1–6, 2021.

[23] 3GPP, "Interface between the Control Plane and the User Plane nodes," Technical Specification (TS) 29.244, 3rd Generation Partnership Project (3GPP), 05 2022. Version 17.4.0.

[24] 3GPP, "Management and orchestration; 5G performance measurements," Tech. Rep. 28.552, 3rd Generation Partnership Project (3GPP), 2021. Version 16.10.10.

[25] 3GPP, "Study on scenarios and requirements for next generation access technologies," Tech. Rep. 38.913, 3rd Generation Partnership Project (3GPP), 2022. Version 17.0.0.

[26] J. Rischke, C. Vielhaus, P. Sossalla, J. Wang, and F. H. Fitzek, "Comparison of UPF acceleration technologies and their tail-latency for URLLC," in *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks*, pp. 19–25, IEEE, 2022.

[27] J. Zhou, Z. Ma, W. Tu, X. Qiu, J. Duan, Z. Li, Q. Li, X. Zhang, and W. Li, "Cable: A framework for accelerating 5G UPF based on eBPF," *Computer Networks*, vol. 222, p. 109535, 2023.

[28] T. A. Navarro do Amaral, R. V. Rosa, D. F. C. Moura, and C. Esteve Rothenberg, "Run-time adaptive in-kernel BPF/XDP solution for 5G UPF," *Electronics*, vol. 11, no. 7, p. 1022, 2022.

[29] R. MacDavid, C. Cascone, P. Lin, B. Padmanabhan, A. Thakur, L. Peterson, J. Rexford, and O. Sunay, "A p4-based 5G user plane function," in *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, pp. 162–168, 2021.

[30] A. Güngör, "Ueransim: an open-source 5G UE and gNodeB." `https://github.com/aligungr/UERANSIM`. (Accessed on 30/07/2023).

[31] J. I. Kwon, *An In-Depth Analysis of Open-Source 5G Core Networks*. PhD thesis, University of Colorado at Boulder, 2021.

[32] W.-E. Chen and C. H. Liu, "High-performance user plane function (UPF) for the next generation core networks," *IET Networks*, vol. 9, no. 6, pp. 284–289, 2020.

[33] Cisco, "Trex traffic generator website." `https://trex-tgn.cisco.com/`. (Accessed on 30/07/2023).

[34] "Generic flow." `http://doc.dpdk.org/guides/prog_guide/rte_flow.html#`. (Accessed on 1/09/2023).

[35] "Output packet batching." `https://docs.openvswitch.org/en/latest/intro/install/dpdk/`. (Accessed on 23/07/2023).

[36] "Emc call graph." `https://www.intel.com/content/www/us/en/developer/articles/technical/ovs-dpdk-datapath-classifier-part-2.html`. (Accessed on 1/09/2023).

[37] "dpif-netdev." `https://github.com/libreswitch/openvswitch/blob/master/lib/dpif-netdev.c`. (Accessed on 25/07/2023).

[38] "Quality of Service (QoS)." `https://docs.openvswitch.org/en/latest/faq/qos/`. (Accessed on 24/07/2023).

[39] "DPDK device memory models." https://docs.openvswitch.org/en/latest/topics/dpdk/memory/. (Accessed on 1/08/2023).

[40] "Quality of Service (QoS) rate limiting." https://docs.openvswitch.org/en/latest/howto/qos/. (Accessed on 1/08/2023).

[41] "Introduction to Redis." https://redis.io/docs/about/. (Accessed on 3/08/2023).

[42] Cisco, "Trex console support." https://trex-tgn.cisco.com/trex/doc/trex_console.html. (Accessed on 30/07/2023).

[43] "Cordoba edge gateway cpe." https://www.silicom-usa.com/wp-content/uploads/2021/03/Cordoba-cpe.pdf/. (Accessed on 3/08/2023).

[44] 3GPP, "Typical traffic characteristics of media services on 3GPP networks ," Technical Specification (TS) 26.925, 3rd Generation Partnership Project (3GPP), 11 2020. Version 16.0.0.