TU WIEN Informatics EURECOM OPEN AIR INTERFACE

Robotics and Edge Computing in 5G

A Prototype for the OpenAirInterface 5G System

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering & Internet Computing

by

Stefan Spettel
Registration Number 01526978

to the Faculty of Informatics

at the TU Wien, supported by EURECOM and the OpenAirInterface Software Alliance

Advisor TU Wien:	Thomas Grechenig
Advisor EURECOM:	Florian Kaltenberger
Assistance TU Wien:	Florian Fankhauser
Assistance OSA:	Rohan Kharade

Vienna, 19th January, 2023

Signature Author

Signature Advisor

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Stefan Spettel

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

I hereby declare that I have written this thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Wien, 19. Jänner 2023

Stefan Spettel

Danksagung

An dieser Stelle bedanke ich mich bei all jenen, die mich beim Schreiben dieser Arbeit unterstützt haben.

Zuallerst danke ich meinen akademischen Betreuern Thomas Grechenig von der TU Wien und Florian Kaltenberger von der EURECOM für die fortlaufende Begutachtung, Betreuung und sehr konstruktiven Anregungen und Hilfestellungen. Ich danke auch Florian Fankhauser für seine Geduld und seine äußerst akribische Arbeit, die erheblich zur Qualität dieser Arbeit beigetragen hat.

Weiters gilt mein Dank Rohan Kharade und meinen Kolleg:innen der OpenAirInterface Software Alliance für die tagtägliche technische und fachliche Unterstützung während meines sechsmonatigen Praktikums in Frankreich. Weiters danke ich dem Director of Operations der OpenAirInterface Software Alliance Irfan Ghauri für die Chance, am OpenAirInterface zu arbeiten.

Im Laufe meines Studiums in Wien und Auslandsaufenthalten in Griechenland und Frankreich habe ich unzählige Kolleg:innen und Freund:innen kennengelernt, die mich ermutigt haben, meinen Weg weiterzugehen. Da es mir unmöglich scheint, hier nur Einige zu nennen, danke ich Euch allen herzlichst.

Ich danke auch meinen Eltern und meiner Familie, die mich über meine gesamte Studienzeit stets unterstützt haben. Hier auch ein besonderer Dank an meine Schwester Kathrin für den wissenschaftlichen Input und Richard Kriz für das Korrekturlesen.

Abschließend drücke ich meine Dankbarkeit aus, dass ich in einem politischen System lebe, das meine Bildung unterstützt hat. Die finanzielle Unterstützung durch das Selbsterhalter:innenstipendium und die Erasmus+ Stipendien haben es mir ermöglicht, mich auf mein Studium zu fokussieren und dabei wertvolle Auslandserfahrung zu sammeln, ohne die diese Arbeit so nie entstanden wäre.

Acknowledgements

At this point I am thanking all those who supported me in writing this thesis.

First of all, I am thanking my academic supervisors Thomas Grechenig from TU Wien and Florian Kaltenberger from EURECOM for their ongoing review, supervision and very constructive suggestions and assistance. I am also thanking Florian Fankhauser for his patience and extremely meticulous work, which contributed significantly to the quality of this thesis.

Furthermore, I am thanking Rohan Kharade and my colleagues at the OpenAirInterface Software Alliance for their daily technical and professional support during my six-month internship in France. Furthermore, I am thanking the director of operations of the OpenAirInterface Software Alliance Irfan Ghauri for the chance to work on the OpenAirInterface.

During my studies in Vienna and stays abroad in Greece and France I met countless colleagues and friends who encouraged me to continue on my way. Since it seems impossible to name only a few, I am thanking all of you very much.

I am also thanking my parents and my family, who have always supported me throughout my studies. Here also a special thanks to my sister Kathrin for the scientific input and Richard Kriz for proofreading.

Finally, I am expressing my gratitude for living in a political system that has supported my education. The financial support through the Selbsterhalter:innenstipendium and Erasmus+ scholarships have allowed me to focus on my studies while gaining valuable experience abroad, without which this thesis would never have been possible in this way.

Kurzfassung

Der neue Mobilfunkstandard 5G bietet viele Verbesserungen im Vergleich zu seinem Vorgänger 4G. Während aus Sicht der Endverbraucher:innen die Erhöhung der Bandbreite am wichtigsten ist, ist der 5G Anwendungsfall Ultra-Reliable Low Latency Communications (URLLC) für das Internet der Dinge (IoT) am wichtigsten. Das IoT besteht aus vernetzten Geräten und Sensoren, die untereinander und mit zentralen Rechnern Daten austauschen. Häufig darf die Kommunikation zwischen den Sensoren und einem zentralen Rechner eine geringe Latenzzeit, z. B. 10 ms, nicht überschreiten. Während URLLC die Latenzzeit in 5G reduziert, kann die Verbindung zu einem weit entfernten zentralen Rechner erhebliche Latenzzeiten verursachen. Eine Lösung zur Behebung dieses Problems besteht darin, den zentralen Rechner physisch so nah wie möglich an das 5G Endgerät zu bringen. Dies ist das Hauptkonzept von Edge Computing in 5G.

Da ein mobiler Roboter als ein oder mehrere „Dinge“ im IoT eingestuft werden kann, besitzt er ähnliche Anforderungen. Wenn ein Roboter sich zum Beispiel in einer bekannten oder unbekannten Umgebung lokalisiert, stützt er sich auf Light Detection and Ranging (LIDAR) Scans, um die Welt zu erfassen. Dieses Verfahren wird als Simultaneous Localization and Mapping (SLAM) bezeichnet. Es ist von entscheidender Bedeutung, dass die Messungen des LIDAR-Sensors mit geringer Latenzzeit verarbeitet werden, da Verzögerungen die Genauigkeit der Lokalisierung beeinträchtigen können. SLAM ist jedoch auch sehr rechenintensiv, weshalb die Auslagerung dessen sehr vorteilhaft und manchmal auch notwendig ist, da Roboter oft an Hardwaregrenzen stoßen.

Die zugrundeliegende Hypothese dieser Arbeit ist, dass es möglich ist, einen SLAM-Algorithmus an den Rand eines 5G-Netzwerks zu verlagern, ohne die Funktionalität und Qualität der Lokalisierung zu beeinträchtigen. Um dies zu erreichen, werden bestehende Ansätze von Edge Computing in 5G analysiert und verglichen. Darüber hinaus wird die quelloffene 5G Implementierung OpenAirInterface angepasst, um den bis dahin nicht berücksichtigen Anwendungsfall des Edge Computings zu unterstützen. Schließlich wird ein Experiment durchgeführt, das die Genauigkeit und Funktionalität des SLAM-Algorithmus in drei verschiedenen Einsatzgebieten vergleicht: Lokal, WiFi und 5G Edge.

Die Ergebnisse dieser Arbeit zeigen, dass eine Berechnung von SLAM am 5G-Edge möglich ist, wobei die Genauigkeit und Funktionalität der Lokalisierung und der erstellten Karte im Vergleich zur WiFi-Option etwas geringer sind. Obwohl vieles noch optimiert werden kann, legt diese Arbeit den Grundstein für 5G-fähige mobile Robotik, vollständig mit Open-Source-Software betrieben.

Keywords: *5G, Edge Computing, MEC, Robotik, ROS*

Abstract

The novel mobile telecommunications standard 5G offers many improvements compared to its predecessor 4G. While from an end consumer perspective, the increase in bandwidth and download speed are the most crucial, the 5G use case Ultra-Reliable Low Latency Communications (URLLC) is the most important for the Internet of Things (IoT). The IoT consists of connected devices and sensors that exchange data with each other and central nodes. Often, the communication between the sensors and a central node must not exceed a low latency, e.g., 10ms. While URLLC reduces the latency in 5G, the link to a distant central node may introduce substantial latencies. A solution to solve this issue is to physically move the central node as close as possible to the connected 5G device. This is the main concept behind edge computing in 5G.

As a mobile robot can be classified as one or more "things" in the IoT, it has similar requirements. For example, when a robot is localizing itself in a known or unknown environment, it relies on Light Detection and Ranging (LIDAR) scans to map the world. This procedure is called Simultaneous Localization and Mapping (SLAM). It is crucial that the measurements from the LIDAR sensor are processed with low latency, as delays can compromise the accuracy of the localization. However, Simultaneous Localization and Mapping (SLAM) is also computationally expensive. Therefore, it is highly desired and sometimes required to offload this computation to another computer, as robots are often hardware-constrained.

The underlying hypothesis of this work is that it is possible to offload a SLAM algorithm to the edge of a 5G network without compromising functionality and quality of the localization. To achieve this, existing approaches to edge computing in 5G are analyzed and compared. Further, the open source 5G implementation OpenAirInterface is adapted to support the edge computing use case, which has not been considered in its initial design. Finally, an experiment is conducted which compares the accuracy and functionality of the SLAM algorithm in three different deployments: Local, WiFi and 5G edge.

The results of this thesis show that offloading SLAM to the 5G edge is possible, whereas the accuracy and functionality of the localization and the produced map are slightly lower compared to the WiFi deployment. While much can still be optimized, this work lays the foundation for 5G-enabled mobile robotics, fully operated with open source software.

Keywords: *5G, Edge Computing, MEC, Robotics, ROS*

Résumé

La nouvelle norme de télécommunications mobiles 5G offre de nombreuses améliorations par rapport à son prédécesseur 4G. Si l'augmentation de la bande passante et de la vitesse de téléchargement sont plus cruciales pour les consommateurs finaux, le cas d'utilisation 5G Ultra-Reliable Low Latency Communications (URLLC) est plus important pour l'Internet des Objets (IdO). L'IdO se compose d'appareils et de capteurs connectés qui échangent des données entre eux et avec des ordinateurs centraux. Souvent, la communication entre les capteurs et un ordinateur ne doit pas dépasser une faible latence, par exemple 10 ms. Bien que l'URLLC réduise la latence en 5G, la connexion à un ordinateur central distant peut entraîner des latences substantielles. Une solution pour résoudre ce problème est de déplacer physiquement l'ordinateur aussi près que possible de l'appareil 5G connecté. C'est le concept principal de l'informatique en périphérie.

Comme un robot mobile peut être considéré comme un ou plusieurs « objets » à l'IdO, il a des exigences similaires. Par exemple, lorsqu'un robot se localise dans un environnement connu ou inconnu, il utilise des scans de Light Detection and Ranging (LIDAR) pour cartographier le monde. Cette procédure est appelée Simultaneous Localization and Mapping (SLAM). Il est essentiel que les mesures du capteur LIDAR soient traitées avec une faible latence, car les retards peuvent compromettre la précision de la localisation. Cependant, le SLAM est coûteux en termes de calcul. Par conséquent, il est fortement souhaité et parfois nécessaire de décharger de ce calcul sur un autre ordinateur, car les robots sont souvent limités par le matériel.

L'hypothèse sous-jacente de ce travail est qu'il est possible de décharger un algorithme SLAM à la périphérie d'un réseau 5G sans compromettre la fonctionnalité et la qualité de la localisation. Pour y parvenir, les approches existantes de l'informatique périphérique dans l'architecture du réseau 5G sont analysées et comparées. En outre, l'implémentation logicielle libre 5G OpenAirInterface est adaptée pour soutenir le cas d'utilisation de l'informatique en périphérie, qui n'a pas encore été pris en compte. Enfin, une expérience est menée pour comparer la précision et la fonctionnalité de l'algorithme SLAM dans trois déploiements différents : Local, WiFi et 5G edge.

Les résultats de cette thèse montrent qu'il est possible de décharger le SLAM sur la périphérie 5G, alors que la précision et la fonctionnalité de la localisation et de la carte produite sont légèrement inférieures par rapport au déploiement WiFi. Même si beaucoup de choses peuvent encore être optimisées, ce travail jette les bases d'une robotique mobile compatible avec la 5G, entièrement avec des logiciels libres.

Keywords: *5G, Informatique en périphérie, MEC, Robotique, ROS*

Contents

CHAPTER 1

Introduction

The 5th generation of cellular broadband networks (5G) has already been commercially launched in most European countries according to 5gobservatory.eu.[W1] One of the most prominent features of 5G for the average user is the increase of experienced data rates by a factor of 10 compared to 4G, as defined by the European Telecommunications Standardization Institute (ETSI)[W2]. Although this is substantial, the promises of 5G lay far beyond higher download speeds and better connectivity. According to Pham et al. [1], edge computing is a key technology in 5G.

1.1 Problem Statement

Compared to the previous standard, 5G delivers an increase in performance, especially in the areas Enhanced Mobile Broadband (eMBB), Massive Machine-type Communications (mMTC) and Ultra-Reliable Low Latency Communications (URLLC). Even though the bandwidth of mobile broadband services are important from a consumer's perspective, 5G also enables many more scenarios such as interconnected factories and robotics. In this application of URLLC the available bandwidth is not as important as the latency and the reliability. The reality of current mobile operator's 5G deployments is that they are focused on eMBB as an added value for mobile data subscriptions. Successfully adding URLLC to the 5G networks requires lower latency and higher flexibility on the radio interface, the Radio Access Network (RAN). Additionally – and this is the focus of this thesis – edge computing is needed.

Edge computing has been introduced to overcome latency concerns in cloud computing, as described by Hassan et al.[2]. In cloud computing, computation-intensive tasks are executed in data centers instead of locally on a device. This provides an efficient way to process large amounts of data. The downside to this approach is that distant data centers introduce substantial latencies. ETSI's target for URLLC in 5G networks is to have latencies of 1 ms and below.[W2] This goal cannot be met with cloud computing

and thus edge computing is introduced. Its aim is to have data centers in proximity to the devices, as described by Shi and Dustdar[3]. One way to realize edge computing is the ETSI Multi-Access Edge Computing (MEC) architecture[4]. MEC allows to deploy virtualized instances in or next to mobile operator's existing equipment such as base stations and radio controllers.[2]

The ETSI MEC framework has been introduced in 2014, in a time where 4G networks were being deployed and 5G was still far on the horizon. Therefore, MEC does not cover how exactly the underlying mobile system is configured to allow edge computing, but focuses on the orchestration and management aspects of edge computing instead. This has led to several different implementations to enable the same goal: Route traffic to the nearest edge data center. These approaches may or may not require interaction with the operator's existing mobile equipment and, therefore, do not scale well on a global scale. Seeing this gap, the 3rd Generation Partnership Project (3GPP), which standardizes the 5G System (5GS), designed 5G with edge computing in mind from the beginning. It also specified its own architecture, called EDGEAPP, to enable edge computing.[5]

Using the standardized approach for local traffic routing requires a specific 5G network deployment, the Stand-Alone (SA) deployment. This comes with a new set of services to handle user sessions, called the 5G Core (5GC). The majority of current 5G networks use the Non Stand-Alone (NSA) deployment, which reuses the packet core of the 4G network and inherits its limited functionality. Even though the faster, more efficient base stations and communication protocols are used towards the user, the 4G core does not have built-in edge computing functionalities. The new SA 5G network is currently being deployed and according to a worldwide survey released by Enea in 2020, 37% of operators plan to have an SA deployment in 2022.[W3] Given the expected increase of SA deployments, there is the need for a working prototype of 5G enabled edge computing as a basis for further research and industrial developments.

1.2 Motivation

When 5G and edge computing are covered in the news, one could assume that the Industry 4.0, autonomous connected robots and remote medical procedures using a robotic arm are already state of the art. In reality, although solutions exist, these applications are not yet widespread (see also Masood and Sonntag[6]). As outlined, edge computing is a hard requirement for most of these futuristic visions. However, current edge computing solutions in the context of mobile networks are tailored towards a specific operator or a specific vendor of networking equipment. This is because the ETSI MEC standard did not specify how this interaction needs to look like. Of course, commercial offers already exist, such as AWS Wavelength, which integrates 5G edge computing with the AWS cloud service.[W4] However, even Wavelength, driven by the most prominent cloud operator, is currently only available for some distinct regions of specific mobile operators.

One of the design principles behind 5G is that third party developers are able to interact with the 5GC and add services on top of the mobile network infrastructure, as described

by Yang[7]. This is in contrast to Over-the-Top (OTT) vendors, which use the mobile network only as access to the internet. Prominent examples of OTT services are Zoom, Whatsapp and Facebook and these directly compete with the telecom operators, as described by Farooq and Valliappan[8]. The openness of 5G together with the concept of network slicing allows operators to offer services such as Quality of Service (QoS) to the OTT providers as an additional source of revenue.[8]

Third party applications cannot only control QoS, but also influence traffic routing decisions in the 5GS. This is the entry point for any edge enabler platform, as influencing the routing is a key requirement for edge computing. In contrast to the already existing solutions for ETSI MEC, this does not require a proprietary interface between the 5GC and the edge platform. In reality, a proprietary interface often means a vendor lock-in for both of these components.

To demonstrate the usability of a standardized solution, the OpenAirInterface (OAI) is used. The OAI is an open source 5G implementation and consists of the RAN and the 5GC. It provides a standard-compliant reference implementation of a 5G SA deployment and is used for industrial and research purposes, as described by Kaltenberger et al.[9]. One of the goals of the OAI is to provide 5G networking components for the community, which should offer alternatives to expensive commercial vendor solutions. There are already two MEC prototypes for the OAI, which are described in Chapter 3. However, both of these solutions do not use the novel standardized approach. Thus, the current implementation of the OAI does not support this traffic routing approach. To have a reliable reference for 3GPP-compliant edge computing, the OAI should be enhanced with this functionality.

Rao and Prasad[10] describe the vital role of 5G in the enablement of the Industry 4.0 and one of its applications: autonomous robotics. Given that there are not many edge computing implementations available for the open source and research community, there are to the author's best knowledge no reliable use cases on robotics in an end-to-end 5G enabled edge computing scenario. As autonomous robots which are connected over 5G are an important part of the Industry 4.0, it is necessary to showcase the edge computing possibilities with a robotics scenario. An existing implementation based on open source software allows researchers to evaluate novel approaches to algorithms in robotics in a realistic scenario.

1.3 Expected Results

A working edge computing prototype, integrated in an existing 5GS such as the OAI, allows to deploy any kind of application in the edge. One of these applications could be a computation-intensive task of an autonomous robot such as SLAM. SLAM is used to localize the robot and create a map of its environment based on the robot's sensors, especially Light Detection and Ranging (LIDAR). Modern robot software such as the Robot Operating System 2 (ROS2) allows to separate different components and have them communicate over the network. In current ROS2 deployments, these tasks are often

offloaded over WiFi. The underlying hypothesis of this thesis is that this is also possible using edge computing in 5G.

Hypothesis H0: MEC in a 5G system allows to offload latency-critical robotic applications to the edge without compromising the functionality and the quality of the output.

Based on the *H0* hypothesis, the author formulates the following research questions:

- **Research Question 1 (RQ1)**: What is a suitable edge computing architecture for offloading latency-critical applications to the edge of a 5G network?
- **Research Question 2 (RQ2)**: What is a feasible approach to implement a MEC prototype and integrate it into an existing 5G system such as the OAI?
- **Research Question 3 (RQ3)**: Given a MEC prototype and a ROS2 SLAM node, how does offloading the node to the edge of a 5G network affect the latency and the quality and functionality of the produced output?

The expected result related to *RQ1* is an analysis of existing frameworks and architectures for edge computing and MEC in 5G. The analysis consists of three parts: (1) Which architectures for edge computing are available, (2) how do the possible technical realizations within the 5GS look like and (3) how do the edge computing platform and the 5GS interact. These approaches are compared based on maturity, standard-compliance and how well they integrate into 5G and the field of robotics. This is described in detail in Chapter 4.

The answer for *RQ2* is a prototype implementation of edge computing integrated into the OAI 5GS. The details of the integration within the OAI are described in Chapter 5. The chosen solution for this thesis relies on standardized interfaces within and towards the 5GC. This means that the User Plane Function (UPF) is used for traffic routing and the decision to enforce the routing is done using the Session Management Function (SMF), together with the policies from the Policy Control Function (PCF). Prior to this thesis, the PCF did not exist in the OAI and has been implemented to satisfy the needs for traffic routing. The SMF has been adapted as well to support this scenario. While the edge computing related procedures are implemented within the OAI 5GC as part of this thesis, the orchestration and management layer is not provided. However, an integration with the orchestration layer is considered in the design of the PCF.

An experiment is set up to answer *RQ3*. A robot simulation is created based on ROS2, as described in the ROS2 documentation[W5]. ROS2 is a framework for communication between different independent parts of the robot, called nodes. ROS2 nodes can be local or connected via the network in a peer-to-peer or client-server fashion. This allows to deploy ROS2 nodes anywhere in a distributed system, as long as there is a network connection. For the experiment, different deployment options are used and compared. In each of these options, a ROS2 SLAM node is deployed in another location, while the simulated environment and robot stays at the same location.

1.4 Structure

The thesis is structured in nine chapters. This first chapter is the introduction.

Chapter 2 explains the foundations of this work, such as 5G, edge computing and cloud computing in general, virtualization and ROS2.

In Chapter 3 the related work is discussed, which contains other approaches to MEC in 5G, details about the used SLAM algorithm, ROS2 in the cloud and the evaluation metrics for latency critical applications.

Chapter 4 contains the answer to *RQ1* from the expected results. Different edge computing architectures are described and evaluated. It also contains different approaches on traffic routing for edge computing and describes how the 5GC can be instructed to create these routes.

Chapter 5 gives a brief introduction of the OAI and describes how the traffic routing procedures discussed in Chapter 4 are implemented and integrated into the OAI.

Chapter 6 describes the robotics use case, which is used to exemplify the usability of the implemented edge computing prototype. It contains the underlying scenario, details on the robot simulation and how the nodes are set up with ROS2.

In Chapter 7 the implemented prototype is evaluated against the scenario defined in Chapter 6. It contains information about the setup, the different deployment options and results of the evaluation.

In Chapter 8 the author provides a personal assessment of the presented edge computing prototype and its applicability to the robotics use case.

Chapter 9 contains the conclusion of this thesis and describes potential research topics for future work.

CHAPTER 2

Foundations

This chapter aims to help the reader in understanding the concepts, architectures and solutions discussed in the next chapters. Thus, an overview of the underlying technologies and concepts 5G, edge computing, virtualization and ROS2 is given.

2.1 5G

Chapters 4 and 5 discuss different architectures for edge computing in 5G and how to implement a prototype thereof. This section describes the fundamental design principles behind 5G and gives an overview of the architecture.

2.1.1 About 5G

As described by Cox[11], 5G builds upon many of the principles from the previous mobile network generations. A fundamental difference between 4G and 5G is that 5G targets a wider range of applications. While 4G or Long Term Evolution (LTE) is mainly focused on consumer's applications such as mobile data and mobile voice, 5G also covers the areas of mMTC and URLLC.[11]

The support of URLLC is a basic requirement for the use case presented in this thesis. The flexibility of 5G leads to a more complex architecture, in the RAN, as well as in the 5GC. As discussed by Rommer et al.[12], the requirements for 5G were first described in 2012 by the International Telecommunication Union (ITU). The ITU is the global institution responsible for defining and regulating telecommunications and is a specialized agency of the UN (see also Cowhey[13]). These requirements are formalized in the ITU IMT-2020 vision, the ITU Recommendation M.2083-0[14], as part of several IMT-2020 standards that specify the next mobile network 5G. Figure 2.1 shows these and compares them with the 4G requirements (IMT-advanced), and Figure 2.2 shows which requirements are important for eMBB, mMTC and URLLC. It can be seen that the latency shall

be reduced from 10 ms in 4G to 1 ms in a 5G URLLC scenario. The user experienced data rate shall improve from 10 Mbit/s in 4G to 100 Mbit/s. The spectrum shall be utilized three times more efficiently and the maximum mobility speed should increase from 350 km/h to up to 500 km/h. This is especially useful in a scenario where devices are roaming while in a high-speed train. Another aspect to be improved in 5G is the connection density, from 100 thousand devices to one million per square kilometer.[14] 5G also improves the network energy efficiency, an aspect that is crucial to support battery-constrained Internet of Things (IoT) devices.

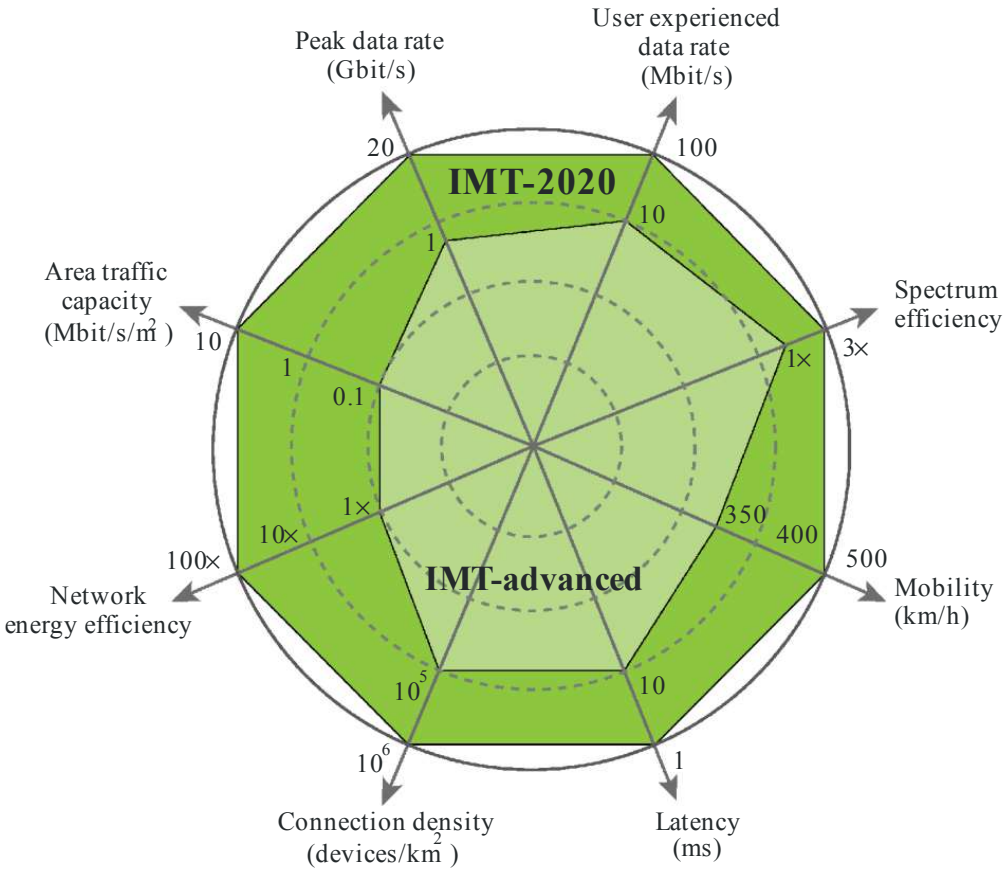


Figure 2.1: Key enhancements in the IMT-2020 standard, taken from ITU-R 2083-0[14]

In Figure 2.2 it can be seen that not all these improvements are equally important for all 5G use cases. The URLLC use case, for example, requires ultra low latency and high mobility speeds, but the user experienced and peak data rate are not that important, whereas the eMBB use case does not require ultra low latency. In reality, it is not feasible to expect 5G to fulfil all these requirements simultaneously, as described by Chih-Lin et al.[15]. Thus, 5G has to support different subscriber profiles and network configurations which emphasize one of the use cases. This flexibility is provided by network slicing, described in Section 2.1.6.

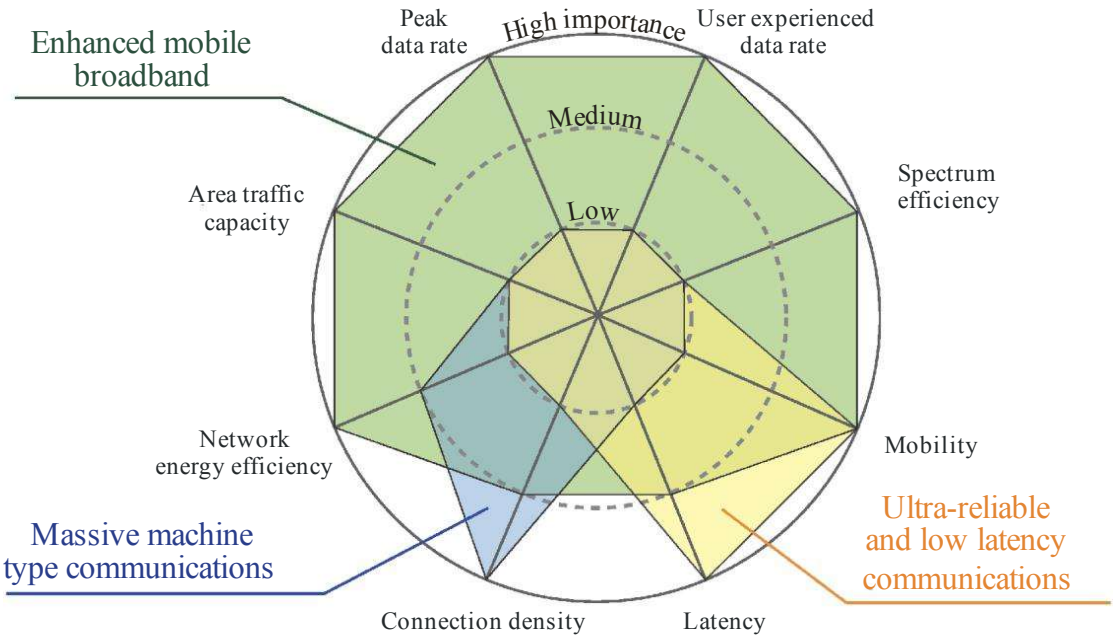


Figure 2.2: Importance of IMT-2020 key enhancements for different use cases, taken from ITU-R 2083-0[14]

2.1.2 Architecture

The 5GS consists of two major building blocks: The RAN and the core network. The 5G RAN is often referred to as New Radio (NR). The RAN consists of the gNodeB (gNB), the base station responsible for sending and receiving information over the air interface. The signal processing, channel coding and resource allocation happens here. The gNB also advertises the Public Land Mobile Network (PLMN) over a broadcast channel. This way the mobile device, the User Equipment (UE), is able to select the correct PLMN, matching the advertised Mobile Network Code (MNC) and Mobile Country Code (MCC) with the information stored on the Subscriber Identity Module (SIM) card. Details about the RAN are briefly discussed in Section 2.1.5. However, this thesis focuses on the procedures within the 5GC. Hence, the interested reader is advised to read Cox[16][17] or Henttonen et al.[18].

The flexibility of 5G does not only apply to the use cases, but the 5G system itself. It is possible to have different deployment options for 5G. The NSA deployment option aims to ease the transition from 4G to 5G. This means, that the 5G RAN is used, but the core network is an updated version of the 4G core network, the Evolved Packet Core (EPC). On the other hand, the 5G specification comes also with its own core network, the 5G SA core (5GC). The concepts described in this thesis rely on a 5G SA network, hence the EPC is not described further in this thesis. The interested reader is advised to follow Olsson et al.[19].

In total, there are four different ways 4G and 5G may be deployed (Rommer et al.[20]):

- LTE for signaling and user traffic
- NR for signaling and user traffic
- Combination of NR and LTE, where LTE is used for signaling and LTE and NR for user plane (UP)
- Combination of NR and LTE, where NR is used for signaling and LTE and NR for user plane (UP)

Considering that there are two different core networks available, there are in total eight different deployment options, as described in Table 2.1.

	Access Network			
Core Network	LTE only	NR only	LTE with NR UP	NR with LTE UP
EPC	Option 1 (4G)	Option 6*	Option 3	Option 8*
5GC	Option 5	Option 2	Option 7	Option 4

Table 2.1: 4G/5G deployment options, adapted from Rommer et al.[20]. Disregarded options are marked with *.

Option 1 is the current LTE deployment and Option 6 and Option 8 were disregarded by the 3GPP during the standardization process, as not all the advantages of NR can be utilized, when using NR with an old core network.

In the end, the 3GPP focused on the options that had the largest market value: Option 2 and Option 3. Option 3 is the described NSA option, while Option 2 is SA. Hence, the underlying deployment option this thesis operates on is Option 2.

Option 5 may yet be used in the future, but it is questionable why one would use a new core network only for LTE and not for NR. The same is true for Option 7. Option 4 is interesting, as a new Mobile Network Operator (MNO) who does not own an LTE infrastructure may use this option to support 4G and 5G radio. Altogether, one could argue that the 3GPP has introduced too much flexibility and also complexity, because only two or three use cases cover the vast majority of real-world scenarios.[20]

2.1.3 User Plane and Control Plane

One of the most important concepts of the 5GC and EPC architectures is the distinction between control plane – also called signaling – and user plane, as described by Rommer et al.[21]. The control plane consists of all the traffic that is used to control and manage the user's registration and Protocol Data Unit (PDU) session. A PDU session is unique

for each of the UEs in the network and provides connectivity to a Data Network (DN). In the vast majority of the cases, the DN is the internet, although in this thesis the edge is also a DN. Hence, a PDU session enables the user to connect to the internet (or another network). All the traffic that is sent over this PDU session is user traffic or user plane, e.g., HTTP traffic. The control plane handles the registration and authentication of the subscriber and the setup and deletion of a PDU session. For example, enabling mobile data on a smartphone triggers signaling traffic over the control plane and once the user is successfully connected, the subsequent traffic is sent over the user plane. Although this logical distinction is part of mobile networking since 2G, it has not been physically separated, meaning all the control plane and user plane terminated in the same components. Only in an update of the EPC architecture, Control and User Plane Separation (CUPS) has been introduced with 3GPP Release 14[W6] in 2017, mere two years before Release 15[W7], the first 5G specification. It allows to terminate the user plane on a different physical and logical host than the control plane. While this is optional for the EPC, the 5GC is built around this principle. As described in Section 4.4.1, this is an important prerequisite for 5G edge computing. Figure 2.3 shows the overall 5G architecture. The `N3` reference point between the gNB and the UPF is part of the user plane, as well as the `N6` reference point between the UPF and the DN. Of course, user plane traffic is also sent over the RAN, as described in Section 2.1.5. All the traffic between the other reference points is part of the control plane.[21]

Note: The user plane is often abbreviated as UP and the control plane as CP in the 5G standardization. The author chooses to use write these out to avoid ambiguity.

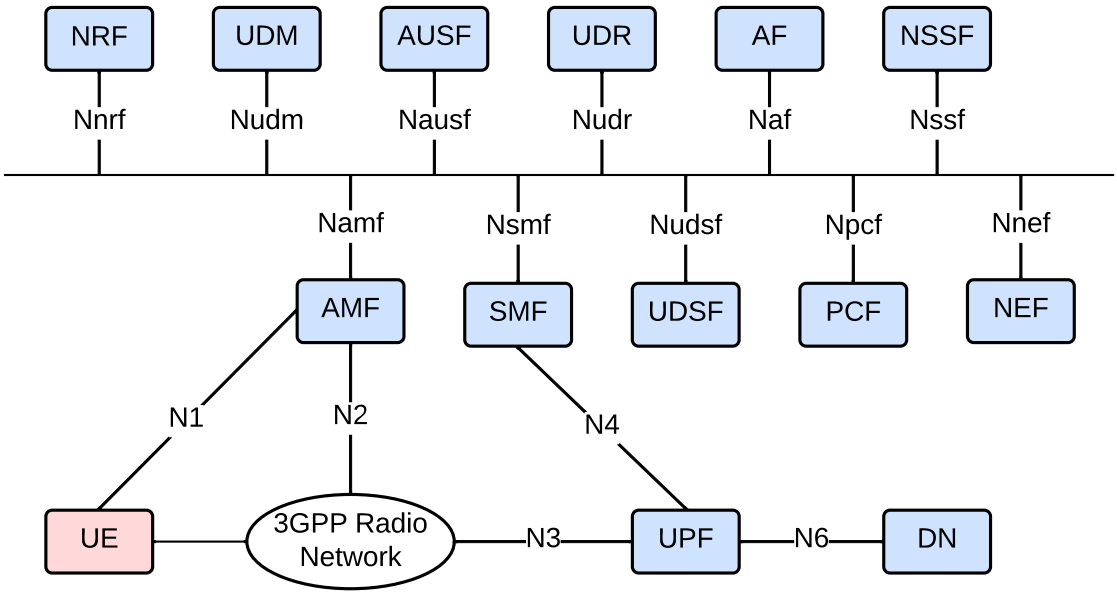


Figure 2.3: 5G core network architecture, adapted from Rommer et al.[20] and 3GPP TS 23.501[22]

Control Plane

The control plane is responsible for managing everything related to the UE's PDU session. There is a control layer in the RAN responsible for managing the radio conditions, as described in Section 2.1.5. Also, the user plane and control plane traffic use different protocols on the air interface, as depicted in Figures 2.4 and 2.5 in Section 2.1.5.

While the EPC consists of three essential components, the 5GC introduces the concept of a Network Function (NF). Each EPC component has been split up into distinct NFs in the 5GC. The NFs that are relevant for this thesis are described in Section 2.1.4.

The control plane interface between the core network and the UE is called Non-Access Stratum (NAS) and is transparently passed through the RAN. The protocol used in 4G is the S1 Application Protocol (S1AP), while 5G uses the NG Application Protocol (NGAP). As described by Chlosta et al.[23], S1AP and NGAP are very similar. Both are binary protocols used for attachment, registration, authentication and IP address allocation. The endpoint for NGAP in the 5GC architecture is the Access and Mobility Management Function (AMF).

User Plane

The user plane traffic is the Uplink (UL) – from UE to DN – and Downlink (DL) – from DN to UE – traffic sent and received by the UE. As the main concept behind the PDU session is to enable connectivity to the internet, the PDU session type is either IPv4 or IPv6. Other types such as Ethernet are supported as well, but only for specific use cases.[21]

The UE gets an IP address assigned by the 5GC over the control plane which it uses to connect to the internet. As described in Section 2.1.5, the user plane between the gNB and the UPF is IP-based. However, there is an additional layer in the protocol stack due to the GPRS Tunneling Protocol (GTP). GTP is used to encapsulate the user's IP session in another IP session. This so-called GTP tunnel allows to correlate the radio channels on the UE side with the IP world on the gNB. Whenever the UPF forwards packets to the DN, it removes the GTP layer and adds it in DL direction.[21]

As the name suggests – General Packet Radio Service (GPRS) is the first mobile data service from 2001 in 2G – the user plane traffic within the core network did not significantly change during the evolution from 2G to 5G. Although different gateways are used to terminate the GTP tunnels, the basic concept of tunneling is the same.

2.1.4 5G Core (5GC)

While the user plane routing and handling is very similar in 4G and 5G, the control plane is vastly different. The representation of the 5GC in Figure 2.3 is called the 5G Service Based Architecture (SBA). It does not contain all the point to point connections between the NFs, but focuses on the producers and consumers of APIs. There is also a reference-point based architecture, where all the NFs which communicate with each other

are connected. However, this representation tends to be unclear and overly complex. The 3GPP standard 3GPP TS 23.501[22] describes the basic 5G architecture and the requirements for each of the NFs.

Each Service Based Interface (SBI) has a distinct name, which starts with the letter `N` and is followed by the component's name, e.g., `Npcf`. These are all RESTful interfaces. The reference-point architecture uses a numbering scheme, e.g., `N4` is the reference between SMF and UPF. As shown in Figure 2.3, even when using SBA, some interfaces are reference-based. These are all binary interfaces:

- `N1`: NAS, described in 3GPP TS 24.501[24], uses NGAP protocol (3GPP TS 38.413[25])
- `N2`: NGAP protocol (3GPP TS 38.413[25])
- `N3`: User plane with GTP, as described in 3GPP TS 29.281[26]
- `N4`: User plane configuration, uses Packet Forwarding Control Protocol (PFCP)
- `N6`: User plane without GTP, i.e., IP

Not all of the NFs from Figure 2.3 are necessary to operate the basic use cases of a 5GS. The components AMF, SMF, Unified Data Repository (UDR), Unified Data Management (UDM), Authentication Server Function (AUSF) and UPF offer the essential services which are necessary to authenticate and register a UE and provide it with a PDU session, hence connection to a DN. It thus may also be called "core of the core".[20]

Session Management Function (SMF)

The SMF is responsible for handling the PDU session of a subscriber and IP address assignment for a given UE. It also configures the user plane routing via the `N4` interface. The PDU session API is described in 3GPP TS 29.502 [27].

Access and Mobility Management Function (AMF)

The AMF is the endpoint for the NAS signaling and is reponsible for access and mobility management. It also selects the appropriate SMF and invokes its PDU session API. The AMF communication API is used mainly by the SMF to inform the AMF of changes in the PDU session and is described in 3GPP TS 29.518[28].

Unified Data Repository (UDR)

The UDR is the main subscriber database and stores various different data, from basic subscription data such as the International Mobile Subscriber Identity (IMSI) to cryptographic keys. It offers access to these values through the data repository API, described in 3GPP TS 29.504[29].

Unified Data Management (UDM)

The UDM is the frontend to the UDR and offers easy access to subscriber's data to the AMF and SMF. It also keeps track which AMF or SMF is used for a specific subscriber. The UDM offers several APIs, but the most important are the subscriber data management API and the UE authentication API, both described in 3GPP TS 29.503[30]. These two APIs are essential for the basic use case of registration and authentication.

Authentication Server Function (AUSF)

The AUSF provides an authentication service to authenticate a UE. It also generates the temporary session keys which are used during the lifetime of a UE registration. It does so by receiving the cryptographic master key from the UDM to derive the session keys. The UE authentication API is described in 3GPP TS 29.509[31].

Note: 5G security is not discussed in this thesis, but Jover and Marojevic[32] provide a security analysis of 5G and Sun and Du[33] discuss physical layer security.

Network Repository Function (NRF)

The NRF is the main repository function and enables other NFs to discover each other. The main discovery API is described in 3GPP TS 29.510[34]. The NRF is an essential component when the 5GC is virtualized and NFs may be spawned, moved or destroyed during their lifecycle.

Unstructured Data Storage Function (UDSF)

The UDSF is used to store unstructured, dynamic data. Most NFs are stateful, e.g., the SMF has to track all the PDU sessions. To fully enable cloud-native 5G core networks, this data can be stored in the UDSF, so that the SMF does not lose all its state when it is restarted. From a software engineering standpoint, it is comparable to a Redis database. Its data repository API is described in 3GPP TS 29.598[35].

Policy Control Function (PCF)

The PCF handles all the different policies for the 5GC. Therefore, it is important for configuration of different QoS parameters for different network slices, but is also necessary for the edge computing solution presented in this thesis. It offers two main APIs, one session management policy API towards the SMF, described in 3GPP TS 29.512[36] and a policy authorization API towards the AF described in 3GPP TS 29.514[37]. Both these APIs are described in more detail in Section 4.4.

Network Slice Selection Function (NSSF)

The NSSF assists the AMF in selecting appropriate network slices. It also helps the AMF to discover other AMFs in case the current AMF does not support the slice configuration. Its APIs are described in 3GPP TS 29.531[38]. The concept of network slicing is described further in Section 2.1.6.

Network Exposure Function (NEF)

The NEF is used mainly for security reasons and to abstract the complexities within the core towards a third party NF, i.e., an AF. Its behavior and functionality is comparable to an API GW and it supports configuration of the 5GC from external third parties as well as providing information and forwarding events. It serves two domains: The southbound domain (the 5GC) and the northbound domain, the external AF. The NEF provides many different APIs. The most important northbound APIs are described in 3GPP TS 29.522[39] and southbound APIs are described in 3GPP TS 29.551[40], 29.541[41] and 29.591[42].

Application Function (AF)

The AF is not specified in great detail, as it is an abstract concept of any third-party provided service. It interacts mostly with the NEF or the PCF, as described in greater detail in Section 4.4. In this thesis, the AF is an Edge Enabler Server (EES) or MEC Platform (MEP), but it may also serve other purposes for other use cases. It can offer APIs to the core network, although this is not necessary[22].

User Plane Function (UPF)

The UPF is the NF for handling and routing the user traffic.

The UPF communicates with the SMF using PFCP. The SMF instructs the UPF how a specific PDU session is to be routed. Therefore, it is easy to virtualize the signaling NFs, as they do not handle any performance-critical traffic. The UPF may be a software or hardware switch, considering that software routing still provides a significant performance challenge, as described by Zhang et al.[43].

2.1.5 5G RAN

The 5G RAN is designed to work on different frequency ranges, from 600 MHz to 6 GHz in the sub-6 GHz frequency bands. Additionally, 5G is able to operate in the millimeter wave range, which corresponds to a frequency between 20 GHz and 60 GHz, as described by Cox[44]. In Austria, several frequency bands are allocated to 5G in the low frequency range such as 700 MHz, 800 MHz and 900 MHz, but also up to 2.6 GHz. The 3.4 GHz - 3.8 GHz and 26 GHz frequency bands are currently under consideration.[W8] This example shows that current 5G networks operate on the same or similar frequencies as the previous mobile generations.

As described in more detail in Chapter 4.2, NR supports different numerologies, resulting in different Sub-Carrier Spacing (SCS) configurations. This allows to increase or reduce the available bandwidth and also the latency over the air interface, supporting the different 5G use cases.[17]

The 5G RAN operates on different protocol stacks for user plane and for control plane, as described in 3GPP TS 38.300[45]. The control plane is used for resource control between the UE and the gNB, but also for the underlying transport for the NAS communication. Figure 2.4 shows the user plane protocol stack and Figure 2.5 shows the control plane protocol stack. These figures indicate that the gNB is acting as a gateway between the air interface and the IP domain of the core network. The gNB is connected via the backbone of an MNO to the core network, but it is important to note that the communication is IP-based. This is also the reason why GTP is necessary. It encapsulates another IP layer, from the device to the final host such as a web server. The IP addresses in the IP layer below the GTP layer belong to the gNB and the UPF.

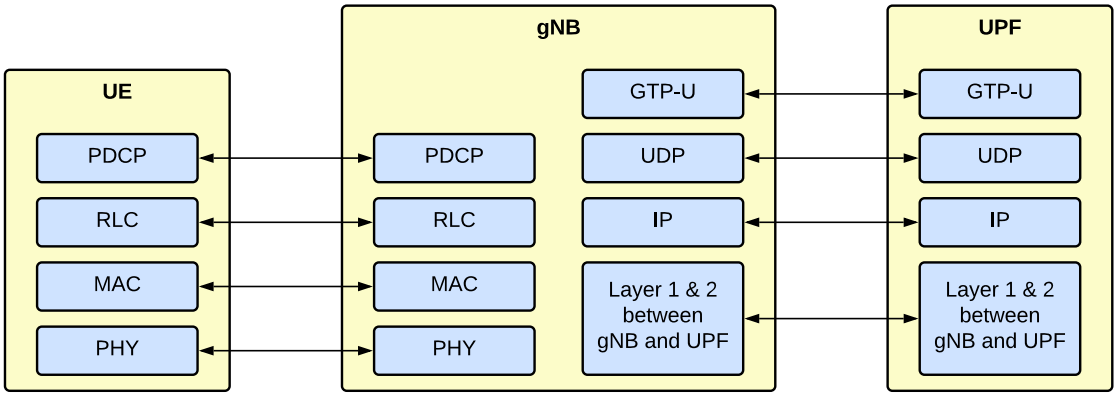


Figure 2.4: NR protocol stack for user plane, adapted from 3GPP TS 38.300[45]

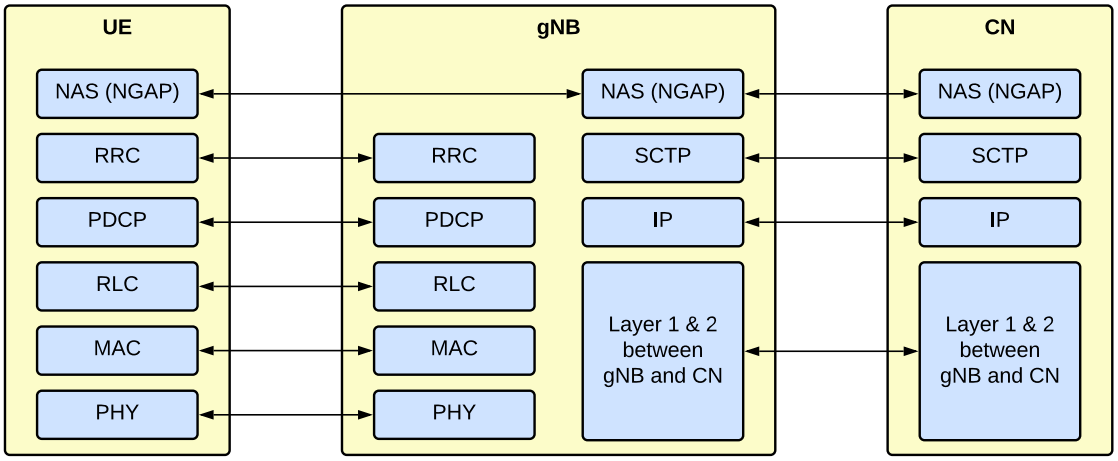


Figure 2.5: NR protocol stack for control plane, adapted from 3GPP TS 38.300[45]

PHY

The physical layer is the backbone of the 5G NR, as described by Zaidi et al.[46]. It has to be flexible enough to support the different – sometimes contradictory – 5G use cases and requirements. Also, it is built to support the NR frequency range. The physical layer handles the modulation of the signal.[46]

The physical layer also defines the waveform used in NR. In contrast to LTE, NR uses the same waveform in UL and DL direction: Cyclic prefix Orthogonal Frequency-Division Multiplexing (OFDM), which simplifies the design of an NR Radio Unit (RU).[46]

Another important aspect of NR is the support for multiple antennas, especially Multiple Input Multiple Output (MIMO), which is an important feature to further increase the throughput of the NR connection. The PHY layer also employs channel coding.[46]

Medium Access Control (MAC)

The MAC layer is mainly responsible for scheduling transmissions over the PHY layer.[17] It provides a mapping between the logical channels of the upper layers and the physical channels of the PHY layer. The MAC layer also performs error correction using Hybrid Automatic Repeat Request (HARQ), a combination of forward error correction and ARQ, which is also used in TCP.

Radio Link Control (RLC)

The RLC layer in NR does not differ significantly from the RLC layer in 4G.[17] The RLC layer supports different modes: Transport Mode (TM), Unacknowledged Mode (UM) and Acknowledge Mode (AM). Choosing the right mode is essential for the application.

Packet Data Convergence Protocol (PDCP)

The PDCP layer is used for the transmission of higher layer IP packets as well as Radio Resource Control (RRC) information. It is also the layer responsible for header compression, decompression, integrity protection, encryption and decryption of the user plane and control plane.[17]

Note: How the cryptographic keys are generated and transported to the gNB are not discussed in this thesis, but the inclined reader is referred to Prasad et al.[47] and 3GPP TS 33.501[48].

Radio Resource Control (RRC)

The RRC layer is the control layer of the air interface. It configures the lower radio layers PHY, MAC, RLC and PDCP. The RRC keeps track of the different states of an NR connection, i.e., if the UE is idle, inactive or connected. Further, RRC is used when the UE communicates with the CN via NAS using the NGAP protocol, as depicted in Figure 2.5.

2.1.6 Network Slicing

As described in Section 2.1.1, 5G has to support contradicting use cases and thus its design must be very flexible. Network slicing is one of the core concepts to handle this flexibility, as described by Zhang[49]. The one-size-fits-all principle of LTE is not sufficient anymore for 5G and building different physical networks with different configurations is economically and technically not feasible. Thus, network slicing is introduced in 5G. It allows to have logically separated instances in the same physical network.

The NSSF supports the UE to select an appropriate slice based on the requirements. The result of this procedure is an NSSAI, a slice identifier. Then, the UE registers to the CN with this NSSAI. The NSSAI is used to select different virtualized NFs.[49] For example, there may be a virtualized SMF serving one specific slice and during registration the AMF selects this SMF, which is configured to support a URLLC scenario.

Therefore, network slicing is also an important concept for edge computing in 5G. Using network slicing, it is already possible to create 5G campus networks. A campus network is a 5G network for subscribers affiliated with an organization, e.g., an industrial site, as described by Rischke et al.[50]. Network slicing allows these organizations to have their own quasi-private 5G network without having to build their own 5G infrastructure. This also allows to deploy and select a UPF based on an SNSSAI to reduce latency, hence supporting the edge computing and URLLC use case. Therefore, network slicing presents an alternative solution to the approach described in this thesis, as described by Ksentini and Frangoudis[51].

An important aspect of network slicing is that the RAN is also slice-aware. This means, that the RAN is able to select different configurations, e.g., for scheduling.[49] This is an important use case for URLLC, as the configuration of the layer 1 and layer 2 of the RAN have to be configured accordingly.

2.2 Edge Computing

As described by Grover and Garimella[52], presently many IoT devices are connected to the cloud infrastructure through either wired or wireless communication. This leads to high costs in terms of latency, which can have a great negative impact for many real-time applications. This leads to the paradigm of edge computing with the primary goal of enabling these latency-critical applications.

2.2.1 About Edge Computing

The edge computing paradigm allows to dynamically allocate resources to an application.[52] Different applications such as IoT devices, autonomous robots and VR applications have different resource constraints in terms of latency, but also computation power and even the availability of specific hardware, such as a GPU for AR/VR. Therefore, edge computing has to provide this flexibility in resource allocation. The different deployment

options for edge data centers form a hierarchy. Grover and Garimella classify the different options as follows[52]:

- Extreme edge: Local data centers in the LAN, similar to a private cloud
- Fog servers: Between the LAN and the WAN. In the context of this thesis, this is a deployment near or within a 5G gNB and the main use case described herein
- Core: Edge data center in the WAN
- Cloud: The traditional cloud deployment model

Hassan et al[2] identify four key requirements for edge computing in 5G:

1. Real-time interaction
2. Local processing
3. High data rates
4. High availability

The first requirement is the main driver for the edge computing use case. Edge computing must ensure low latency communication for latency-constrained applications. In 5G, this is achieved using the URLLC features on the RAN and the different approaches discussed in this thesis, especially in Chapter 4.

Further, local processing helps to avoid bottlenecks in the communication between a gNB and the core network or a cloud.[2] Even if there is a global cloud available and necessary, in many cases data is aggregated at the edge to reduce the communication overhead of the internet.

Although not all low latency applications require high data rates, some have this requirements, especially in the area of AR, VR and cloud gaming.[2] Hence, this is also a key requirement for edge computing.

High availability is a major challenge in cloud computing, as discussed by Mesbahi et al.[53] and there are many aspects to it. For edge computing solutions to be technically and commercially viable, they should guarantee the same availability as cloud computing.

Khan et al.[54] describe similar key requirements for edge computing, although they focus more on the requirements on how to enable edge computing. A solution integrated into 5G is presented in this thesis in Chapters 4 and 5.

2.2.2 Edge Computing and Cloud Computing

Edge computing and cloud computing share many similarities. In fact, they both rely on virtualization as underlying technology, as described by Abbas et al.[55]. In any case, edge computing is not a replacement for cloud computing, but it is a part of the cloud computing hierarchy. Thus, edge computing is a specific flavor of cloud computing.

Table 2.2, adapted from Bragadeesh and Arumugam[56], provides a comparison between cloud computing and edge computing.

Characteristic	Cloud computing	Edge computing
Computational capacity	High	Medium to low
Deployment strategy	Centralized	Distributed
Latency	Medium to High	Low
Real-time possible	No	Yes
Long-term storage	Yes	No
Backbone communication	High	Low
Mobility Support	Low	High
Cloud Provider hosted	Yes	Yes
MNO hosted	No	Yes
Automated deployment	May be done upfront	Necessary
Devices	UEs , PCs	UEs, IoT devices
Network Type	WAN	LAN/WAN

Table 2.2: Comparison between cloud computing and edge computing[56]

2.2.3 Orchestration

Orchestration in the context of cloud computing is a framework for managing the applications and the hardware of a cloud data center, as described by Costa et al.[57]. Orchestration is common in large cloud service providers, as it allows to help with automatic scaling of applications and efficient resource allocation.

When applications are deployed using edge computing, the role of orchestration becomes even more important. As described in Table 2.2, mobility is an important factor in edge computing, especially in 5G. The distributed edge resources also tend to be more heterogeneous. These challenges are managed with an orchestration framework. The framework takes the role of a centralized entity to handle resource management, life cycle management and mobility[57].

Section 4.3 describes two standardized orchestration frameworks in 5G: ETSI MEC and 3GPP EDGEAPP, whereas the orchestration aspects of ETSI MEC are more detailed.

2.3 Virtualization

As described in Section 2.2, virtualization is the key technology behind cloud computing and edge computing.

2.3.1 Virtual Machines and Container Virtualization

Virtualization allows to logically separate the resources of a computer into different systems. As described by Sharma et al.[58], there are different approaches to virtualization:

- Hardware-level virtualization
- OS-level virtualization

The hardware-level virtualization operates based on a Virtual Machine (VM). As the name suggests, a VM can run its own Operating System (OS) and applications and has thus a great degree of independence of the underlying OS and hardware. There are two main types of virtual machine managers. One operates on the hardware level (Type 1 Hypervisor) and the other uses a guest OS to run the virtual machine manager. Famous examples are VMWare[W9] (Type 1 and Type 2) and Virtualbox[W10] (Type 2)

The most important distinction between VMs and containers is that containers share the underlying host operating system, i.e., the kernel of the OS, which is why it is called OS-level virtualization. This has the advantage that the instruction set of the hardware does not have to be emulated, resulting in higher speed. However, the logical separation of containers is weaker compared to VMs.

The greater flexibility of containers make them the preferred choice for edge computing use cases[57]. Hence, the following sections describe mainly container-based virtualization techniques.

2.3.2 Network Function Virtualization (NFV)

NFV is a framework for virtualization of networking infrastructure, standardized by ETSI. The specification ETSI GS NFV 002[59] describes the architecture.

NFV has been created to utilize the advantages of cloud-computing without compromising complying with standardized technology. It allows to bring virtualization into network of MNOs in a "carrier-grade" quality and stability. Further, it not only focuses on virtualizing applications usually deployed in the cloud such as web servers, but also on virtualizing the networking equipment itself, as discussed by Mamushiane et al.[60].

The NFV architecture consists of three main components[59]:

- A Virtualized Network Function (VNF) is any virtualized NF or application running on a Network Functions Virtualization Infrastructure (NFVI)
- The NFVI is the virtualization infrastructure. It consists of the hardware and the underlying hypervisors for virtualization.
- The NFV Management and Orchestration (MANO) is responsible for the orchestration and life cycle management of VNFs.

An important aspect of the Network Function Virtualization (NFV) architecture is that it supports dynamic instantiation and deployment of VNFs. To accomplish this, the VNFs are described in a standardized, interchangeable format.

The ETSI NFV architecture is also able to interact with the ETSI MEC architecture, as briefly discussed in Section 4.3.1.

The open standardization of the ETSI NFV architecture led to several open source implementations thereof, where OSM and ONAP are arguably the most prominent.[60]

2.3.3 Docker

Docker is an open source container virtualization solution, especially prominent among software developers. It provides an exchangeable description format to build container images, the Dockerfile. These images can be uploaded and downloaded from a central repository. Docker also provides the user-facing frontend to run containers. Additionally, `docker-compose` and Docker swarms are orchestration solutions to manage several Docker containers, as described on the Docker homepage[W11].

2.3.4 Kubernetes

Kubernetes – also referred to as K8s – is an open source container orchestration framework. K8s and Docker are often used together, but it is important to note that it does not replace Docker. While Docker provides the format and the container runtime, K8s handles the orchestration and management of containers, as described on the Kubernetes homepage[W12].

2.3.5 Red Hat OpenShift

Red Hat OpenShift is a container orchestration platform. It is built on top of Kubernetes. According to Red Hat[W13], it can be used to manage hybrid cloud, multicloud and edge deployments. Openshift is built on the Red Hat Enterprise Linux and is compatible with the Red Hat Ansible Automation platform. Thus, Red Hat is offering a complete application package for container orchestration.

2.4 ROS2

ROS is an open source robotics framework and was developed and popularized by Willow Garage in 2007, as described by Macenski et al.[61]. It provides libraries for developing robot applications. Furthermore, it contains a communication middleware for exchanging data between different modules.

2.4.1 Fundamentals

The main functionality of ROS2 is to provide a rich ecosystem for robotics applications. There are many parts of a robot's software, from handling the sensors and actuators correctly, implementing path planning and steering to obstacle avoidance. To support development, ROS2 allows to logically (and physically) separate these components into packages. Therefore, ROS2 must define how the different packages communicate with each other, as well as define a messaging format.[61]

In ROS2, a package is organized into nodes. Each ROS2 node has a distinct functionality, e.g., taking a steering command as input and moving the robot's wheels. This separation of concerns allowed developers to focus on a specific part of the robot and nowadays the ROS ecosystem consists of many open source packages by different contributors. The ROS2 navigation stack, as described by Macenski et al.[62] is a prominent example. Also the Google SLAM algorithm used in this thesis is an open source ROS2 package.

ROS2 Topics

ROS2 topics is the most common communication pattern used. It is an asynchronous exchange of messages in a publish/subscribe manner. Each ROS2 node describes which topics it subscribes to and which topics it publishes. The topic description must be defined and available to consumers and producers up front. This mechanism allows for anonymous many-to-many communication over a distributed system.[61]

Services

ROS2 also supports synchronous communication using the request - response pattern. A client sends a request to a server and waits for the response. ROS2 also supports non-blocking client calls to support an asynchronous message handling.[61]

Actions

The action communication model is a more specific use case, which is necessary for many procedures in a robot. It allows for a goal-oriented communication. A client sends a goal to a server and asynchronously receives feedback on a task. This is used for long-lasting tasks, e.g., when the robot should move to a target pose.[63]

2.4.2 Transformations (TF)

An important concept of ROS2 is the TF tree. It provides transformation matrices from one pose to another. Within the TF tree, there are many different reference frames. For example, the origin of the map is a static frame. In many cases, its child reference frame is the odometry frame. Upon start of the robot, it normally coincides with the map frame, but when the robot is moving, there is a visible drift. When the robot is able to locate itself, there is a transformation from the odometry frame to the robot's base link. Then, subsequently from the base link there are TF frames to different parts of the robot. Altogether, to get the linear and rotational difference from one pose to the other, all these matrices along the TF trees are used to calculate the final transformation. ROS2 provides libraries for programmers for TF transformations. The ROS2 TF tree is described in the ROS2 documentation[W14].

2.4.3 ROS and ROS2

ROS got a lot of traction in the last years. It had been used mainly as a research platform and was very successful in that regard. However, as soon as some projects moved to industrial and commercial applications, some weak points of ROS were revealed. There were security and reliability concerns and the communication middleware was slow and inefficient.[61]

To overcome these limitations, ROS has been redesigned from scratch and is now called ROS2. One of the most important differences between ROS and ROS2 is the communication middleware. While ROS operates on a custom-built middleware, ROS2 uses the existing Data Distribution Service (DDS) standard and relies on open source implementations thereof. This enables peer-to-peer discovery without using a centralized master node. Details about DDS and communication in ROS2 are discussed in Section 6.4.

CHAPTER 3

Related Work

This chapter describes the related work of this thesis, in the areas of MEC in 5G NSA, TSN and ROS.

3.1 MEC in 5G NSA and 4G

Ksentini and Frangoudis[51] describe MEC in 5G in combination with network slicing. They have implemented a MEC orchestrator and a MEP (see also Section 4.3.1). The MEP is integrated with the OAI 4G network stack, i.e., the EPC in the core network. FlexRAN is used to interact with the RAN, the 4G eNB. FlexRAN is described by Foukas et al.[64] and allows to configure eNBs or remotely receive RAN-level information. The traffic rules are provisioned over the Mp2 interface. As discussed in Chapter 4.3.1, this interface is not specified by the MEC standard. Thus, the OAI 4G core has been adapted to support this use case. They use Open vSwitch (OVS) to configure the user plane routing. OVS is an open source implementation of a virtual switch to enable Software Defined Networking (SDN), as described by Pfaff et al.[65]. It is designed to fully support the OpenFlow protocol. OpenFlow has been created to have a standardized interface for configuring switches of different vendors, as described by McKeown[66]. The approach presented in this thesis uses the 3GPP PFCP and the standardized scenarios in the 5G core network to achieve the same result. In fact, the PFCP protocol borrows many concepts from OpenFlow, but has been designed with mobile networking use cases in mind (see also Rezazadeh et al.[67]).

LL-MEC, presented by Nikaein et al.[68], is an open source low latency MEC platform. It uses the FlexRAN[64] and OpenFlow protocol[66] as well. It provides two APIs on the Mp2 interface: The Radio Network Information Service (RNIS) and the Edge Packet Service (EPS). RNIS is a standardized API in the MEC architecture (see also Chapter 4), whereas the EPS is part of the LL-MEC design. It is responsible to dynamically adjust the routing of the core network to serve the edge computing use case. LL-MEC

targets 4G systems and due to the lack of standardized procedures in 4G to influence traffic routing, the LL-MEC EPS needs to be integrated into an EPC. LL-MEC has also been integrated into the OAI 4G network stack. The conducted work described in this thesis takes concepts from both LL-MEC[68] and the MEC platform by Ksentini and Frangoudis[51] and implements these in a standardized fashion in the OAI 5GS.

The VTT Technical Research Centre of Finland[69] has implemented a testbed which allows companies to test new production methods in factories. They describe a proof of concept where they conduct a similar experiment as in this thesis: A mobile robotic platform is connected to a 5G base station. The point cloud from the robot's 3D camera is sent to an edge server which calculates the pose of an object. Their experiment is conducted on the 5G Test Network (5GTN), a 5G research platform, as described by Piri et al.[70]. According to 5GTN, the underlying 5G implementation is provided by Nokia[70], which consisted of an NSA deployment for the VTT experiment[69].

3.2 Time-Sensitive Networking

Time-Sensitive Networking (TSN) is a new networking paradigm, as described by Finn[71]. It is based on the prevalent best-effort networking scheme. TSN allows to define a contract between client and application. It limits a transmitter to a certain bandwidth, allowing the network to reserve this bandwidth and scheduling resources explicitly for this participant. This contract offers bounded latency and no congestion. Further, there is no out-of-order delivery of packets and duplicates are removed. The latency may differ within the boundaries described in the contract. This is in contrast to the constant-bit rate scheme, where the jitter is essentially zero and the latency does not vary.

TSN is an important requirement for industrial control applications. These were never able to rely on the best-effort networking offered by Ethernet and thus many different solutions among the industry exist. TSN allows to support these applications using best-effort solutions with a manageable overhead. As Ethernet is so widespread, the costs are considerably lower than industrial networks. This creates an incentive for the industry to use TSN and Ethernet to reduce the costs, but also to unify different networking technologies and reduce the complexity.[71]

However, in the space of industrial IoT and autonomous robotics and especially cars, TSN may not be applicable, as the mobility does not allow a fixed-net connection. Therefore, it is possible to integrate TSN with a 5G network. The 5G Alliance for Connected Industries and Automation discuss in a white paper[72] how TSN and 5G can be integrated. Essentially, there is a TSN translator in the UE as well as in the UPF to route packets from a TSN Ethernet network to a 5G network and back to a TSN Ethernet. The 5G network acts as a TSN bridge in this scenario. This relates to edge computing in 5G in two aspects. First of all – as TSN targets latency-constrained applications – the concepts discussed for edge computing in this thesis apply. Also for a TSN integration, the nearest UPF shall be selected to minimize the latency as much as possible. Secondly,

the control plane of the TSN network integrates using a 5G Application Function (AF). This AF uses the same APIs as described in Section 4.4.

3.3 2D LIDAR SLAM

The ROS2 SLAM node used for 2D LIDAR mapping and localization in the experiment of this thesis is the Google cartographer. The cartographer is a software package that can be integrated with ROS, as described by the cartographer documentation[W15].

Hess et al.[73] describe the principles behind the Google cartographer. It is able to generate 2D grids of an environment with a resolution of 5 cm. As input source, the cartographer uses 2D LIDAR scans and optionally the robot's Inertial Measurement Unit (IMU) and odometry measurements. All recent laser scans are inserted into a submap. When a submap is finished, i.e., there are no new measurements, it is considered for loop closure. If the submap matched against the current estimated pose of the robot is good enough, it is used as a loop constraint for the optimization problem. The optimization is completed every few seconds. The loop closure scan matching has to finish faster than newly added scans. To ensure that the optimization problem is executed fast enough, a branch-and-bound approach is used.

3.4 Containerization of ROS2 Nodes

Aldegheri et al.[74] show how ROS-based robotics applications can be deployed on a cloud-server-edge architecture and have performed a similar experiment as in this thesis. Their work does describe the architectural setup and the way the ROS nodes are containerized using Docker and KubeEdge. Their findings presents an important reference point for the evaluation performed in this thesis. They conclude that the performance overhead of containerized ROS nodes is within 5% compared to the non-containerized nodes.

Shibuya et al.[75] show that it is also possible to containerize ROS2 nodes. In their experiment they use the Navigation2 stack and the Google cartographer SLAM node. Their findings show that the experiment conducted in this thesis is feasible and comparable to a non-containerized ROS2 experiment.

3.5 Evaluation of SLAM Performance

Filatov et al.[76] describe methods for comparing and evaluating performance of different SLAM algorithms. They include different quantitative evaluations of the quality of 2D SLAM results. However, they state that the easiest and most straight-forward approach is to compare the estimated trajectory of the robot with the ground truth pose. This approach is described by Huletski et al.[77] and is also chosen for the evaluation of the SLAM quality in the different deployment options in the experiment described in this thesis.

3.6 Performance Metrics for Low Latency Applications

Schulz et al.[78] describe latency critical IoT applications and the requirements for the RAN and the core network architecture to enable these. They provide a comprehensive list of applications with their respective latency requirements. Factory automation for example has latency requirements between 0.25 ms and 10 ms, depending on the application. The paper also contains an experiment, measuring the latency of 4G networks in 2017. Their findings include different concepts for the RAN such as fast UL access and new waveforms, findings that have been mostly incorporated into the NR URLLC requirements and features. On the core network side, they express the need for edge computing and in particular MEC integrated in a 5G CN, as discussed in this thesis.

Voigtländer et al.[79] describe ultra-low latency control of a distributed robotics system. As part of their findings they describe the overall performance of the distributed robotics system connected to a 5G network, a similar approach as chosen in this thesis. However, they use steering control to balance a ball to evaluate the low latency performance, whereas this thesis uses an offloaded SLAM node.

3.7 Relevance

Section 3.1 describes different approaches for MEC in 5G NSA and 4G. The solutions from Ksentini and Frangoudis[51] and Nikaein et al.[68] have been integrated into the OAI EPC. As the EPC was not designed with edge computing in mind, the chosen approaches are proprietary and may only work with the OAI. This is in contrast to the solutions described in Chapter 4 and 5, which enable standard-compliant edge computing in 5G SA. Hence, the findings presented herein can be applied to different 5GC implementations.

TSN requires low latency by design. Thus, it has similar requirements as the edge computing use case described in this thesis. Furthermore, the procedures to enable TSN in a 5GC follow the same concepts and also require an AF, a PCF and an SMF. While TSN is not further elaborated in this thesis, the theoretical analysis in Chapter 4 and the OAI implementations in Chapter 5 are prerequisites to enable TSN in the OAI.

The Google cartographer described by Hess et al.[73] is the most important ROS node used in the experiment described in Chapter 7. Thus, an understanding of its workings are important to setup the experiment, and are crucial to interpret the results correctly.

The studies described in Section 3.4 and the evaluation methods described in Section 3.5 are necessary to define the methodology of the experiment. While Sections 3.3 and 3.4 provide the means to conduct the experiment, Section 3.5 describes how the results can be evaluated in a comprehensible and repeatable fashion.

The performance metrics described in Section 3.6 give valuable insights on the state-of-the-art URLLC applications and thus were used by the author to formulate the underlying hypothesis of this thesis.

CHAPTER 4

Edge Computing in 5G

This chapter discusses the different available architectures, platforms and approaches to bring edge computing into a 5GS. This includes the layer that communicates with external applications and the 5GS as well as the procedures how the user plane traffic is routed and how this routing is enabled via the control plane.

4.1 Overview

An end-to-end edge computing solution in a 5GS consists of several different building blocks. The foremost requirement of edge computing is to reduce the latency between a client and an application server.[2] In a 5G edge computing scenario, the overall response time T is defined as (see also Choy et al.[80]):

$$T = t_{\text{client}} + \overbrace{t_{\text{access}} + t_{\text{mno}} + t_{\text{transit}} + t_{\text{datacenter}}}^{t_{\text{network}}} + t_{\text{server}}$$

t_{client} is the processing time in the client (UE), from the application until starting the 5G transmission.

t_{access} is the transmission time over the air from the UE to the gNB.

t_{mno} is the packet delay between the gNB and the UPF, hence the delay within the operator's network until it is sent to any DN such as the internet or an edge data center.

t_{transit} is the packet delay between the operator's exit node (UPF) and the front-end router of the data center provider. In case of cloud computing, this is the transit time of the internet and the network is often provided by third parties and, therefore, neither in control of the MNO nor of the cloud provider.

$t_{\text{datacenter}}$ is the packet delay between the data center's front-end router and the server where the application service is hosted.

t_{server} is the processing time in the server.

In a cloud computing scenario, an MNO can only influence $t_{\text{access}}, t_{\text{mno}}$ and to some extend t_{transit}.

In edge computing, an MNO is able to reduce also t_{transit} and possibly $t_{\text{datacenter}}$ (when the MNO hosts the edge data center).

Figure 4.1 shows where in the 5GS the different latencies occur.

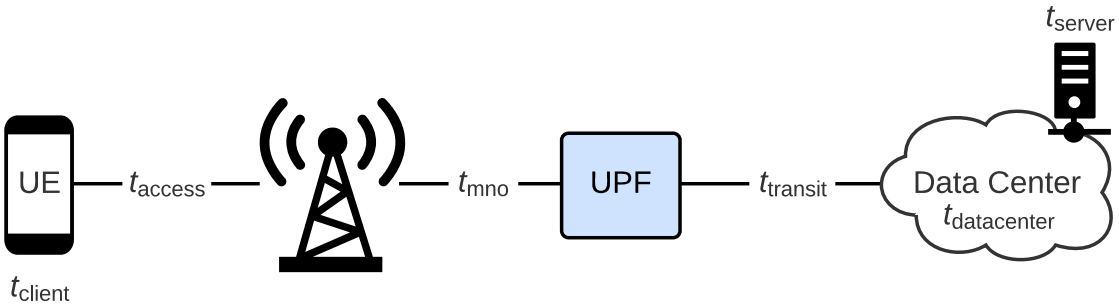


Figure 4.1: Latencies in the 5GS

To reduce the latency as much as possible for edge computing, all the following transit networks need to be taken into account:

1. RAN (t_{access})
2. Between the RAN and the UPF (t_{mno})
3. Between the UPF and the application server (t_{transit})
4. Inside the data center ($t_{\text{datacenter}}$)

Requirements for (1) are discussed in Section 4.2.

To reduce the latency for (2), the UPF should be as close to the gNB as possible, preferably co-hosted. In the end, this depends on the operator deployments and is also a trade-off between operational costs and latency requirements. In case a disaggregated RAN is used, the edge computing capabilities may be provided together with the Centralized Unit (CU). However, in any case, the 5GC has to be flexible enough to support different deployment use cases. This is described in Sections 4.4 and 4.5.

Requirement (3) is the reason why the cloud computing approach is not compatible with URLLC applications. As discussed by Choy et al.[80] in 2012, the latency between the client and a server in the cloud may be greater than 80 ms. The situation has improved over the last years because cloud providers deploy their data centers in more and more regions. However, in 2020, still more than 50% of users have latencies of more than 20 ms to the nearest cloud provider, as described by Charyyev et al.[81]. They also observe

that the number of covered users differ significantly between continents. While users in Western Europe experience low latencies in cloud services, users in Africa have mean latencies of 100 ms and above.[81]

Even if regional cloud data centers manage to reduce the mean latency to maximum 10 ms, the requirements of URLLC are not met. As described by Schulz et al.[78], the factory automatization use case has latency requirements between 0.25 ms and 10 ms. This can only be met with edge computing.

Thus – depending on the requirements – the distance between the UPF and the application service should be as small as possible. Ideally, they should be deployed in the same physical location in the same data center, possibly even together with the gNB. Another aspect of edge computing is mobility: As 5G is a mobile network, users may move and be served by another gNB. To ensure ongoing edge computing services, the application service should move together with the user. The management of deploying (and possible re-locating) edge application servers is done by edge computing platforms. These need to have an accurate user location and need to interact with the underlying 5G system to receive information, but also instruct it to route the traffic to satisfy the latency requirements. This is described in Section 4.3.

Satisfying requirement (4) is the responsibility of the data center provider and not described in this thesis.

4.2 RAN Requirements

The 3GPP coined the requirements for URLLC in the RAN in Release 15 and has extended them ever since, as described by Le et al.[82]. The 5G RAN features flexible SCS and sub-slot based transmission allow to reduce the latency. It introduces the concept of the numerology, a value that indicates SCS and the number of slots per sub-frame. Table 4.1 shows for each numerology (μ) which SCS frequency is used and the number of slots per frame.

μ	SCS [kHz]	No. of slots per subframe $= 2\mu$	No. of slots per radio frame $= 10 * 2\mu$	slot duration [ms]
0	15	1	10	1
1	30	2	20	0.5
2	60	4	40	0.25
3	120	8	80	0.125
4	240	16	160	0.0625

Table 4.1: Numerologies in 5G, from 3GPP TS 38.211[83]

One radio frame has the length of 10 ms. In LTE the SCS is fixed to 15 kHz and each frame consists of 20 slots, hence the slot duration is 0.5 ms. It is worth to note that the definition of a subframe has changed between 4G and 5G, thus the slot duration is 1 ms in 5G with an SCS of 15 kHz. Flexible SCS decreases the time length of the OFDM symbols[82]. A high SCS frequency reduces the duration of the slots and, therefore, reduces the overall latency for transmission.

LTE uses slot-based transmission, meaning that transmission is started in the beginning of the slot. Whenever a packet is received after the transmission has started, it needs to wait for the next slot. These wait times hurt the principle of low latency communication. 5G allows to start a transmission every 2, 4 or 7 OFDM symbols, further reducing the latency and allowing packets to be sent at additional times in the slot.[82]

5G NR has other features to reduce the latency of the transmission over the air interface even further, such as preemption indication, configured grant transmission for UL and adapted code rate and modulation schemes for URLLC[82].

4.3 Edge Computing Architectures

As discussed in Chapter 2.2, the concept of edge computing is very broad and involves many different technologies, architectures and aspects. Within this chapter, the most widely spread ETSI MEC and the novel 3GPP approach are discussed, as both are tightly-coupled to the 5GS.

4.3.1 ETSI Multi-Access Edge Computing (MEC)

The MEC architecture is defined by ETSI in the ETSI GS MEC 003[4] standard. When work on MEC started in 2014, the intention was to have "Mobile Edge Computing", targeted towards 4G, as described by Sabella[84]. In later developments of the standard, the meaning has been changed to "Multi-Access Edge Computing" and nowadays includes 4G, 5G, WiFi and also fixed net. Therefore, it is fair to say that MEC is access agnostic.[84]

Figure 4.2 shows the MEC reference architecture from ETSI MEC 003[4], adapted by Sabella[84], who added the categorization of the reference points. The reference points in green color (such as the Mp1) are specified, whereas the ones in red color are not specified, hence implementation specific. The device app may use the Mx2 reference point to interact with the MEC system to request instantiation and termination of MEC applications. This is done through the User Application Life Cycle Management (UA LCM) Proxy. The CFS stands for Customer-Facing Service Portal and allows customers to select MEC applications.[85]

The Operations Support System (OSS) and the MEC Orchestrator (MEO) in the MEC system level are mainly responsible for orchestration and management of the MEC system. The Mm1 reference point is used to trigger instantiation and termination of

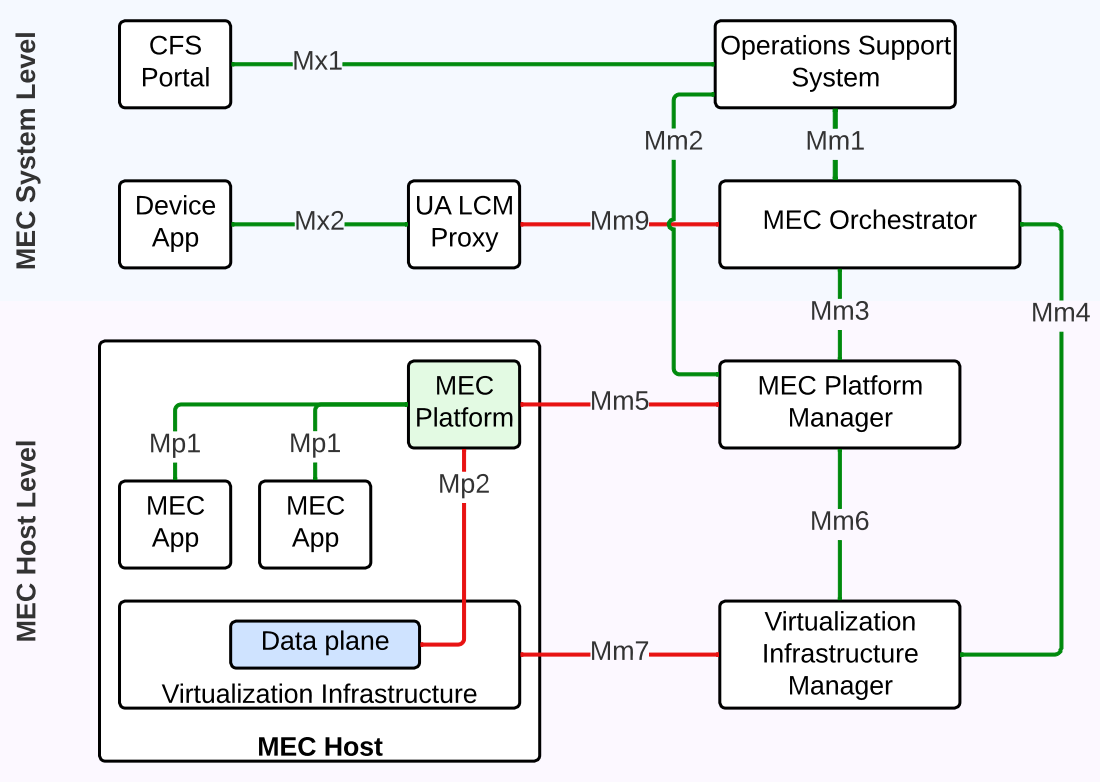


Figure 4.2: MEC architecture with specified (green) and unspecified (red) interfaces, adapted from Sabella[85]

MEC applications, whereas the Mm3 reference point is used to manage the lifecycle of the applications. The MEO has an overall view on the MEC system, including the MEC hosts and the topology. It is responsible to select the appropriate MEC host based on constraints (such as locality) upon app instantiation.[85]

On the host level, the MEC platform manager handles the life cycle and app instantiation for one MEC host. The Virtualized Infrastructure Manager (VIM) receives the configuration and the application images from the MEC platform manager and is responsible to instantiate these in the virtualization infrastructure.[85] As indicated in Figure 4.2, the interface between the virtualization infrastructure and the VIM is not specified. In practice, this depends on the available infrastructure in the edge. In an operator environment this is likely to be a NFVI, which is described in Chapter 2.3.2.

The components in the system level and the MEC platform manager are very important, as they have an overview of the MEC architecture, allowing the management of virtual machines and containers and on-demand instantiation and termination thereof. However, the focus of this thesis lays on the MEC host and especially on the unspecified Mp2 interface and the data plane. This is due to the fact that the Mp2 interface is responsible for interacting with the 5G SA core network and provides traffic routing information.

The MEC host is the main component for interaction from an application's perspective (application meaning in this context an actual application on a device). The host is deployed in the edge near the physical location of the device, for example near or next to the gNB or in a central node connecting several gNBs from one area. The MEC system level on the other hand may be a central component and does not need to be in the edge. This deployment is depicted in Figure 4.3.

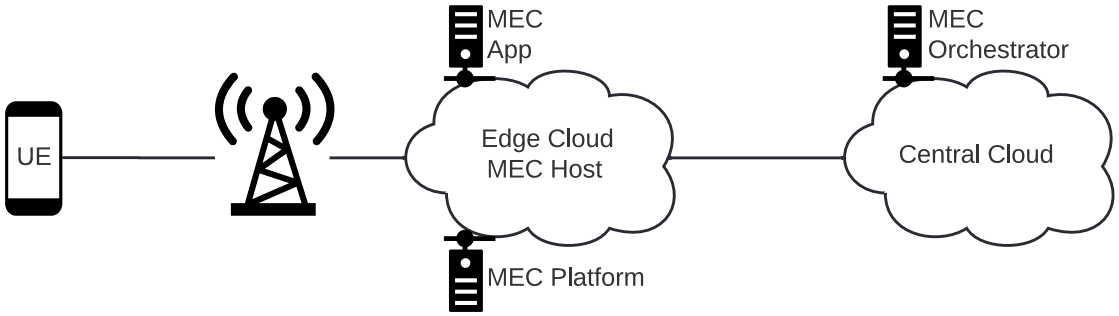


Figure 4.3: MEC host level and MEC system level deployment

The UE contains an application client which uses a service provided by a MEC application in the edge cloud. In the MEC architecture, the client is normally not aware of the edge. This is solved by providing a domain, which is resolved by a DNS. The MEC platform adapts the resolution in the DNS accordingly to ensure that the nearest MEC application's IP address is resolved. As indicated in Figure 4.2, the Mp1 reference point between MEC applications and the MEP is specified. It is used for application enablement and allows edge applications to offer and consume services. Upon start of a MEC application, the application registers itself on the MEP. The MEC application can instrument the platform to activate or deactivate DNS rules and activate, deactivate or update traffic rules. The traffic rules are used by the platform to enable the routing from the UE to the MEC application and vice versa. The platform uses the unspecified Mp2 reference point to interact with the underlying network, i.e., a 5G or a 4G network.[85] The Mp2 interface may differ depending on network technology and vendor and is discussed in detail in Section 4.4 for 5G SA.

Apart from application enablement, the MEC standard also specifies a set of essential services, which may be offered by a MEC application or the MEP itself. One of these services is the RNIS, which provides up-to-date network information about the radio conditions. This API may be used to improve other application services. It allows to make applications access-aware and act accordingly, e.g., when a handover happens. However, this is optional in the MEC architecture, as described by Sabella[86].

4.3.2 3GPP EDGEAPP

Although the ETSI MEC standard was originally aimed at 4G, there is a wide consensus in the industry that edge computing is a key enabler for 5G (see Sabella[87], Pham et al.[1] and ETSI White Paper #11[88]). This is indicated by the paradigm shift from

monolithic 4G network elements to a built-in CUPS in the 5GC, as described in Section 4.4.

This flexibility in the 5GC allows for 3GPP-compliant edge computing with the MEC architecture.[89] Nevertheless, the 3GPP has also standardized a novel edge computing architecture in Release 17 with the TS 23.558[5]. The standardization work for Release 17[W16] started in 2019 and has been finished (frozen) in May 2022.

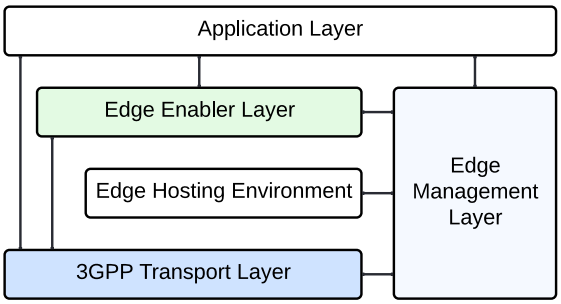


Figure 4.4: 3GPP edge computing architecture, adapted from TS 23.558[5]

Figure 4.4 shows the 3GPP edge computing architecture. One important thing that comes to mind is that the transport layer is explicitly stated as 3GPP transport layer and the standard refers to the 3GPP TS 23.501[22], which describes the 5G architecture (see Section 2.1). It is not surprising that the 3GPP edge computing standard focuses on the 3GPP transport layer. This allows for a closer and better specified interworking between the edge enablement layer and the transport layer. The Application Layer is a consumer of the edge computing capabilities and the Edge Hosting Environment is not specified by the 3GPP.[5] This relates to the Virtualization Infrastructure in the MEC architecture.

The Edge Management Layer is described in 3GPP TS 28.538[90] and is responsible to instantiate and terminate the components of the Edge Enabler Layer such as the Edge Application Server (EAS), EES and the Edge Configuration Server (ECS). Therefore, it has a similar role as the OSS, MEO, MEP and VIM of the ETSI MEC architecture.

The Edge Enabler Layer is the heart of the 3GPP EDGEAPP architecture and is the component which is responsible for interacting with the 3GPP Transport Layer. It is responsible for the lifecycle management of deployed edge applications and influences the routing and UPF selection in the 5GC. The architecture for the enabler layer is shown in Figure 4.5.

The EES is the central piece of the Edge Enabler Layer. It interfaces with all the other relevant components and provides APIs for these to interact with the server. The Edge Enabler Client (EEC) registers on the EES and uses its API for EAS discovery. It is also capable to trigger instantiation of an EAS on demand. Furthermore, it is the main component to influence traffic routing within the 5GC via the EDGE-2 reference point.[5]

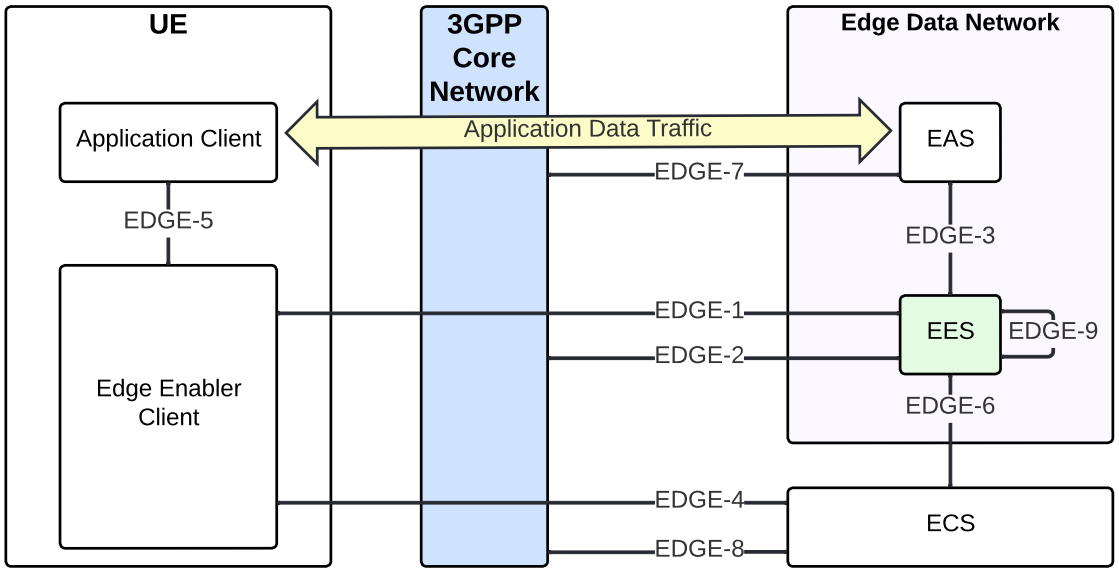


Figure 4.5: 3GPP edge computing enabler architecture, adapted from TS 23.558[5]

The EEC is located within the UE, meaning that the UE is edge-aware. This is a major difference to the ETSI MEC architecture, where normally the UE is not aware of the edge. The Application Client (AC) is the application which consumes edge services. The interaction between the AC and the EEC (EDGE-5) is implementation specific. From a software developer perspective, this may be a library which helps application developers to utilize the edge without the need to understand the whole architecture in an operator's network.[5]

The ECS provides supporting functions for the EEC, such as information how an EES may be reached and which area it serves. To accomplish this, an EES registers on the ECS. The EAS is the actual application running in the edge DN and is serving the AC. It registers on the EES and is able to communicate time and location constraints. It may also interact with the 5GC, but this is optional. Depending on the underlying Edge Hosting Environment, the EAS is a container or virtual machine which can be started, terminated and relocated on demand.[5]

The only elements providing APIs in the enabler layer are the EES and the ECS. The other elements consume these and the APIs exposed from the 5GC.

4.3.3 Synergy between ETSI MEC and 3GPP EDGEAPP

As discussed in the Sections 4.3.1 and 4.3.2, the ETSI MEC and 3GPP EDGEAPP are different approaches to solve similar issues. However, this does not mean that they replace each other, but rather that there are synergies between the two architectures, as depicted in Figure 4.6. Especially the central piece of both architectures, the MEP or respectively the EES have very similar responsibilities.

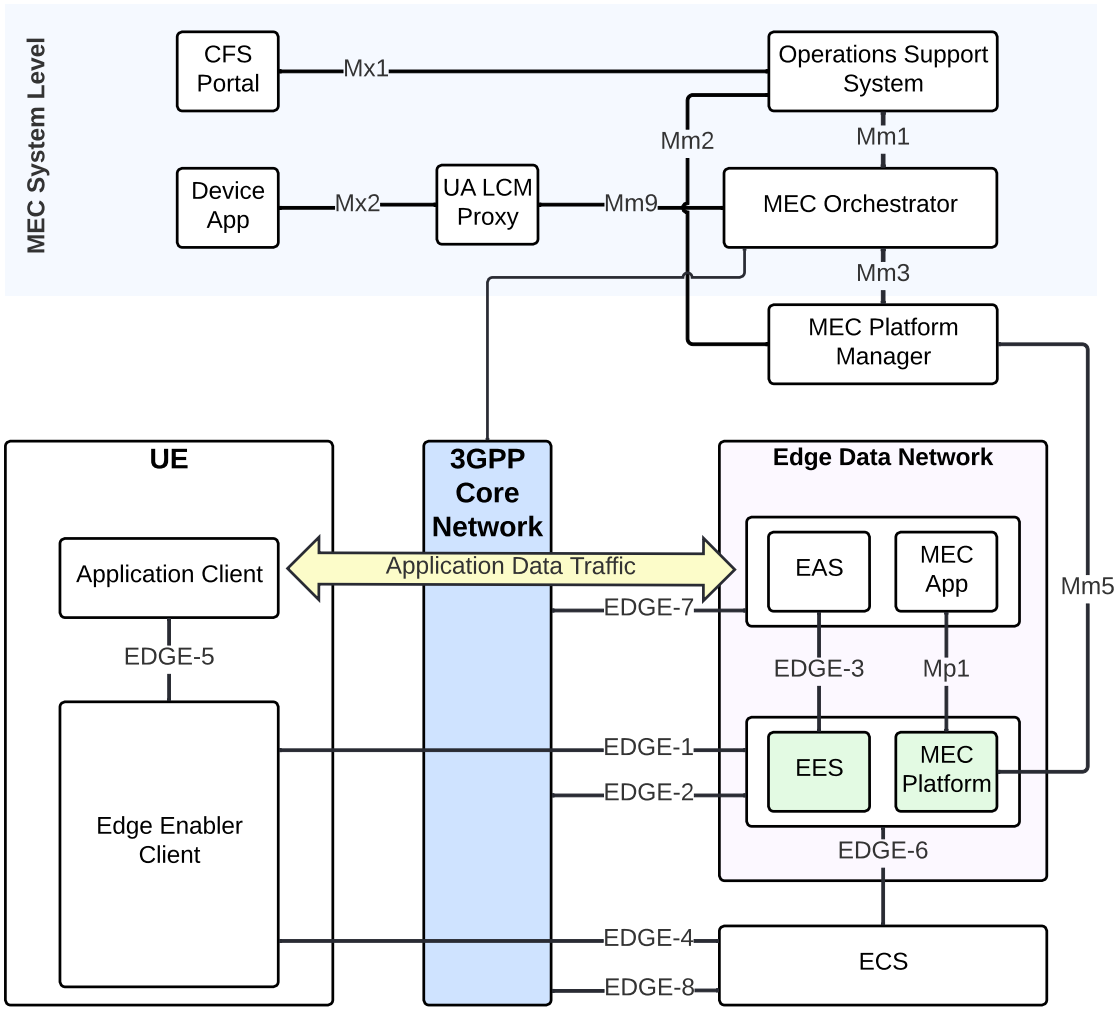


Figure 4.6: Synergies between ETSI MEC and 3GPP EDGEAPP, adapted from TS 23.558[5]

Therefore, an actual implementation of either could expose both APIs.[87][5] This would allow for an interworking between the different approaches. The same applies to the MEC application and the EAS. The advantage of supporting both architectures is that already existing MEC applications which use the APIs from the Mp1 reference point can be integrated into the EDGEAPP architecture.

Figure 4.6 shows that the orchestration and management components from the MEC architecture may be used instead of the ones specified by the 3GPP. This exemplifies the interoperability between the two approaches. Even though the underlying network and Edge Enabler Layer is 3GPP-centric, the orchestration of these components may use another architecture. ETSI White Paper #36[91] describes possible deployment options to harmonize the 3GPP and the ETSI architecture. It also discusses the major difference

between the architectures: Edge awareness. While the ETSI MEC architecture requires to configure DNS rules to route the traffic to the correct edge application, the 3GPP approach uses the EEC to discover a suitable EAS. According to the white paper, it is possible to use the simpler DNS based mechanism and still support the 3GPP Edge Enabler Architecture without using the EEC.

4.4 Interworking with the 5GS

The SA 5GC has been designed with edge computing in mind.[22] Compared to the previous generation it offers standard-compliant procedures and interfaces how to enable edge computing within the core network. These are described in this chapter.

4.4.1 Architecture

The initial 4G core network (EPC) architecture was centralized, meaning that the control plane and the user plane was always co-located. In 2017, 3GPP Release 14[W6] introduced CUPS. It allows to have the purely control plane components (such as the Mobility Management Entity (MME) and the Home Subscriber Server (HSS)) in a central location and split the Serving Gateway (SGW) and Packet Data Network Gateway (PGW) in a user plane and control plane part, which can be deployed in different locations. This architectural split is already built-in in the 5GC, namely with the separation between SMF and UPF. This allows to deploy UPFs anywhere in the network, while the control nodes (such as AMF, SMF and PCF) are hosted in a central location. Without CUPS, all the user plane traffic is sent to the same central location before being routed to a data network such as the internet. As discussed in Section 4.1, this corresponds to the transmission time t_{mno}. As the overall goal is to reduce the latency, the centralized architecture contradicts this edge computing requirement.

As MEC initially targeted the centralized 4G architecture, it has been designed as an add-on to mobile networks. The ETSI White Paper #28[89] discussed first in 2018 how to bring MEC into an SA 5GC. As depicted in Figure 4.5, the EDGE-2 interface of the EDGEAPP architecture is specified by the 3GPP. This is the counterpart of the unspecified MEC Mp2 interface. In fact, the approach described by the ETSI[89] has been standardized by the 3GPP in Release 17 in the EDGEAPP architecture[5]. The relevant components shall be deployed as an AF in the 5GC. An AF is able to influence traffic steering and even UPF selection, as defined in 3GPP TS 23.501 in Chapter 5.6.7[22]. Therefore, for the rest of this chapter, when not otherwise specified, "AF" could either mean a MEC platform and/or a 3GPP EDGEAPP EES.

The AF may be a trusted or an untrusted component. Depending on the scenario, an AF is either able to directly influence traffic routing using the PCF (trusted) or needs to connect via the Network Exposure Function (NEF) first. Figure 4.7 depicts this scenario. The interfaces towards the NEF are represented with dashed lines, indicating that this component is optional. Both the AF and the NEF use the `Npcf_PolicyAuthorization`

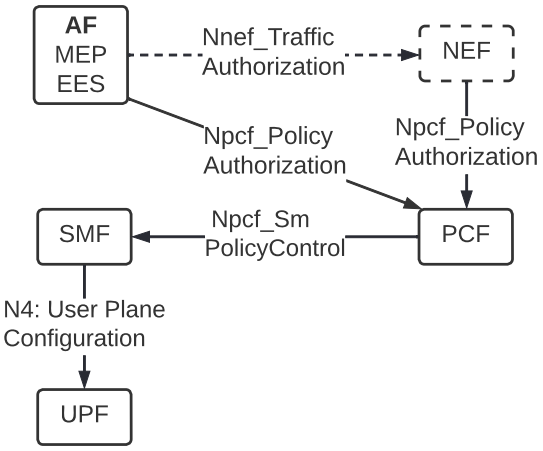


Figure 4.7: Architecture for AF-influenced traffic routing for single UE

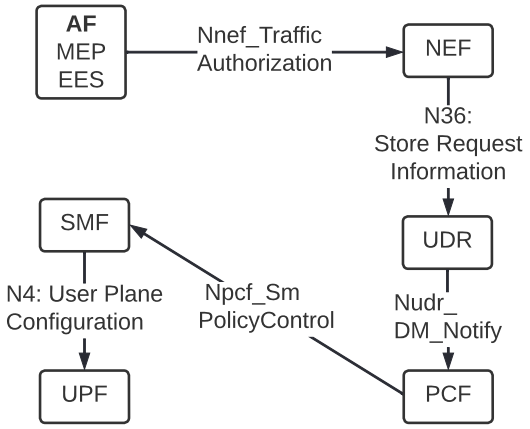


Figure 4.8: Architecture for AF-influenced traffic routing for multiple UEs

API to influence traffic routing in the 5GC. The reason to use the NEF is security: The NEF provides authentication capabilities towards the AF and acts as an external gateway to access 5G core functionality, similar to an API gateway. However, this is only valid when the AF targets one distinct UE, meaning that the edge platform wants to enable edge computing for a specific user, as specified in 3GPP TS 23.502[92]. When targeting multiple UEs at once, for example with a pre-defined user group identifier, the architecture depicted in Figure 4.8 shall be used.[92] Here, the AF sends a request to the NEF, which then updates the profile of subscribers in the UDR. The UDR notifies the PCF of changes to these subscribers.

From the author's point of view, the architecture depicted in Figure 4.7 could be used for both scenarios, with or without NEF. In the end, it may be possible for the AF to send distinct requests to the PCF for each UE. However, this may result in higher overhead. Another solution would be to adapt the standard to allow a user-group identifier in the Policy Authorization API. Nevertheless, the update of the SMF needs to happen for each UE in any case.

When the PCF receives the traffic influence information from the AF, it uses this information to update the policy association towards the SMF, which is described in Section 4.4.2. It is important to consider the timeline of the requests. In the scenarios of Figure 4.7 and 4.8, the AF requests to influence the traffic of already existing PDU sessions. The scenario where the Edge Enabler Layer requests traffic routing for future PDU sessions is not covered in the standards.[22][92] The reasoning behind this may be that the 3GPP EDGEAPP requires the EEC to interact with the EES, which can only happen after the UE has a data connection. This scenario can, however, easily be adapted to also support scenarios where the UE is not edge aware: The AF uses the same request towards the PCF, which then either: (1) updates existing policy associations or (2) stores the request information and considers it upon initial PDU session establishment.

When the SMF receives the policy update or the initial policy rules, it needs to select an appropriate UPF and instrument it accordingly to route the user plane traffic. The details of this procedures are described in Section 4.5.

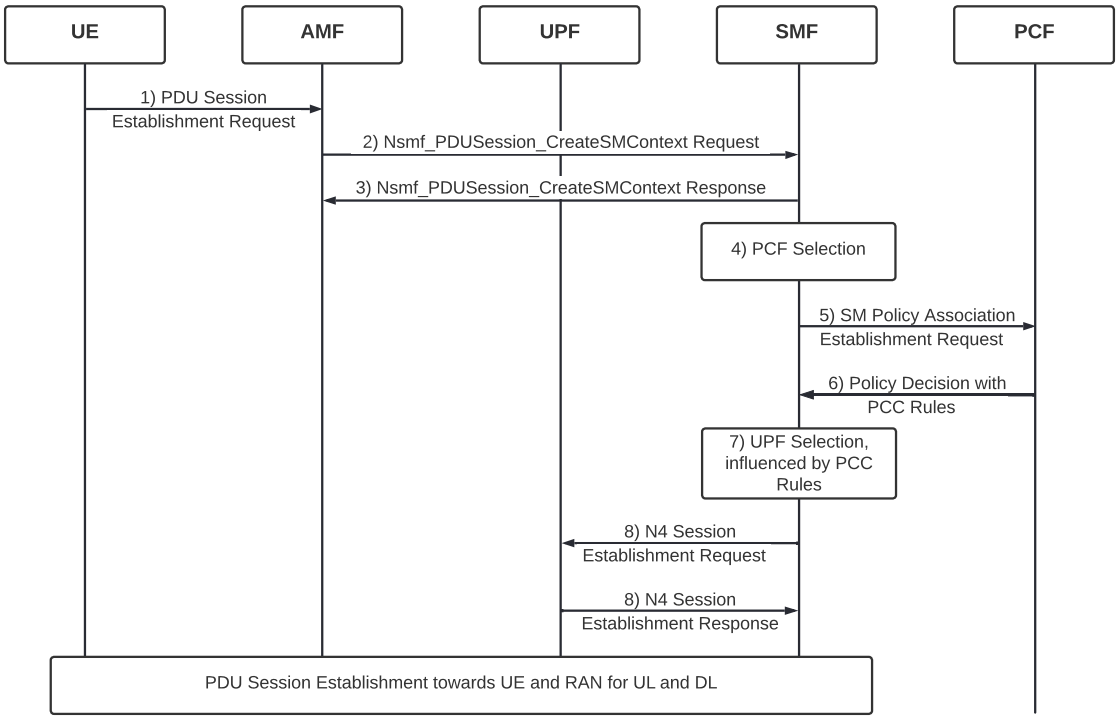


Figure 4.9: Simplified PDU session establishment request with Policy Association, adapted from 3GPP TS 23.502[92]

Figure 4.9 shows a simplified version of an initial PDU session establishment request, adapted from 3GPP TS 23.502[92]. The session establishment happens after a UE is already connected, registered and authenticated towards the network. For simplicity, the RAN is omitted. All requests from and towards the UE are sent over the air interface. The UE sends a PDU session establishment request to the AMF (1), which forwards the essential information to the SMF (2). In step (3), the SMF sends a response to the AMF, which indicates that the session establishment is acknowledged or rejected. It is worth to mention that between steps (2) and (3), the SMF queries subscription data from the UDM in case it is not locally available. The steps (4)-(7) are optional in the standard, but are essential for edge computing. First, the SMF needs to decide which PCF to use for the request. In the simplest case, only one is available. Otherwise, the SMF may decide based on local configuration or information from the Network Repository Function (NRF). In any case, the SMF sends a request to the PCF to create a Session Management Policy Association (5). This request contains context information such as the subscriber identity (Subscription Permanent Identifier (SUPI)), Data Network Name (DNN), and the slice information. The PCF queries its local configuration or the subscriber profile stored in the UDR to find a matching policy for this SUPI. In case the AF requested

to influence traffic routing, the PCF should consider this and select the correct policy. As noted above, this scenario is not described in the standards. The PCF replies with a policy, as described in more detail in Section 4.4.2. Upon receiving the policy, the SMF uses this policy to influence the selection of the UPF or multiple UPFs. This is the most essential step to enable edge computing, as all the user plane traffic traverses through the UPF. In order to enable edge computing, a UPF which is close or even co-located with an edge data center should be selected. Additionally, the user location (e.g., the cell id or Tracking Area Code (TAC)) shall be taken into account for selecting the correct UPF. After selecting the correct UPF(s), the SMF uses the `N4` reference point and PFCP to instrument the UPF accordingly. Different options for UPF selection are described in Section 4.5.

4.4.2 SM Policy Control API

Note: If not otherwise stated, the concepts described in this section are specified in the 3GPP TS 29.512[36].

The SM (Session Management) Policy Control API is used to provision session related Policy and Charging Control (PCC) rules to the SMF. The PCC rules are defined in 3GPP TS 23.503[93]. A PCC rule consists of information how to identify user plane traffic, together with a set of rules how this traffic shall be handled and charged. PCC rules cover a wide area of applications and are, e.g., used for charging, QoS, access network reporting, usage monitoring and AF influenced traffic steering. In the scope of this thesis, only the AF influenced traffic steering aspect is considered as policy decisions. The information how to differentiate the edge traffic from the internet traffic is necessary as well in scenarios where an UL CL is used, as described in Section 4.5.3.

The SM Policy Control API is specified in 3GPP TS 29.512[36]. It is a HTTP REST API which allows to provision PCC rules for different users and use cases. The data is exchanged using the JSON format. The API provides routes to create, update, get and delete policy associations. A policy association is distinct per PDU session and is created by the SMF during PDU session establishment (see Figure 4.9). Examples for requests and the corresponding JSON body can be found in Section A.1.

Creation of an SM Policy Association

During establishment of a PDU session, the SMF sends a request to create a policy association with the PCF. The SMF includes information of the `SmPolicyContextData` type. The context data must contain a SUPI, DNN, PDU session ID, PDU session type (IPv4, IPv6 or dual), notification URI and slicing information.

The notification URI is an HTTP endpoint on the SMF and it indicates that the SMF subscribes to updates from the PCF. This enables the PCF to notify the SMF in case a policy changes.

Apart from the mandatory information, the context contains many more optional fields. Interesting fields from the edge computing perspective are the Radio Access Type (RAT) and the user location information. These fields would allow the PCF to select different policies based on the user location. It would be possible to only allow edge computing when the UE is connected to a 5G NR, but use default traffic rules (via the internet) when the UE is connected over WiFi or any other mobile access (such as 4G). Additionally, the user location information may be used to select a different UPF based on the location. This enables the PCF or the AF – using the spatial validity feature of the Policy Authorization API – to enforce location based routing.

Upon receiving the context information, the PCF selects a policy. The standard does not define the details of the policy selection. To enable edge computing scenarios, the following hierarchical selection options may be considered:

- SUPI based selection: This is mandatory, as the SUPI is a key parameter in the request.
- DNN based selection: May be used when the SUPI does not match any policies in the PCF
- Slice based selection: May be used when neither the SUPI, nor the DNN match any policies. This is useful in a scenario where there is a dedicated slice for edge computing, which simplifies the provisioning for the PCF.
- Default policy: A default policy when no other policy matches. This policy is used to prevent a failure in session establishment in case of a provisioning error. It is arguable whether this is a good approach, as it hides errors in the provisioning of policies in the PCF, but on the other hand does not interfere with the user experience.

These selection methods can be enhanced to support additional features such as selecting different policies based on the user location and/or RAT.

When the PCF has selected an appropriate policy, it sends the reply to the SMF. The reply contains data with the `SmPolicyDecision` type. The policy decision consists of a set of PCC rules together with policy decisions. For edge computing, only the policy decision type `TrafficControlData` is considered. The PCC rules contain mandatory information how to identify the traffic, a unique rule ID and a reference to the traffic control data, which should apply for this specific traffic.

The traffic control data consists of a unique ID (which is referenced in the PCC rule) and information how to route the traffic. Two important fields are the redirect information and the route to locations. The redirect information allows the 5GC to redirect packets to another IP address or even HTTP URL. As an example, requests to `1.2.3.4` are redirected to `5.6.7.8`, where the IP header is adjusted accordingly. While this is an interesting feature, it does not satisfy the latency requirements of edge computing. The

reason is that the redirect information does not give the SMF any indication which UPF shall be used. In the worst case, the SMF selects a UPF in a central location, which enforces the redirection and the packets are routed back to the edge of the network.

The route to locations information in the traffic control data allows to define for each Data Network Access Identifier (DNAI) either route information or a policy ID. The policy ID needs to be pre-configured on SMF, so that it has sufficient information which UPF to select and how to set up the routing. The route information is an IP address and port combination. Section 4.5 discusses how this information may be used to select an appropriate UPF and set up the routing to the edge data network.

The PCF can inform the SMF that it would like to receive updates based on specific triggers. The standard defines many triggers such as a change of the access type, IP address, RAT type, user location or QoS. The same procedure is also used to report data usage for charging purposes. In the context of edge computing, the change of access type, RAT and user location can all be used to guarantee that the selected data network is still close to the user. User mobility may trigger the relocation of an edge application and, therefore, the routing needs to be adapted as well.

SMF Update of an SM Policy Association

The SMF is able to update an SM Policy Association. This happens when one of the triggers subscribed by the PCF are met on the SMF. For example, when the UE receives a new IP address, the PCF is notified. Based on this information, the PCF is able to re-decide the policy association. The PCF response for an update is the same data structure as for creating a policy association. The difference is that the PCF only informs about changed values. Therefore, when the PCF decides not to change anything, the response is an empty object. Apart from that, the same procedures apply. This mechanism allows the PCF to enforce that the user plane is still routed to the closest UPF, even if the device changed its location, RAT or access type. It is also possible to enable edge computing only for a specific technology. The PCF can enforce this by updating the PCC rules accordingly.

PCF Update Notify

As described, the SMF provides a notification URI when it requests an initial policy association. The PCF is able to use this URI to update the SM Policy Association, hence the PCF is acting as a client in this scenario. The content of the message is of the type `SmPolicyDecision`, the same as used in the response to an update of an SM Policy Association. Figures 4.7 and 4.8 show the scenario where this mechanism is used. The AF instruments the PCF to enforce a specific traffic routing, as described in Section 4.4.3. When a UE that matches the criteria has an ongoing PDU session, the PCF needs to notify the SMF so that it can update the traffic routing accordingly.

PCF Termination of an SM Policy Association

The PCF can also request the termination of an SM Policy Association, using the notification URI provided by the SMF. In this case, the SMF also triggers the termination of the corresponding PDU session.

Deletion of an SM Policy Association

The SMF can request that the PCF deletes an existing SM Policy Association. This scenario happens when the UE de-registers from the network and, therefore, the PDU session is terminated.

Retrieving an SM Policy Association

The SMF can also request to receive an existing policy association based on its ID. This may be the case when the SMF does not store the information locally or needs to retrieve it for another reason, i.e., for failure recovery.

4.4.3 Policy Authorization API

Note: If not otherwise stated, the concepts described in this section are specified in the 3GPP TS 29.514[37].

The Policy Authorization API of the PCF is used to provide the PCF with the necessary information for its policy decisions. It does not contain any PCC rules, but the information from this API together with the pre-configured PCC rules of the PCF are considered in the policy decision of the PCF. This API is defined in 3GPP TS 29.214[37]. As all the other service-based interfaces in the 5GC, this interface is an HTTP REST API.

Creation of a Policy Authorization

The AF is able to create a policy authorization towards the PCF. In the context of edge computing, the information contained in this request is the DNAIs used for a specific user, identified by IP address and/or IMSI. Upon receipt of this request, the PCF has to act accordingly. When a PDU session matching this user is not yet established, this request shall be stored and considered in future decisions. This means that the PCF needs a temporary storage for policy requests from an AF. As it is not defined in the standards how the implementation shall handle this, the author proposes to have a policy decision service in the PCF, considering the pre-configured PCC rules and the incoming requests from an AF. These decisions shall be stored for the different users and selection options.

When on the other hand the PCF already has an established SM policy context, it needs to invoke the Update Notify API towards the SMF. In the case of edge computing, this would require the SMF to eventually re-select and re-anchor different UPFs.

Update of a Policy Authorization

The AF is able to update the policy information which it has previously given to a PCF. A scenario where this is useful is when the AF has registered to events on the SMF. When, e.g., the user location changes, the AF acting as an EES or MEP can move an edge application from one DN to another. Then, it has to inform the SMF over the PCF to re-anchor the subscriber's PDU session.

Deletion of a Policy Authorization

The AF is able to delete its association with a PCF, eventually triggering the PCF to remove the information from the AF and re-decide its earlier policy decision. This may be the case when an edge application does not exist anymore due to configurations on the orchestration level.

Notifications

The AF can subscribe to or unsubscribe from PCF notifications. There are different notification triggers available to the AF, such as a change of access type or PLMN. When the AF subscribes to notifications, it is required to provide an API endpoint, where the notifications shall be received. This is an alternative configuration, where the AF does not receive notifications from another NF (such as the SMF) directly, but over the PCF.

4.5 UPF Selection and User Plane Traffic Routing

Section 4.4 describes how the control plane is able to influence the traffic routing decision. This process mainly involves the AF, NEF, PCF and the `Npcf` reference point of the SMF. This section focuses on the interactions between the SMF and the UPF and the different possibilities to route traffic to a local edge DN.

4.5.1 Architecture and Access Models

The 3GPP Technical Report TR 23.748[94] is a study on edge computing enhancement in 5G. Based on this study, there are two reasonable architectures, which result in three different deployment options.

Figure 4.10 shows the 5G architecture for edge computing using one UPF. This UPF is located in the edge of the network. There is a GTP tunnel on the `N3` reference point between the gNB and the UPF. Traffic to and from the DN on the `N6` reference point is not encapsulated in a GTP tunnel, hence the gNB's GTP tunnel terminates here.

Distributed Anchor Point

The distributed anchor point scenario is used in the deployment option depicted in Figure 4.10. The UPF is located near or within the edge DN.[94] The SMF is responsible to

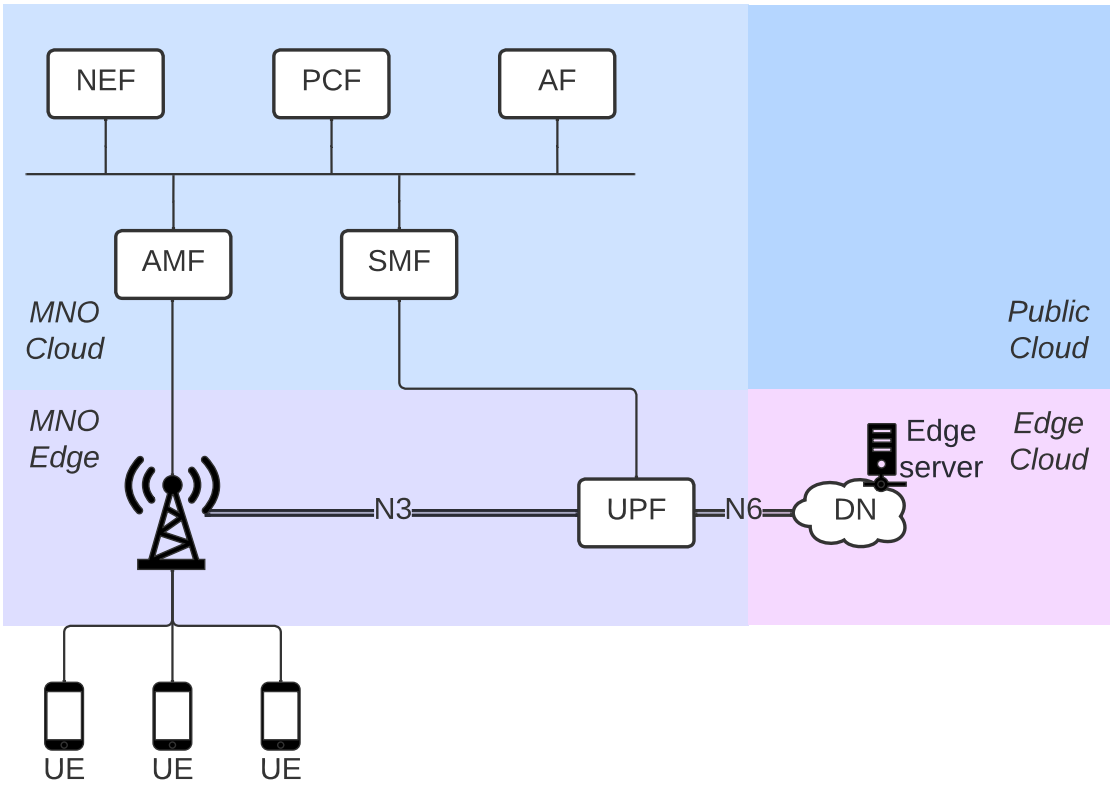


Figure 4.10: 5G edge computing architecture with distributed anchor point

select the correct UPF based on local configuration or PCC rules. This is described further in Section 4.5.3.

All the traffic from the UE is routed to the edge DN.[22] Therefore, if the UE also needs to connect to the cloud or other DNs (such as the internet), routing capabilities must be provided within the edge DN. This may be out of scope for the MNO, when the DN is operated by a third party.

Multiple PDU Sessions

The multiple PDU sessions scenario is an enhancement of distributed anchor point. In this case, there is a distinct PDU session for each application.[94] This means that the client establishes two PDU sessions, one for the edge DN with a local UPF and one for the central DN with a central UPF. Thus, the client has two different source IP addresses and is responsible to use the correct interface when routing traffic to the edge DN. This scenario requires the UE to be edge-aware, as the routing tables on the device need to be adapted accordingly. One might say that here the UE is acting as a Uplink Classifier (UL CL), as the IP routing table in the UE is used to select the correct interface.

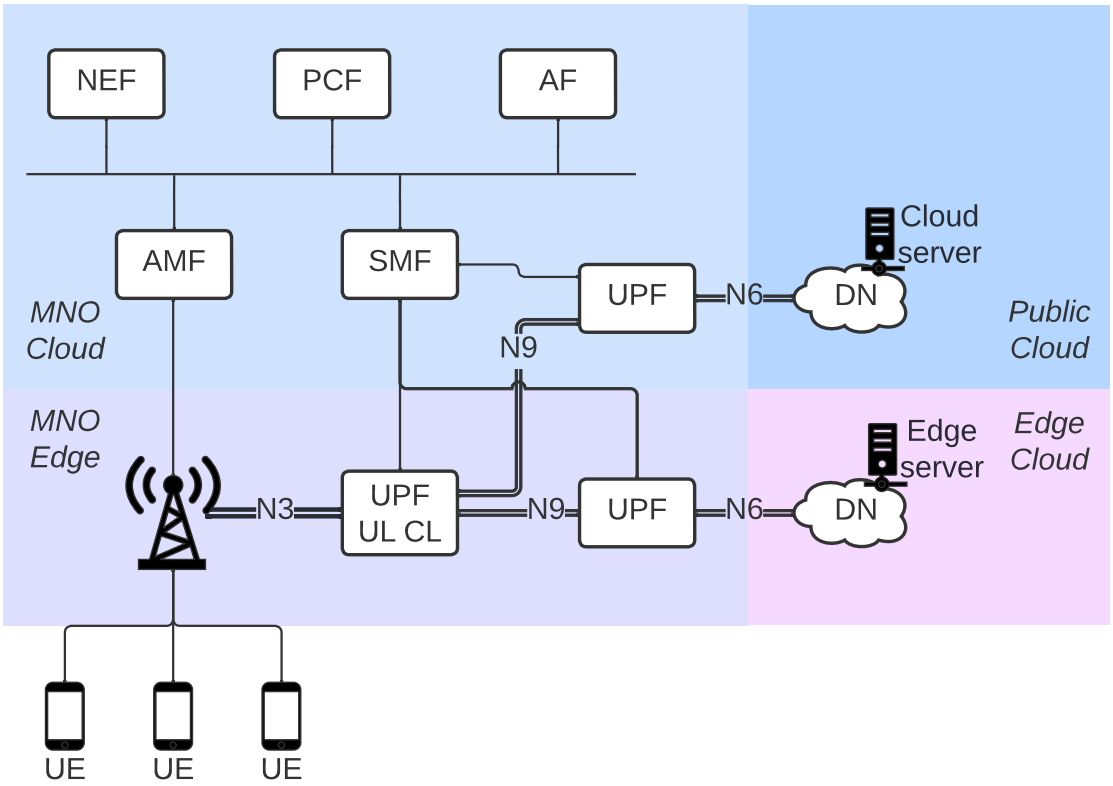


Figure 4.11: 5G edge computing architecture with UL CL

Session Breakout

The session breakout scenario is depicted in Figure 4.11. It shows the 5G architecture for edge computing using an UL CL. One UPF is acting as a UL CL and connects to the gNB via the N3 reference point and to two other UPFs via the N9 reference point. Each N3 or N9 reference point consists of a distinct GTP tunnel. The UPFs which connect to the internet DN respectively the edge DN over the N6 reference point are referred to as PDU Session Anchor (PSA). In this architecture, the UE is able to send and receive packets from the internet, but also from the edge DN. This architecture requires having multiple PSAs within one PDU session. This is described in 3GPP TS 23.501[22] in Chapter 5.6.4 and requires the UL CL or the Branching Point (BP) feature.

The UL CL is able to route the UL traffic based on flow information such as the destination IP address towards one of the PSAs.[22] It is also possible to have a default route and a more specific route based on an application. In the context of edge computing, a specific IP address or IP range may be routed to the edge DN, whereas all the other traffic is routed towards the internet DN.

The BP feature also allows to route traffic to different PSAs. The crucial difference compared to a UL CL is that the BP requires the use of IPv6 and especially IPv6

multi-homing. This means that the client needs to have different IPv6 prefixes and the BP routes towards one of the PSAs based on the source IPv6 prefix.[22] Therefore, the client needs to be aware of this and send traffic destined to the edge DN with the correct prefix.

Comparison

Table 4.2 compares these three different deployment options. When requirements are indicated as "none", it means that there are no additional requirements related to edge computing. The mandatory procedures for PDU session establishment on the SMF and N3/N6 packet forwarding on the UPF are required in any case.[22][92]

	Distributed Anchor Point	Multiple PDU Sessions	Session Breakout
# UPFs	1	2	3
# GTP tunnels	1	2	3
# UPFs per PDU session	1	1	3
# PFCP messages	4	8	min 12
Edge routing	direct	direct	via UL CL
Internet routing	no	direct	via UL CL
UPF requirements	none	none	UL CL feature
SMF requirements	none	2 PDU sessions	UL CL feature
Client edge awareness	no	yes, select edge DNN	no

Table 4.2: Comparison between the different edge computing deployment options

The distributed anchor point is the simplest option, as there is no difference to a central PSA from the SMF's perspective. There is one UPF and one GTP tunnel and there is no change in the PFCP message procedures. The sole reason why this is an edge computing deployment is because the UPF is physically located at the edge. While the multiple PDU sessions approach requires SMF support, it is relatively similar to the distributed anchor point. Each PDU session is set up in the same way and also the PFCP procedures are the same, targeting different UPFs. Further, this is a basic requirement for any 5GC, as Voice over NR (VoNR), the 5G IP-based voice service, is using a distinct PDU session as well. When using this option, the UE must be able to route the traffic to the correct PDU session, i.e., it performs a classification. The session breakout option requires substantial changes on the SMF as well as on the UPF. As depicted in Figure 4.11, the UPF deployment is more complex, increasing the required number of UPFs per GTP tunnel from 1 to 3. Also, the UPFs have to support the N9 interface for UPF interworking, although this is just a GTP tunnel for the UPF. While this option is the most complex, it also provides the most flexibility, an important quality for edge

computing. Possible solutions how the SMF can support this scenario are described in Section 4.5.3 and the updated PFCP procedures are described in detail in Section 5.3.4. The impact of the different deployments on real-world scenarios are discussed in Chapter 8.

4.5.2 Re-Anchoring

All the different deployment options from Section 4.5.1 have the same requirements towards mobility: The UPF for the edge (either UL CL or PSA) might not be close to the UE anymore when the UE changes the cell or even the tracking area. In this case, re-anchoring should be used to route the traffic to the new local UPF. The mechanisms for re-anchoring are the same as the ones used for Session and Service Continuity (SCC) modes 2 and 3.[94]

In the 5GC there are three SCC modes[22]:

- Mode 1: The UPF is acting as PSA. Upon UE mobility, the PSA stays the same and, therefore, the UE keeps the same IP address. As the UPF does not change, the physical distance between the new gNB and the old UPF may increase and, therefore, also the latency t_{mno} increases. As this may contradict the edge computing latency requirements, this mode is not usable in this context.
- Mode 2: The network releases the PDU session and instructs the UE to establish a new PDU session towards the network. The consequence is that there is a period where the UE does not have a PDU session and, therefore, no connection to a DN.
- Mode 3: The network instructs the UE to establish a new PDU session in parallel to the existing one. The 5GC maintains the PDU session with the old PSA for a defined period of time until it is released.

When the SMF decides that re-anchoring in mode 2 is required, it informs the UE that it needs to release the PDU session via the AMF and the RAN. The message contains a code that indicates to the UE that it shall establish a new PDU session to the same DN.[92] Essentially, the new PDU session request by the UE triggers the SMF to start the UPF selection anew, and the SMF is able to take the new information into account, especially the location information. When the distributed anchor point is used, a new UPF is selected as the PSA. In the session breakout scenario, it is likely that the internet PSA stays the same, as there are no latency constraints on this path. However, the UL CL and the edge PSA shall be re-located, as the selection of these is required to minimize t_{mno}. When using multiple PDU sessions, only the edge PDU session has to be relocated. Therefore, if this is supported by the SMF, it is a good approach to use SCC mode 1 for the internet PDU session and SCC mode 2 for the edge PDU session.

The scenario using SCC mode 3 is similar to SCC mode 2. The crucial difference is that instead of releasing the current PDU session immediately, another PDU session is

established in parallel and the old PDU session is released after a period of time. This effectively doubles the amount of parallel PDU sessions. Therefore, if the multiple PDU sessions access model is used, there are four PDU sessions at a certain time.[92] Again, using SCC mode 1 for the internet PDU session can reduce this overhead. In this case, the internet PDU session is not released. Thus, there are three parallel PDU sessions: The unchanged internet PDU session, the old edge PDU session and the new edge PDU session. When a UL CL is used, the SMF may select three new UPFs, one UL CL and two PSAs. Thus, at a certain time, there may be six UPFs involved for a single UE. As one can see, the combination of SCC mode and edge computing deployment option can significantly increase the complexity and signaling overhead.

Table 4.3 compares the SCC modes for the three edge computing access modes. The service interruptions are minimal for mode 1 and mode 3, as the UE keeps its PDU session and only some packets are lost during the handover on the air interface. This process is much faster than establishing a new PDU session. Hence, mode 2 has a more noticeable service interruption.

	SCC Mode		
	Mode 1	Mode 2	Mode 3
UPF changes	no	yes	yes
Service interruptions	minimal	noticeable	minimal
Edge relocation	no	yes	yes
AF notifications	no	yes	yes
Max parallel PDU sessions Distributed anchor point	1	1	2
Max parallel PDU sessions Multiple PDU sessions	2	2	4
Max parallel PDU sessions Session breakout	1	1	2

Table 4.3: Comparison of the three different SCC modes for edge computing

In edge computing, the support for SCC mode 2 and mode 3 entails that the edge enabler layer is able to relocate edge applications as well. After the re-anchoring, the user plane traffic is routed to a new edge DN, as t_{transit} must be minimized. Therefore, to ensure ongoing service, the EAS needs to be relocated before. To coordinate this process, the AF can instrument the SMF (via the PCF) to send early or late notifications for re-anchoring scenarios. When early notifications are used, the SMF informs the AF of the planned change before it communicates with any other component. The AF may accept or reject the PDU session change. Upon accept, the procedure continues as described. Upon

reject, the SMF does not modify the PDU session. Late notifications are used to inform the AF that a re-anchoring has happened.[22][92]

This feature allows the edge enabler layer to decide if re-anchoring should happen or not. This may depend on different constraints. If the latency constraints are still expected to be met even if the UE moves further away, the AF may reject the re-anchoring. As the AF is also controlling the SMF's UPF selection, it is able to make this decision. Also, if the EAS is stateful and a relocation is not feasible or unsupported, the AF can reject. This decision is a trade-off between simplicity and latency constraints.

4.5.3 UPF Selection and Instantiation

Independent of the three different deployment options described in Section 4.5.1, the SMF should always select the closest UPF(s) for edge computing. Closest in this context means that the UPF is in physical proximity of the base station that serves the UE. The 5G core network standards TS 23.501[22] and TS 23.502[92] describe requirements for UPF selection, but do not specify in detail how this shall be done. Based on the procedures described in Section 4.4 and the standardized procedures in 3GPP TS 23.501[22] and TS 23.052[22], the author of this thesis proposes three solution options:

- Pre-configured steering policy
- Utilize N6 routing information
- Dynamic UPF selection based on NRF information

As the 5GC allows NFs to dynamically be instantiated with the help of the NRF, this is considered as well in the solution options.

Pre-configured Steering Policy

As described in Section 4.4.2, the SM Policy API allows to specify a steering policy ID per DNAI. One option to realize edge computing is to locally configure steering policies on SMF. These steering policies should define the UPF endpoints which are used within a specific policy. Additionally, the role of each UPF shall be configurable to decide if the UPF is a UL CL, edge PSA or central PSA. This is sufficient to create a mapping between DNAI and UPF, allowing the SMF to choose one (or multiple) UPFs based on DNAI. However, this does not solve the requirement to select the closest UPF. It is possible that the PCF does not take the user location into account and informs the SMF of all the possible edge DNs by providing a list of DNAIs. In this case, the SMF still needs to choose which DNAI shall be selected for a specific PDU session. Therefore, there is the need to pre-configure this mapping in the SMF as well. Another solution is that the AF or PCF receive updates to the user's location and act accordingly. This means that the PCF only signals the DNAIs which are in the current scope. In case of

mobility, the PCF updates the traffic description and replaces the DNAI with the one located closest to the user.

```
ul_cl_policy_1:
  dnai:       internet
  ul_cl:      true
  next_hop_1: PSA1 (N9)
  next_hop_2: PSA2 (N9)

edge_policy_1:
  dnai:       edge_dn_1
  next_hop_1: Edge DN (N6)

internet_policy:
  dnai:       internet
  next_hop_1: Internet DN (N6)
```

Listing 4.1: Example of a local steering policy in SMF

An example steering policy in SMF is shown in Listing 4.1. Each policy has a unique ID and defines a DNAI. The first is used to enforce a UL CL. Here, either the internet DNAI or the edge DNAI should be configured. The `edge_policy_1` is used to create a tunnel between the UL CL and the UPF and to enable routing in the `N6` interface, where the GTP tunnel is removed. The `internet_policy` is used to route traffic to the internet. The example configuration may be used to create a scenario as depicted in Figure 4.11.

It is important to note that the SMF does not decide which rules to apply. This configuration is only used to create a mapping between DNAI and UPFs and to configure the architecture. The PCF uses the SM Policy API to inform the SMF which rules to apply. The identifiers of the rules need to be contained in the traffic description in the `PolicyDecision`. Also, the PCC rules contain the information how to identify traffic.

An alternative to the presented solution is to not use the DNAI as identifier, but to configure policy steering rules on a higher level. In that case, the PCF sends only one traffic description with a distinct rule and the SMF needs to pre-configure each possible combination of different UPFs.

Utilize N6 Routing Information

As described in Section 4.4.2, the API allows to either define a pre-configured policy ID or `N6` routing information. In the current version, this routing information is an IP address and a port. Therefore, there is no need to configure local steering policies in the SMF, as the mapping between DNAI and next hop (UPF) is already defined in the `PolicyDecision` data type from the PCF.

Nevertheless, as the N6 routing information is only an IP address, much of the information is implicit. The SMF needs a logic to decide which UPF shall be the UL CL and needs to create the architecture based on a combination of:

- Number of next hops (DNAIs)
- Number of PCC rules (which define the traffic identification)
- Available UPFs in the system

Dynamic UPF Selection Based on NRF Information

As described in Section 2.1, the NRF has a vital role in the 5GC: It allows for discovery of components similar to a DNS, but with the added value that each component can define additional information.

The NF repository service is described in 3GPP TS 29.510[34]. It defines the `UpfInfo` data structure, which is stored in the NRF and can be retrieved by any other NF in the 5GC. This data structure allows to define for each UPF the DNAIs it serves, together with a mapping between DNAI and network instances. The network instance is an identifier of a network interface of the UPF and can be signaled with the PFCP protocol. This mapping is introduced in V16.8.0 of the standard, hence the reference is V16.8.0 instead of V16.5.0 as for most of the other core-related specifications. To enable edge computing, each edge UPF should configure only the DNAI for the corresponding edge DN. An example of the relevant parts of the `UpfInfo` can be found in Section A.2.

In this solution, the steering policy ID in the SMF can be used as well. The difference to the pre-configured steering policy option is that the configuration in the SMF does not define the architecture, as this is implicit in the UPF info from the NRF. The configuration is used to indicate to the SMF how this information shall be used. For example, a steering policy may indicate that a UL CL shall be inserted.

The `UpfInfo` item also allows to define the TACs that a specific UPF is serving. This information can be used by the SMF as well to ensure that the TAC of the user corresponds to the one of the UPF. This allows the SMF to select the UPF that is closest to the gNB, hence reducing the time t_{mno}.

Dynamic Instantiation of an Edge UPF

The use cases in this chapter rely on the assumption that all the components in the 5GC are controlled by a mobile operator. The nature of the edge computing enablement layers (such as 3GPP EDGEAPP and ETSI MEC) allows for dynamic instantiation of application servers in the edge DN. When the UPF is seen as an application server, it can be spawned when needed. This would allow the operator of the edge DN (which might be a different party than the mobile operator) to use its own infrastructure. For example, when the UPF of the operator does not support the UL CL feature, the edge solution

may instantiate one or more of its own UPFs per edge DN. This does only work when the dynamic UPF selection solution is used, as these UPFs cannot be pre-configured in the SMF.

Comparison

Table 4.4 compares the different UPF selection options. It is apparent that while the pre-configured steering policy option is very simple, it is also the least flexible. The dynamic option requires an NRF and thus also that the UPF registers towards the NRF and that the SMF subscribes for notifications. Hence, the complexity is high. The advantage is that this also increases the flexibility of the solution and it is the only solution that allows for a cloud-native deployment, as UPFs may be instantiated or terminated during the operation. The N6 routing information option is between these two options in terms of flexibility and complexity.

Furthermore, while the pre-configured UPF selection option seems simple in this example, its complexity increases with the number of UPFs. Manually configuring hundreds of UPFs in one or more configuration files on the SMF is error-prone and not well maintainable.

Option	Pre-configured	N6 Information	Dynamic
PCF required	yes	yes	yes
UPF configuration	SMF local	SMF local	in UPF
NRF required	no	no	yes
Flexibility	low	medium	high
SMF Complexity	low	low	high
UPF NRF Registration	no	no	yes
SMF NRF Registration	no	no	yes
Dynamic UPF Instantiation	no	might	yes

Table 4.4: Comparison between the three UPF selection options

CHAPTER 5

Edge Computing Prototype in the OAI

5.1 About the OAI

The OAI is an open source implementation of 3GPP components, supporting 4G and 5G. The goal of the OAI is to run on Common-Off-The-Shelf (COTS) hardware for all the computation nodes and on Software Defined Radio (SDR)s such as the ETTUS Universal Software Radio Peripheral (USRP) for all the radio communication.[9] The USRPs are high-performing SDRs that cover frequency ranges between 3 MHz and 6 GHz, as described by ETTUS[W17].

As the OAI targets to provide a full-stack 3GPP-compliant 5GS, it consists of a 5G RAN and a 5GC. In the context of this thesis, only the 5G aspects of the OAI are considered.

5.1.1 RAN

The OAI RAN supports several different SDRs, where the most popular ones are the USRP devices B210, X310 and N310.[9] Other SDRs are also supported such as the Benetel RRU[W18] for LTE and the AW2S RRUs for LTE and NR[W19].

The OAI RAN consists of the gNB, which implements the architecture described in 3GPP TS 38.401[95]. Following this, the gNB may be split into separate components, namely the CU, Distributed Unit (DU) and the RU. Within the OAI, the interface between the CU and the DU is the 3GPP-standardized F1 interface. The Small Cell Forum Femto Application Programming Interface (FAPI) interface is used between the MAC/RLC layer and the PHY layer and the O-RAN 7.2 interface is used between the CU and the DU, as shown in Figure 5.1.[9]

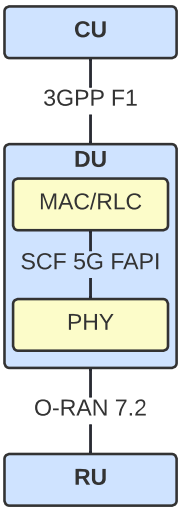


Figure 5.1: gNB functional split protocols, adapted from Kaltenberger et al.[9]

The OAI gNB supports all the main UL and DL channels defined in 3GPP TS 38.211[83][9]:

- Primary Synchronization Signal
- Physical Broadcast Channel (PBCH)
- Physical Downlink Control Channel (PDCCH)
- Physical Downlink Shared Channel (PDSCH)
- Physical Random Access Channel (PRACH)
- Physical Uplink Control Channel (PUCCH)
- Physical Uplink Shared Channel (PUSCH)

The MAC layer supports scheduling and uses the FAPI protocol to instrument the PHY layer. The RLC layer has been updated from the 4G version to support the 5G procedures.[9]

The RRC layer supports the procedures for the NSA Dual-Connectivity mode[9], especially on the X2 interface between the 4G eNodeB (eNB) and the 5G gNB.

The OAI RAN supports the NSA as well as the SA option.[9] As this work focuses on the interfaces inside the 5GC to enable edge computing, only the SA deployment option is considered.

5.1.2 5GC

The 5GC implements the basic functions which are necessary to establish a PDU session between the UE and the 5GS. As shown in Figure 5.2, the OAI 5GC provides the "core of the core", as described in Section 2.1.4. The NFs AMF, SMF, UDM, UDR, AUSF and UPF are implemented. Thus, the basic use cases registration, authentication and PDU session establishment are all supported. This fundamentally enables simulated UEs, as well as COTS UEs to have an internet connection. Additionally, the NRF is provided, which is a prerequisite for the solution described in this chapter. The NEF and NSSF are not used in this thesis. The SMF has been adapted to support the underlying edge computing use case, as discussed in Section 5.3. Further, the PCF has been created to support the edge-computing related procedures, which is described in Section 5.2. The author's contribution in the PCF and the created interface between SMF and PCF is the foundation for future use cases, such as QoS. The author has also contributed to the UPF to enable the different configurations necessary to support the edge computing deployment options presented in Section 4.5. The NRF was adapted to support the DNAI configuration in the `UpfInfo` data structure.

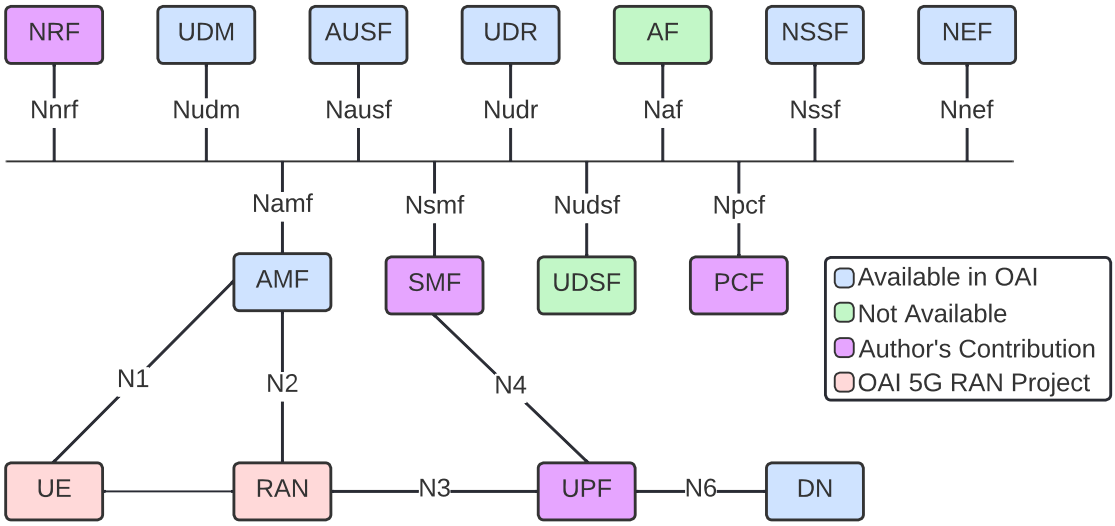


Figure 5.2: OAI CN architecture, adapted from openairinterface.org[W20]

Service Based Architecture (SBA)

All the 5GC NFs support the SBA, where the NFs provide services which are consumed by other NFs. Each NF provides a SBI for other NFs to consume.[9] Thus, the NFs communicate over the SBI, which is described in 3GPP TS 29.500[96]. The SBI provides RESTful services, therefore, the communication protocols are HTTP or HTTP2 and TCP. All the SBI specifications provide `yaml` definitions compliant with the `open-api` specification provided by swagger.io[W21]. This allows to use swagger `codegen` to generate code for the APIs. All the core components are developed in `C++` and `Pistache`

is used as REST web server. The documentation of Pistache can be found on the Pistache GitHub repository[W22].

5.1.3 Deployment

The deployment of the OAI consists of the gNB and the 5GC.

gNB

The work conducted by Mufutau et al.[97] shows that the OAI RAN can be deployed virtualized in a cloud environment. However, their results also show that there is a significant performance impact. Due to this reason, the OAI gNB is mostly deployed on bare-metal. This deployment works on any COTS hardware and it does not require specialized hardware. The server where the OAI is deployed, however, needs access to a radio unit, in most cases a USRP.

5GC

Due to the fact that the 5GC NFs do not significantly differ from other web micro-services, the deployment in a virtualized environment is more straight-forward. Therefore, all the core components can be deployed as Docker containers and Dockerfiles for each of them exist. Also, the latest stable and development images are published in the Docker Hub in the `oaisoftwarealliance` namespace[W23].

The OAI 5GCN provides `docker-compose` files to bring up all the different network functions with a default configuration that can be changed by setting the environment variables in the `docker-compose` files accordingly.

Apart from that, Kubernetes and Openshift are supported. The OAI 5GCN repository provides helm charts, which are used to launch the components in a Kubernetes/Openshift environment.

The NFs can also be deployed in a bare-metal scenario. This is useful when there are performance concerns, especially on the UPF.

5.2 PCF Policy Decisions

As described in Section 4.4, the PCF plays a crucial role in enabling edge computing in the 5GC. As this component did not exist in the OAI, it has been created as part of this thesis.

5.2.1 Policy Provisioning

The standards on the PCF and policy management (3GPP TS 23.503[93], 3GPP TS 29.512[36]), 3GPP TS 29.513[98], 3GPP TS 29.514[37]) do not specify how the policy decisions are provisioned in the PCF. In an operator's deployment, the PCF policies

are most likely provisioned using a custom API, which is provided to the customer and subscriber management services. There is the standardized PCF Policy Authorization API[37], but it does not allow to provision all the contents which may be contained in a policy decision.

Therefore, the author decided to use file-based provisioning for the first version of the PCF. The format of the files is `YAML`, but the structure and the names of fields is the same as in the SM Policy Control API[36].

There are three different parts of the policy to enable edge computing:

1. Traffic Control Descriptions
2. PCC Rules
3. Policy Decisions

The implementation allows to create different files for each type. Therefore, three directories need to be created, one for each type of rule. These directories are provided in the configuration file of the PCF, which contains other configuration such as the FQDN, port and protocol type of the SBI interface.

(1) follows the syntax of the `TrafficControlData` data structure and (2) follows the syntax of the `PccRule` data structure from the SM Policy Control API[36]. An example for the configuration can be found in Section B.1.

(3) uses a custom format, as the standard models do not consider this use case. The syntax is described in Listing 5.1.

```
<rule-name>:
  supi_imsi: <imsi>
  dnn: <dnn>
  slice:
    sst: <sst>
    sd:  <sd>
  pcc_rules:
  - <rule1>
  - <rule2>
  - ...
```

Listing 5.1: PCF policy decisions syntax

Items described with `<name>` are variable fields.

The order of importance of the fields is: `supi_imsi`, `dnn`, `slice`. In case neither of the SUPI, DNN or slice fields are there, the rule is considered a default rule, i.e., it matches always when no other rule matched. The `pcc_rules` fields is mandatory and must contain at least one item.

It is important to mention that the items of the `pcc_rules` list need to be the same as the ID of the PCC rules created in the PCC rules directory. The same applies for the traffic descriptions. Each PCC rule can have a reference to a traffic description ID. This ID must map to the one provisioned in the traffic descriptions.

The rules are loaded upon startup of the PCF. In case of an erroneous configuration, the user is informed. The PCC rules and the policy decisions are mandatory, therefore, if not at least one correct policy decision is available, the PCF exits. The traffic descriptions are optional, but in case there is an error, the user receives a warning on the terminal.

For handling the filesystem and to verify if the provided directories exist and are not empty, the `boost` filesystem library is used. Reading and parsing the `YAML` files is done using the `yaml-cpp`[W24] library.

5.2.2 SM Policy Control API

The current implementation of the PCF provides the SM Policy Control API that is used by the SMF to receive information about the PDU session.

The REST API has been auto-generated using the `openapi-generator-cli` tool together with the `TS29512_Npcf_SMPolicyControl.yaml` file, which is provided by the 3GPP TS 29.512[36]. The targeted framework and language for the generator is `cpp-pistache-server`. The generator creates the HTTP routes together with all the necessary model classes, which serve as Data Transfer Object (DTO) in the code.

From the procedures described in Section 4.4.2, the following have been implemented:

1. Creation of an SM Policy Association
2. Deletion of an SM Policy Association
3. Retrieving an SM Policy Association
4. SMF Update of an SM Policy Association

Thus, all the procedures where the PCF acts as a server and the SMF act as a client are implemented in the PCF. The notification mechanism, where the PCF acts as a client towards the SMF, are not implemented in the current release.

The SM Policy Control API uses the policy decisions, traffic descriptions and PCC rules, which have been provisioned via file.

5.2.3 Code Structure

Figure 5.3 shows the class diagram of the relevant parts of the PCF implementation.

The `sm_policy_decision`, `context`, `delete_data` and `update_data` are auto-generated DTOs from the API. The `&` indicates that a parameter is an output parameter and only a reference is passed.

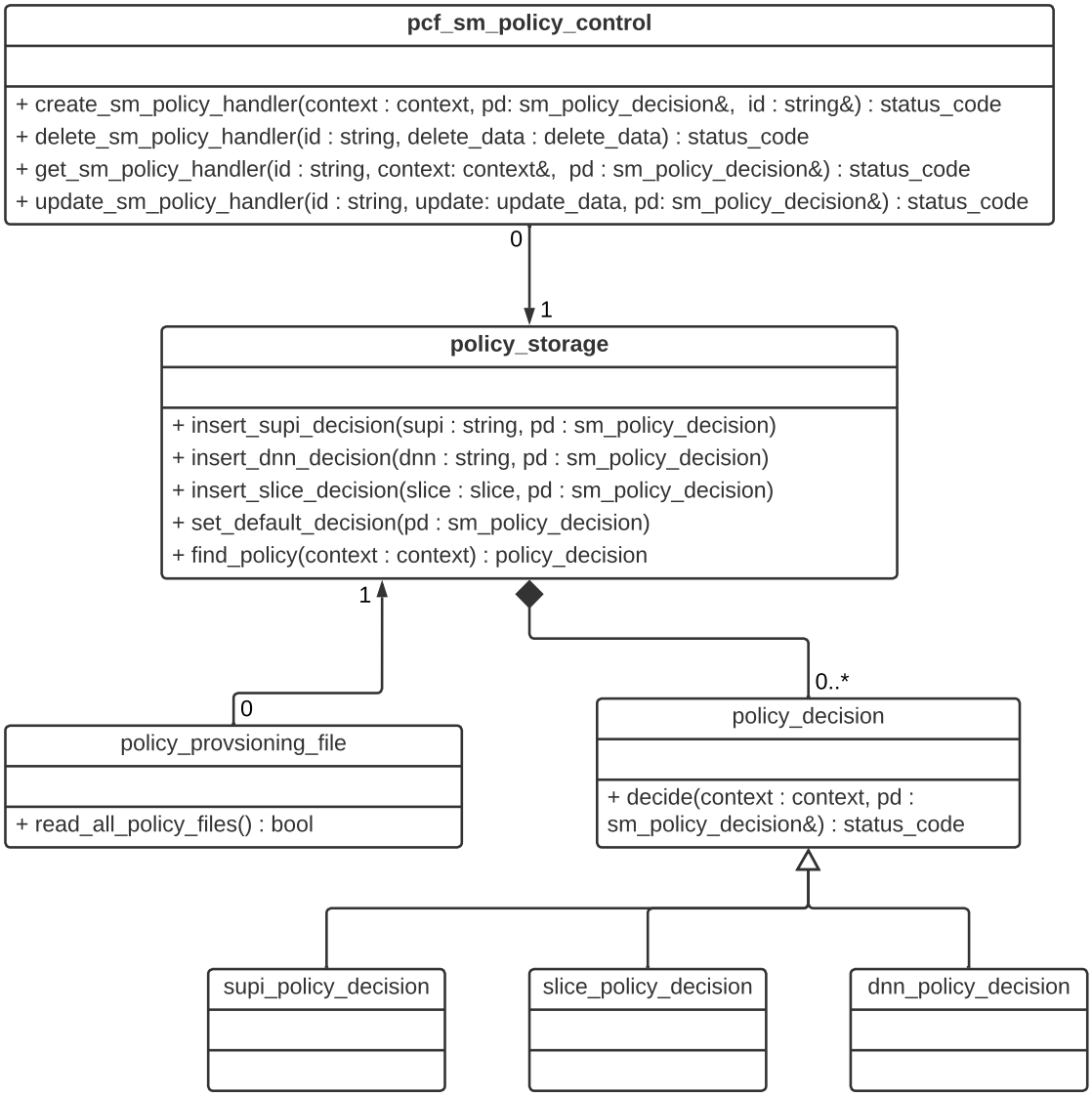


Figure 5.3: PCF class diagram

All the policy decisions are stored in a singleton `policy_storage` object. Upon startup of the application, the `policy_provisioning_file` class takes care of reading all the policies from the file system and inserting `policy_decisions` of different type into the `policy_storage`. In a future PCF version, other policy sources need to be considered as well, i.e., from the PCF policy authorization API. This design considers that and allows for extensions.

All the sub-types of `policy_decision` override the `decide` method, allowing to have different decisions based on the class type.

The `pcf_sm_policy_control` is the service class which handles all the requests from

the auto-generated REST endpoints. As it can be seen in the class diagram, these are the Create, Read, Update, Delete (CRUD) requests. When a policy association is created, the `find_policy` method of `policy_storage` is called. The calling function does not know the exact type of the decision, e.g., it could be a SUPI or a slice-based decision. The service class calls the `decide` method of the sub-class, although it only has a reference to the superclass. The decide method is thus called through dynamic binding.

5.3 UPF Selection and Traffic Steering on SMF

The PCF provides the policy rules for the SMF, as described in Section 4.4. However, the standardized API does not allow to define an exact UPF configuration inside the PCC rules. In fact, the only information available is the routing IDs and the DNAIs. For the solution within the OAI, the author of this thesis has implemented the "Dynamic UPF selection based on NRF information" option, as described in Section 4.5.3.

5.3.1 UPF Data Structure on SMF

Before exploring how the DNAIs can be mapped to a specific UPF configuration, there is the need to consider different UPF deployment scenarios. These scenarios are – with some exceptions – not defined by the standard, but are up to operator policy. An operator could, e.g., decide to have a dedicated UPF in each department of a state to optimize the internet routing. This of course depends on many factors such as the internet backbone of the operator. Thus, the solution shall be as dynamic as possible. Figure 5.4 depicts an example deployment scenario.

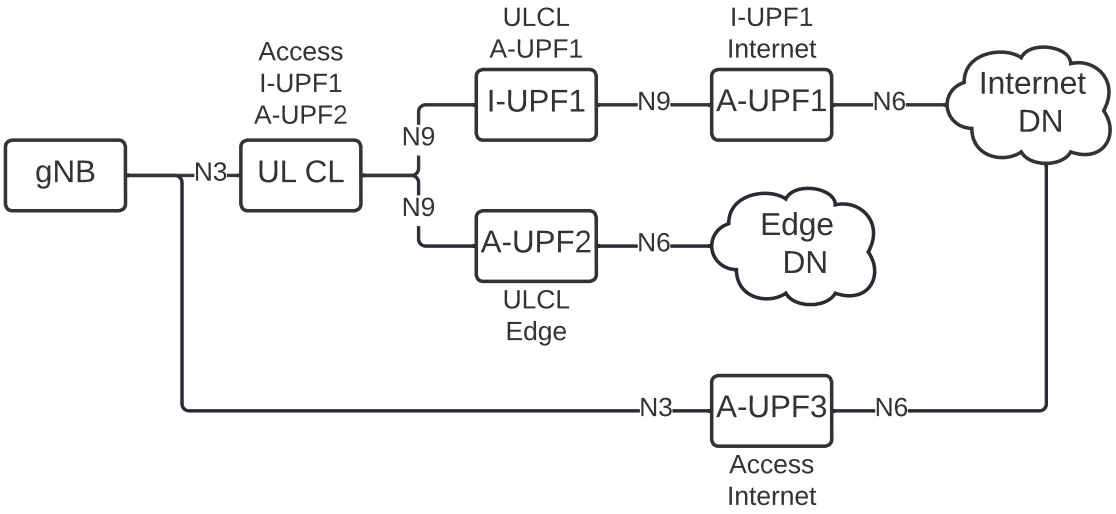


Figure 5.4: Example UPF deployment scenario

This scenario consists of 5 UPFs. An Intermediate UPF (I-UPF) is classified as not having an `N6` connection, thus not having a direct access to any DN. The Anchor UPF

(A-UPF) is used to anchor the PDU session and, therefore, acts as PSA.

A-UPF3 is acting as a central UPF and has an N3 interface towards the access and one N6 interface towards the internet DN. The other UPF with an N3 interface is acting as a UL CL and has N9 interfaces towards the I-UPF1 and the A-UPF2 respectively. The A-UPF2 is connected to the local edge DN, whereas the I-UPF1 is connected to the internet DN via the A-UPF1. Each UPF is configured with the DNAIs it serves. For example, UL CL serves the DNAIS "Access", "I-UPF1" and "A-UPF2".

This example indicates that the different UPF deployment scenarios form a graph. The previous implementation of the SMF stored all the available UPFs in a list and, therefore, there is no indication of the hierarchy between the different UPFs. Thus, only one UPF can be selected for any PDU session, e.g., based on DNN or slicing information. Each UPF in SMF has a PFCP association, which is created once between SMF and UPF when either becomes available. The PFCP heartbeat procedure ensures that the link between the components is working.

To allow selecting multiple UPFs for one PDU session, the PFCP associations need to be stored as a graph. Thus, the list has been replaced with an adjacency list. A new class `upf_graph` is responsible for adding, deleting, getting and updating UPF associations in the graph. The adjacency list is implemented as a hashmap of lists, where the unique UPF node ID is the key.

Given the graph data structure, the UPF selection can be split into three phases:

1. UPF graph creation
2. UPF subgraph selection
3. PFCP session handling

5.3.2 UPF Graph Creation

There are multiple ways how the SMF may be configured which UPFs are available. The straightforward approach is to configure one or more UPF IP addresses within the SMF configuration. In this case, the UPF does not have an available UPF profile or the UPF profile is very limited. In this case, the SMF is not able to read the implicit graph configuration from the UPF profile and, therefore, each UPF is added as a vertex without edges in the UPF graph.

The other possibility – which is also used for the prototype – is to utilize the NRF. Upon startup, the SMF indicates that it would like to receive notifications for the NF type UPF. This is done using the NRF NFManagement API, as described in 3GPP TS 29.510[34]. Whenever a UPF comes available, the NRF notifies the SMF and includes the UPF profile.

The UPF profile contains an FQDN and a list of interfaces. Whenever a UPF association is added to the graph G, the SMF verifies if one of the FQDNs in the interfaces of the

to-be-added UPF is equal to the FQDN of one of the existing UPFs. If this is the case, an edge between these two is added. If not, the UPF is added without edges. This is described in Algorithm 5.1

Algorithm 5.1: Algorithm to add UPF in UPF graph

```
1 InsertIntoGraph (NewUPF)
2   foreach UPF ∈ G do
3     foreach interface ∈ NewUPF do
4       if interface.fqdn = UPF.fqdn then
5         AddEdge(NewUPF, UPF)
6       else
7         AddNode(NewUPF)
8     end
9   end
```

Without considering the `AddEdge` and `AddNode` functions, the time complexity of this algorithm is $\mathcal{O}(|V|i)$, where V is the number of nodes in G and i is the number of interfaces in the new UPF. The `AddNode` function is $\mathcal{O}(1)$, as the graph is represented as `std::unordered_map`, which has a constant search and insertion in the average case. To prevent double edge insertions, the `AddEdge` iterates through the list of edges. Therefore, its time complexity is $\mathcal{O}(i)$, when we assume that the average number of edges is equally distributed for all UPFs.

Therefore, the total time complexity for insertion is $\mathcal{O}(Vi^2)$.

5.3.3 UPF Subgraph Selection

As described in Section 4.5.3, each UPF needs to declare which DNAIs it serves. Depending on the subscribed scenario, the PCF includes all the DNAIs that need to be served for a specific PDU session. If, e.g., the top path from Figure 5.4 should be selected, the policy decision contains the DNAIs "Access", "UL CL", "I-UPF1", "A-UPF1" and "Internet". When a UL CL should be inserted, the PCF needs to send one PCC rule per path, i.e., two when the internet DN and the edge DN need to be covered. The reason is that the PCC rule contains the traffic description, which identifies the user plane traffic. This is necessary for the UL CL to decide which path should be selected for a specific user plane traffic.

Based on this information, the SMF is able to select the UPFs that serve these DNAIs from the graph created in the previous step. The graph is traversed in a Depth-First-Search (DFS). The starting nodes are all the nodes which have an `N3` interface, also called access nodes. Algorithm 5.2 shows a modified version of the DFS, which also takes disconnected graphs into account.

Algorithm 5.2: Algorithm to select subgraph based on PCC rules

```
1  SelectUPFNodes (PCCRules)
2      subGraph ← empty graph
3      visited ← hashmap, where ∀V ∈ G : visited[V.id] = false
4      foreach rule ∈ PCCRules do
5          ruleDnais ← all DNAIs ∈ rule
6          foreach UPF ∈ G do
7              if IsAccess(UPF) ∧ ¬visited[UPF] then
8                  SubGraphDFS(UPF, subGraph, visited, ruleDnais)
9              end
10             if ¬VerifyGraph(subGraph, ruleDnais) then
11                 reset subGraph and continue with other access node
12             else
13                 subGraph is correct. Break and continue with next rule
14         end
15     end
16     allDnais ← n|∀n ∈ DNAIs from PCCRules
17     if ¬VerifyGraph(subGraph, allDnais) then
18         return Error
19     end
20 SubGraphDFS (start, subGraph, visited, dnais)
21     stack ← empty stack
22     stack.push(start)
23     while stack not empty do
24         current ← stack.pop()
25         visited[current] = true
26         if current serves any dnai ∈ dnais then
27             subGraph.AddNode(current)
28             foreach edge ∈ current.edges do
29                 if ¬visited[edge] then
30                     subGraph.AddEdge(current, edge)
31                     stack.push(edge)
32                 end
33             end
34         end
35     end
```

The algorithm starts with the `SelectUPFNodes` function, which takes the PCC rules from the PCF as input. The `SubGraphDFS` function is implementing the DFS. This function takes an existing graph and adds nodes and edges, therefore, it is possible to update the graph in each iteration. The `VerifyGraph` function ensures that all DNAIs from the PCC rule are covered and that the graph consists of exactly one access node (`N3` interface) and at least one exit node (`N6` interface). In case the verification fails, the graph is set to the state before the last execution of `SubGraphDFS`. The `visited` hashmap is not reset, ensuring that each node is only traversed once.

The `SubGraphDFS` function is a slightly adapted version of a DFS (see also Cormen et al.[99]). The difference to the generalized DFS is that only nodes which serve any of the given DNAIs are explored. If that is the case and an edge of the current node is not yet visited, it is added to the stack and also added as an edge in the subgraph.

The DFS is executed for each PCC rule. The reason is that the DNAIs in different PCC rules create different paths. To ensure that all paths are covered for each of the PCC rules, the DFS is used to create several paths, which are merged into the same subgraph. This requires that at the end of the algorithm, the subgraph is verified again, but with all the DNAIs from all PCC rules. This ensures that there is still only one access node, i.e., there is an UL CL somewhere in the path or there is only one path.

The time complexity for a DFS is $\mathcal{O}(|V| + |E|)$[99], where V is the vertices (nodes) and E is the edges of the graph. The `VerifyGraph` function iterates through all of the DNAIs for each of the nodes in the graph to verify if a node serves the DNAI. Therefore, its time complexity is $\mathcal{O}(d|V_s|)$, where d is the number of DNAIs and V_s is the number of nodes in the subgraph. Each DNAI refers to a UPF, the access DNAI and one or more exit DNAIs. There are as many exit DNAIs as PCC rules, denoted as r. Thus, $|V_s| = d - 1 - r$ and, therefore, the number of steps is $d^2 - d - rd$. As rd is significantly smaller as d^2, the time complexity is $\mathcal{O}(d^2)$.

The `VerifyGraph` function is executed as many times as the DFS algorithm. Thus, the total time complexity for the algorithm is $\mathcal{O}(r(d^2 + (|V| + |E|))$.

In a realistic scenario, r is assumed to be very small and it is unlikely that it is ever greater than 10 (which would mean 10 different paths for a PDU session). The same applies for d. The number of DNAIs/UPFs per PDU session is likely to be in the same magnitude. Therefore, even though the time complexity is quadratic in number of DNAIs, the quadratic part of the algorithm is always small in a realistic deployment. The number of UPFs in the total graph, however, can be very large and traversing through these is done in linear time.

5.3.4 PFCP Session Handling

3GPP TS 23.502[92] defines the flow of a PDU session establishment. From the perspective of the SMF, it can be divided into two parts: (1) UL procedure and (2) DL procedure. Figure 4.9 shows the UL procedure. The communication protocol between the SMF and

the UPF over the N4 interface is PFCP, which is defined in 3GPP TS 29.244[100]. In the UL procedure, the SMF instructs the UPF to create a GTP tunnel in UL direction, from the UE to the DN. One important aspect of the GTP protocol is that each tunnel has a unique ID, the Fully Qualified Tunnel Endpoint Identifier (F-TEID). 3GPP TS 29.244 Chapter 5.5[100] specifies that this ID must be created by the user plane components. Hence, the SMF is not allowed to create F-TEIDs. This restriction requires that the SMF creates the session for each of the UPFs in a specific order. For session establishment in UL, the SMF needs to traverse the graph from the exit nodes.

In DL, the procedure is inverted, i.e., the SMF starts at the access nodes. It sends session modification requests to the UPFs to create the rules for the DL direction.

Note: The abbreviations in Figures 5.5, 5.6, 5.7, 5.8 and 5.9 are: Network Instance (NWI), Packet Detection Rule (PDR) and Forwarding Action Rule (FAR)

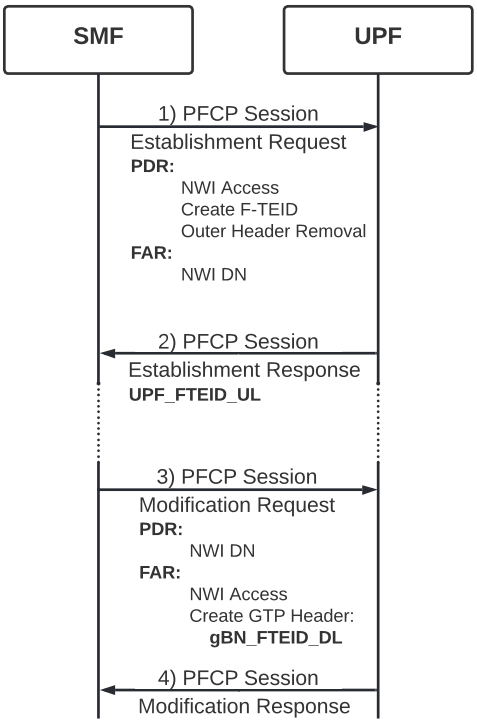


Figure 5.5: PFCP messages for a central UPF scenario

Figure 5.5 describes the call flow for a central UPF scenario or an edge UPF scenario using the distributed anchor point access model in UL and DL direction. This is the most common and simple case, when one UPF is used to route the user plane. The NWI uniquely identifies a network interface and is used by the UPF to install the PDRs and FARs correctly. In message (1), the PFCP session is established with a PDR matching the NWI `Access`. Although this NWI can be chosen freely on the UPF, it is a good practice to have a naming scheme.

In these examples the following naming scheme has been used:

- "Access" : N3 interface towards gNB
- "DN": N6 interface towards a DN
- "UPF": N9 interface to the next UPF, coincides with UPF name

It is worth to note that according to 3GPP TS 29.244[100], the NWI is an optional field. In this case, the UPF uses other information from the PFCP protocol to identify the interface, i.e., there is a flag indicating UL or DL direction. This is sufficient for the scenario depicted in Figure 5.5, but as soon as more than one UPF is part of the PDU session, the NWI is necessary to correctly identify the interfaces. This is especially true for the UL CL scenario, as the UL CL has two N9 interfaces and without the NWI it is impossible to properly select the correct one.

The PDR of message (1) instructs the UPF to remove the outer header, as the GTP tunnel terminates here and forwarded packets must not contain a GTP header. Also, the UPF is instructed to create an F-TEID in UL direction. The response (2) contains the created F-TEID UPF_FTEID_UL.

In the PDU session establishment flow, the UPF_FTEID_UL is communicated to the gNB via the AMF. Then, the SMF receives a PDU session SM context modification request which contains the F-TEID of the gNB, called gNB_FTEID_DL in the call flows. This procedure is not shown in Figure 5.5, but it is happening between messages (2) and (3). In message (3), the SMF installs the rules in the DL direction. The NWI in the PDR is DN, indicating that traffic is coming from the DN (e.g., the internet). There is no need to create an F-TEID or remove an outer header, but it is required to add a GTP header using the gNB's F-TEID. This is the content of the FAR. Message (4) is the acknowledgment that the rules of message (3) have been successfully installed.

Figure 5.6 shows the call flow using one I-UPF and one A-UPF in UL. This example indicates that the order of the PFCP messages matter. Message (1) is used to create the rules in UL for the A-UPF, meaning the packets should be forwarded to the DN. However, the NWI is the I-UPF in the PDR. Also, an F-TEID is created and the outer header is removed. The A-UPF replies with the created F-TEID A-UPF_FTEID_UL in message (2). In message (3) the I-UPF is instructed in the PDR to use the NWI Access (the gNB) and also create an F-TEID. It also has to remove the outer GTP header. This is because there is a distinct GTP tunnel between gBN and I-UPF and between I-UPF and A-UPF. The FAR in message (3) configures the UPF to create a GTP header using the previously created A-UPF_FTEID_UL. Therefore, at this point the UPF removes the GTP header from the gNB and adds a new GTP header with the F-TEID of the A-UPF. Message (4) contains the newly created F-TEID I-UPF_FTEID_UL. One can see that the F-TEID created by the A-UPF is required in message (3). Thus, it is crucial that the messages are sent in this order.

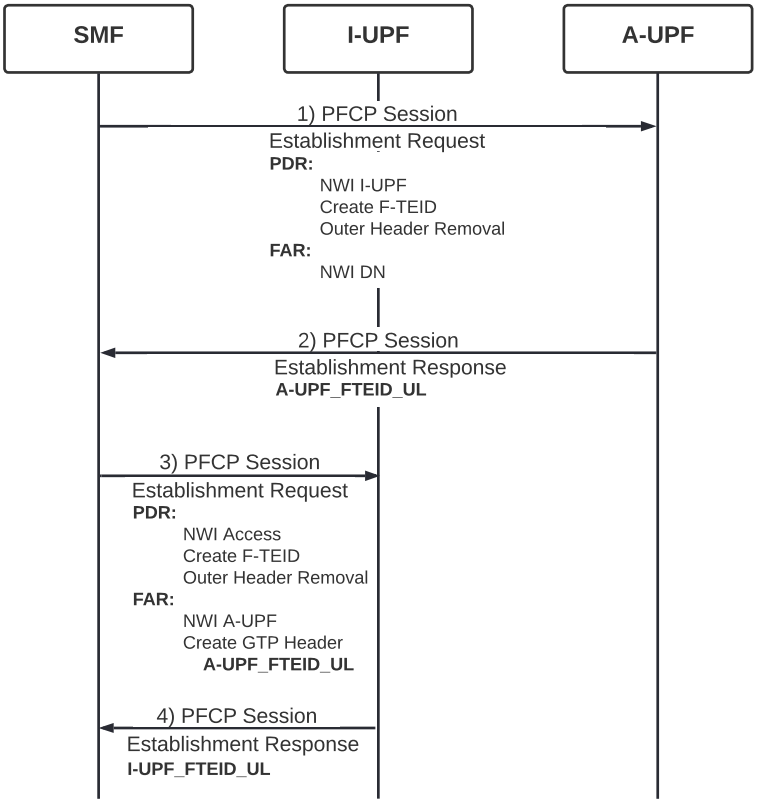


Figure 5.6: PFCP messages for an I-UPF/A-UPF `N9` scenario in UL direction

Algorithm 5.2 selects the subgraph for this specific PDU session. In the case depicted in Figure 5.6, two UPFs have been selected. The underlying graph structure implicitly defines if a UPF acts as I-UPF or A-UPF, as the presence of an `N6` interface indicates an A-UPF and the presence of an `N3` interface indicates an I-UPF. In case both are present, it is a central UPF scenario as depicted in Figure 5.5. Therefore, before sending the first message, it is already known how the scenario should look like. This is the main reason why the PFCP session handling and the UPF subgraph selection are split into two different stages. The scenario in Figure 5.6 shows that the algorithm has to start at the A-UPF. To accomplish this, the subgraph is traversed with a DFS as well. The crucial difference is that for PFCP session handling the DFS is asynchronous. Upon receiving the PDU session establishment request from the AMF, the SMF starts the asynchronous DFS. Then the SMF performs a similar procedure as the `SubGraphDFS` function outlined in Algorithm 5.2. The major difference is that instead of going through all the nodes in the graph, only the current node is returned and the neighbors are only pushed to the stack, but not processed. This is achieved by declaring the stack as a member variable. Therefore, only one asynchronous DFS procedure can be active for each UPF graph. Upon receiving a PFCP Session Establishment/Modification Response, the SMF continues the DFS, starting with the first edge from the previous UPF. This is repeated until all

nodes in the graph are covered. Only then, the PDU Session Establishment/Modification Response is sent to the AMF.

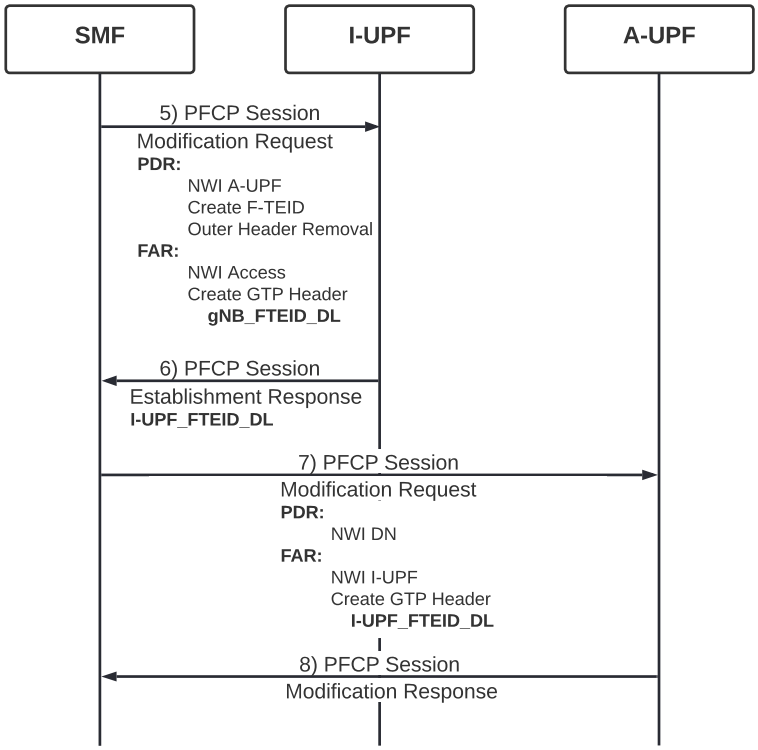


Figure 5.7: PFCP messages for an I-UPF/A-UPF N9 scenario in DL direction

Figure 5.7 is the scenario depicted in Figure 5.6 in DL direction. It can be seen that in DL, the algorithm starts at the I-UPF. Analyzing the content of the messages reveals the reason. In message (5), the I-UPF is instructed to create an F-TEID for the NWI A-UPF. Also, as this is the DL direction, the I-UPF is instructed to create a GTP header using the F-TEID of the gNB. Message (6) contains the created F-TEID I-UPF_FTEID_DL. In message (7), the GTP tunnel between I-UPF and A-UPF is set up. The A-UPF is instructed to match packets on the DN NWI and to forward them to the I-UPF. Here, it creates a new GTP header using the I-UPF_FTEID_DL created in message (7). Message (8) is the acknowledgment of message (7).

The reason the author has chosen to implement the different UPF scenarios as nodes in a graph is flexibility. The I-UPF/A-UPF scenario contains one I-UPF, but it is possible that several UPFs are linked together and that more than one I-UPF is part of the graph. However, for each branch only one A-UPF is present. The implemented DFS allows to cover all these scenarios without any configuration or code change. The representation of the UPFs is implicit in the graph and, therefore, most scenarios are covered. There are some exceptions though, especially when it comes to branching. In the current version, starting from the access nodes, branching out is supported, as shown in Figures 5.8 and

5.9, but it is not supported to merge branches. In short, the graph must be acyclic. However, there are valid scenarios where the UPF graph is cyclic. Thus, the author chose a general graph structure and not a tree. Enhancing the SMF to support this scenarios is possible without having to change the underlying data structure.

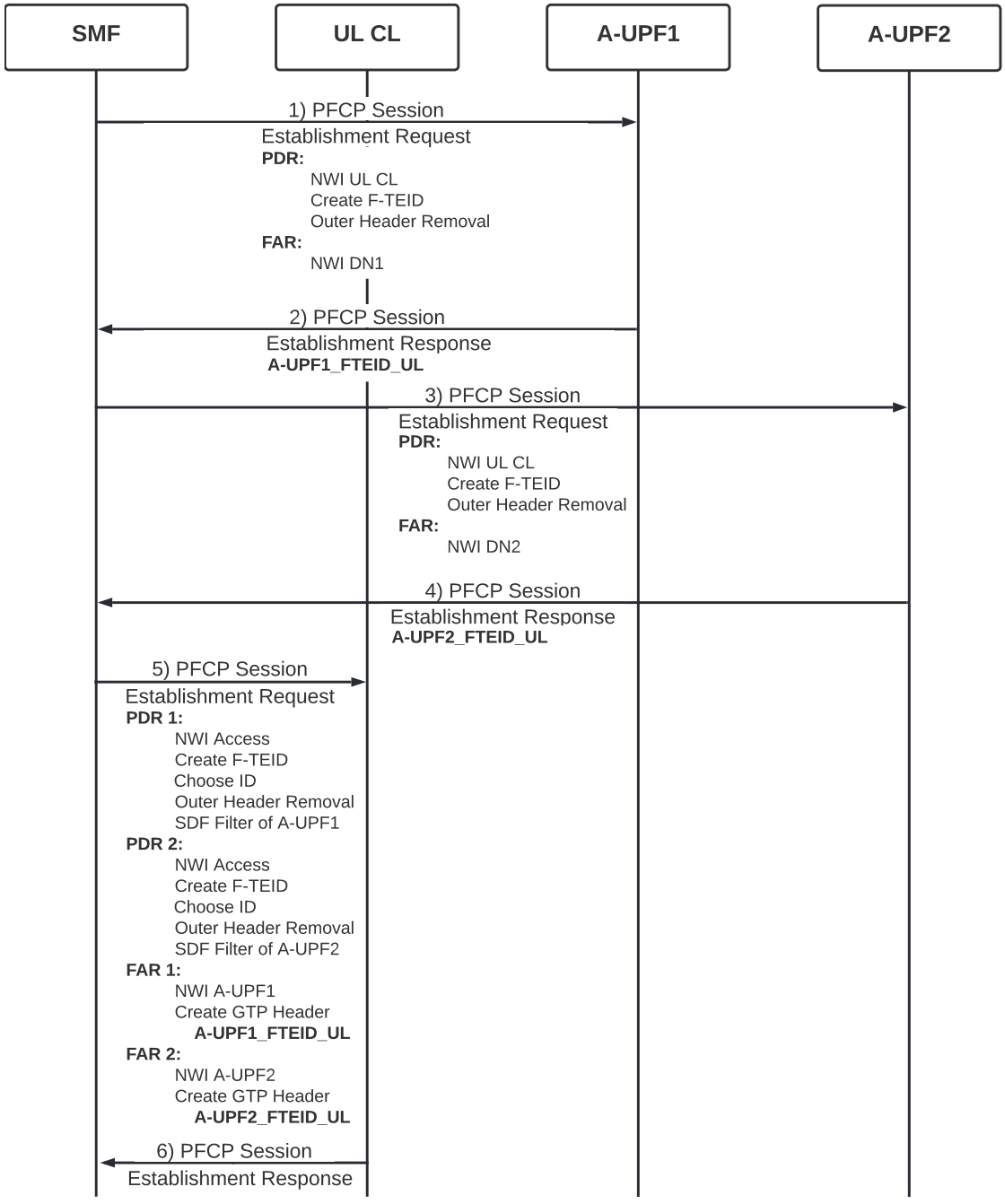


Figure 5.8: PFCP messages for a UL CL scenario with two N9 A-UPFs in UL direction

Figure 5.8 shows the call flow for a UL CL scenario in UL direction. As with the previous examples, the algorithm starts at the A-UPF. As there are two A-UPFs, one of them is chosen, whereas the order does not matter. Messages (1) and (2) are analogous to the corresponding messages in Figure 5.6. At this point, the DFS continues with the unvisited neighbors of A-UPF1, which is the UL CL. However, the point of the UL CL is to have a distinct GTP tunnel towards A-UPF1 and one towards A-UPF2. The SMF already has the F-TEID for the A-UPF1, but not for the A-UPF2. There are two possible solutions to this problem:

- Continue with the UL CL and make a tunnel for the A-UPF1
- Skip the UL CL for now and continue with the A-UPF2

The author has implemented both options in the OAI SMF, but finally the latter became part of the official code base. The former requires an additional PFCP message. As – at this point – the UL CL has a tunnel only to the A-UPF1, at a later stage, there is the need for a PFCP Session Modification Request to add the tunnel to the A-UPF2. This happens after the A-UPF2 has been instructed to create its F-TEID.

Due to the increased complexity and the additional message in UL, the author chose the option as depicted in Figure 5.8. After message (2), the algorithm identifies that the UL CL has unvisited `N9` neighbors (in this case the A-UPF2). The UL CL is removed from the stack and the DFS algorithm starts again from the A-UPF2. It is safe to remove the UL CL at this point from the stack, as it is also a neighbor of A-UPF2, hence it is added to the stack again. Message (3) and (4) have the same purpose and nearly the same content as (1) and (2) with the exception that different F-TEIDs are created. Naturally, the SMF has to store these accordingly in the graph data structure.

Message (5) is interesting, as it differs from the Session Establishment Requests of the previous examples. It can be seen that it contains two PDRs and two FARs. The NWI for both PDRs is `Access`, as this is the UL direction. Both contain the instruction to create an F-TEID. However, this UPF is an UL CL, meaning it has one `N3` interface to the gNB and two `N9` interfaces to A-UPF1 and A-UPF2, respectively, as depicted in Figure 5.4. Therefore, it has to use the same F-TEID towards the gNB. This is indicated using the `Choose ID` option. The PFCP protocol defines that when this flag is set, the UPF creates the same F-TEID for both PDRs. Another important difference is the SDF Filter. SDF stands Service Data Flow and must not be confused with the Simulation Description Format (SDF) described in Chapter 6. Section 4.4 discusses the PCF PCC rules and how traffic can be identified. The SDF filter is a representation thereof. It is a string describing the IP address and network mask. Listing B.3 shows an example of an SDF filter in the PCC rules. The exact same rule from the PCC rules is applied here on the PFCP layer. Therefore, the information how to identify traffic, originated from the PCF, is enforced on the UL CL. The FARs in message (5) instruct the UPF to forward the traffic with a new GTP header, using A-UPF1's or A-UPF2's F-TEID created in messages (2) and (4).

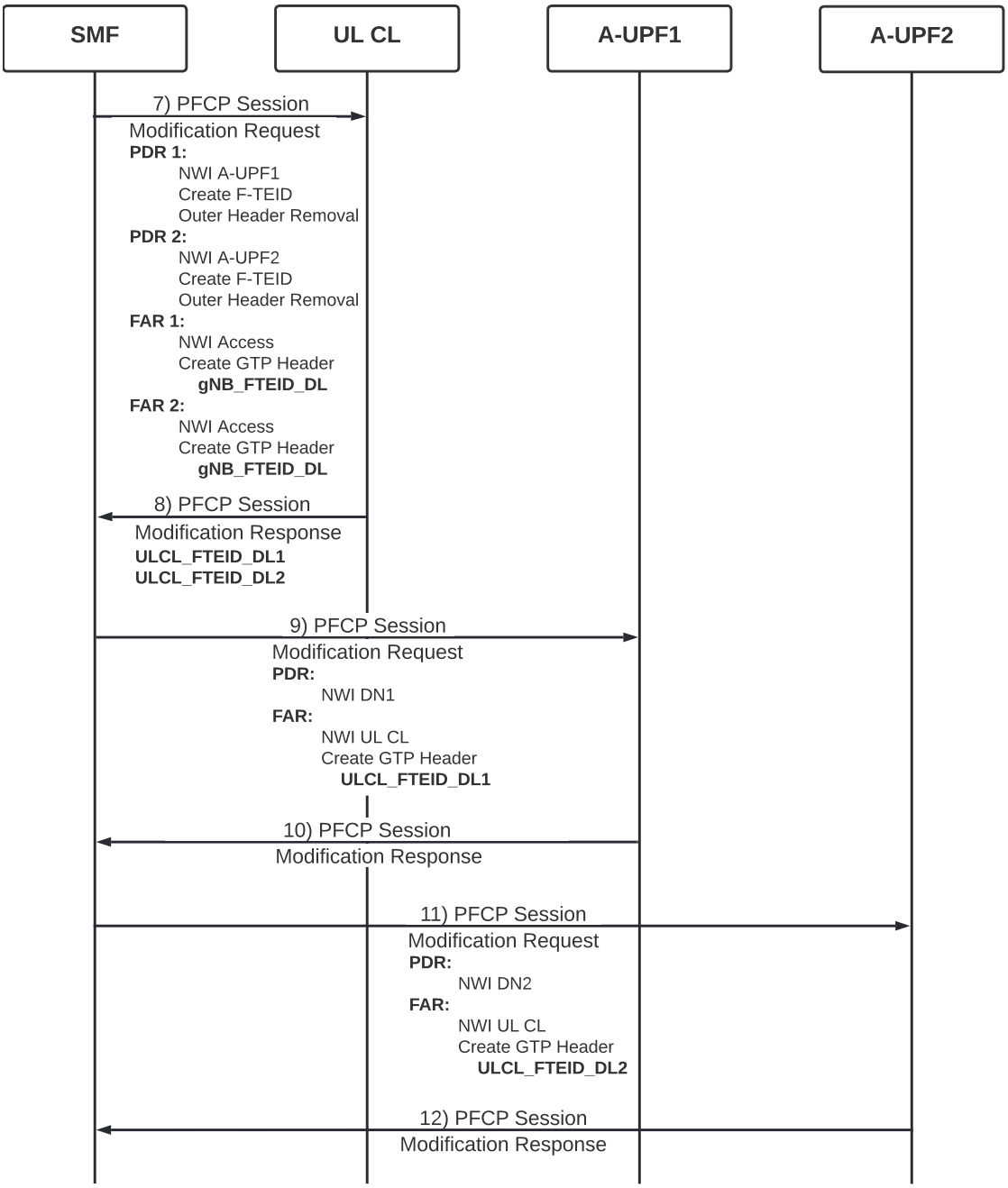


Figure 5.9: PFCP messages for a UL CL scenario with two N9 A-UPFs in DL direction

Figure 5.9 shows the call flow of the PFCP messages for the UL CL scenario in DL direction. Message (7) is similar to message (5) from Figure 5.7. The difference is that there are two N9 interfaces in this scenario, thus two PDRs and two FARs are required. The PDRs are again used to create an F-TEID for the A-UPFs. However, in the FAR the

same F-TEID is used, namely the `gNB_FTEID_DL`, as the traffic from both the A-UPF1 as well as the A-UPF2 shall be forwarded to the same gNB. Messages (9) to (12) are analogous to messages (5) and (6) from Figure 5.7.

Note for the inclined reader: The 3GPP specifications on the SDF format are not easy to locate. 3GPP TS 29.244[100] references the PCC rules specification 3GPP TS 29.212[101], which describes the "Flow-Description AVP" in the Gx reference point, an interface from the 4G EPC. However, the format's specification is found in IETF RFC 6733[102], which describes the Diameter Base Protocol. Diameter has been used in 4G for parts of the control plane, which is signaled with REST in 5G. Still, some message definitions remain the same, such as the SDF format

The time complexity of this procedure is $\mathcal{O}(|V| + |E|)$, not considering the networking delay and the procedures at the UPF.

5.4 Traffic Filtering and Forwarding in the UPF

There are two different UPFs available in the OAI: The first one is the SPGWU, which has been upgraded from a 4G SGW/PGW to support 5G procedures. The second is the VPP-UPF, which is based on the VPP-UPG from Travelping[W25]. Vector Packet Processing (VPP) is a Linux Foundation based project that aims to provide fast and reliable routing and switching capabilities, as described by Cerović et al.[103]. The VPP platform is dynamic and can be extended using plugins[103]. In fact, the VPP-UPG consists of plugins for the VPP platform.

For the described solution, the VPP-UPF is used, as it supports more features than the OAI SPGWU. The current version V1.4 of the VPP-UPF supports all the scenarios and PFCP procedures described in this chapter. However, as discussed in Chapter 7.4, a performance issue on the VPP-UPF prevented the use of the VPP-UPF for the robotics use case. Therefore, the experiment is conducted using the SPGWU. As this UPF does not support all PFCP procedures, the option described in Figure 5.5 is used, forcing the deployment to follow the distributed anchor point access mode.

The configuration for the VPP-UPF needs to be adapted, especially the UPF profile and – depending on the deployment scenario – the UPF init configuration. Examples of the configuration can be found in Section A.3. As the configuration of the different UPF scenarios can become complex for the user, the author has provided a Python script that generates the configuration based on the environment variables set in the `docker-compose` files.

5.5 Code Contributions

The author's contributions described in this chapter can be found in the official OAI 5GC GitLab project's repository: https://gitlab.eurecom.fr/oai/cn5g.

All the contributions for the NFs SMF, PCF, VPP-UPF and NRF are merged into the `master` branch of the respective repositories and are part of the OAI 5GC v1.5 release. Table 5.1 shows how the individual merge requests from the author can be found on the OAI 5GC GitLab repositories. The "federated" repository `oai-cn5g-fed` is used to host documentation and `docker-compose` files, which are relevant for all NFs.

Repository	Merge Requests	URL
SMF	6	https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-smf/-/merge_requests?state=merged&author_username=spettel
PCF	9	https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-pcf/-/merge_requests?state=merged&author_username=spettel
NRF	1	https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-nrf/-/merge_requests?state=merged&author_username=spettel
VPP-UPF	1	https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-upf-vpp/-/merge_requests?state=merged&author_username=spettel
Federated	1	https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/merge_requests?state=merged&author_username=spettel

Table 5.1: OAI GitLab repositories with merge requests from the author

An overview of the author's activity in all the OAI 5GC repositories can be found at the following location:
https://gitlab.eurecom.fr/spettel.

CHAPTER 6

Use Case: Robotics

This thesis is motivated by the robotics use case. As described in Section 1.3, the hypothesis is that edge computing in 5G allows to offload latency-critical robotic applications. This chapter describes how the robotics use case can profit from edge computing in 5G and how the robot simulation is set up.

6.1 Scenario

The underlying scenario for the robotics use case is a factory with autonomous robots. As it is the case for nearly all robotics use cases, the robot has the task to create a map of its environment. Although other solutions exist, such as fixed paths in the factory hall, using SLAM for navigation allows the robot to locate itself and navigate freely in the world.

SLAM is chosen as an algorithm to exemplify the edge computing use case, as it is very essential on the one hand and also latency-constrained on the other hand. Showing that offloading SLAM algorithms in a 5G connected factory is possible can lead to other use cases such as pose estimation, path planning or robot steering.

For successfully offloading nodes of the robot, the robot has to have a network connection. As a fixed-net option is not feasible, two different scenarios are compared: WiFi-enabled scenario and 5G-enabled scenario. Other wireless IoT technologies such as Zigbee, Bluetooth or LoRa are not considered, as neither of them provide the necessary bandwidth for this scenario, as described by Sanchez-Gomez et al.[104].

6.1.1 WiFi-Enabled Scenario

In the WiFi-enabled scenario the factory is equipped with multiple WiFi Access Points (AP) in infrastructure mode. As an average factory is larger than the typical reach of one WiFi AP, they are configured to use WiFi roaming, essentially allowing the handover of a session over the serving area of one AP.

6.1.2 5G-Enabled Scenario

The 5G-enabled scenario replaces the WiFi APs with gNBs. Depending on the size of the factory, one cell may be enough. It is also possible that the factory is equipped with several 5G micro-cells.

The advantage of this scenario compared to WiFi is that the handover support is more robust. While cellular networks are built upon mobility, many WiFi devices do not support handover correctly, which may lead to interruptions. Also, the larger range of 5G networks ease the deployment.

In this use case it does not make a difference whether a private 5G network is used, where the factory operators provide their own 5G network. In that case most likely the RAN and the 5GC is deployed in the same physical network, together with the edge data center where the SLAM algorithm is executed. However, the solution described in Chapter 4 and the OAI implementation in Chapter 5 target edge computing enabled in a public 5G network offered by an MNO.

6.2 Simulation in Gazebo

ROS2 and the simulation tool Gazebo allow to simulate a robot with its actuators and sensors and its environment (world). Gazebo was initially developed in the year 2001, as described by Koenig and Howard[105], but has gained a lot of traction in the last twenty years and is the most important open source simulator in the ROS ecosystem, as they are tightly coupled.

The Gazebo version used for the simulation is `11.10.2`. For this use case, the classic Gazebo has been used. Since 2019, Gazebo is being modernized in a new project called Ignition Gazebo, as described in the Ignition documentation[W26] and the classic Gazebo documentation[W27].

The characteristics of the world and the robot, such as which sensor plugins to load and how the world is built are described with the SDF. As stated by the SDF documentation[W28], it is an XML format that allows to define the environment, the used models, the lighting and the size and shape of a robot as well as the actuators and sensors.

6.2.1 Environment and World

A realistic scenario for 5G edge computing and robotics are factories. Therefore, the simulated environment shows a factory setting. The world SDF files, meshes and models

are from the Automatic Addison blog[W29] and from the warehouse simulation GitHub repository[W30]. The GitHub repository also contains examples for navigation and SLAM, but these have not been used for this simulation. The reason is that these files are built upon ROS, the predecessor of ROS2 and are not compatible. It also contained moving obstacles, which have been reduced to increase the reproducibility of the performed experiments. Figure 6.1 shows the world in a perspective view and Figure 6.2 shows the outline of the world in an orthographic view from the top. The orthographic view can be used to assess the quality of the produced map of the SLAM algorithm. It is possible to compare the map of the simulation with the generated map. However, this requires that the scale of both images is the same.

The simulated world has the dimensions 20x20 m and is thus relatively small to depict a factory hall. Nevertheless, given that the robot is 10 cm in diameter and there are several obstacles present in the world, it is sufficient for the underlying experiment. Also, the corners and pillars can be used to show that the SLAM algorithm correctly maps these. A comparison with the produced maps in Figures 7.6, 7.7 and 7.8 in Section 7.4 shows that this is the case.

Figure 6.1: Factory world in perspective view

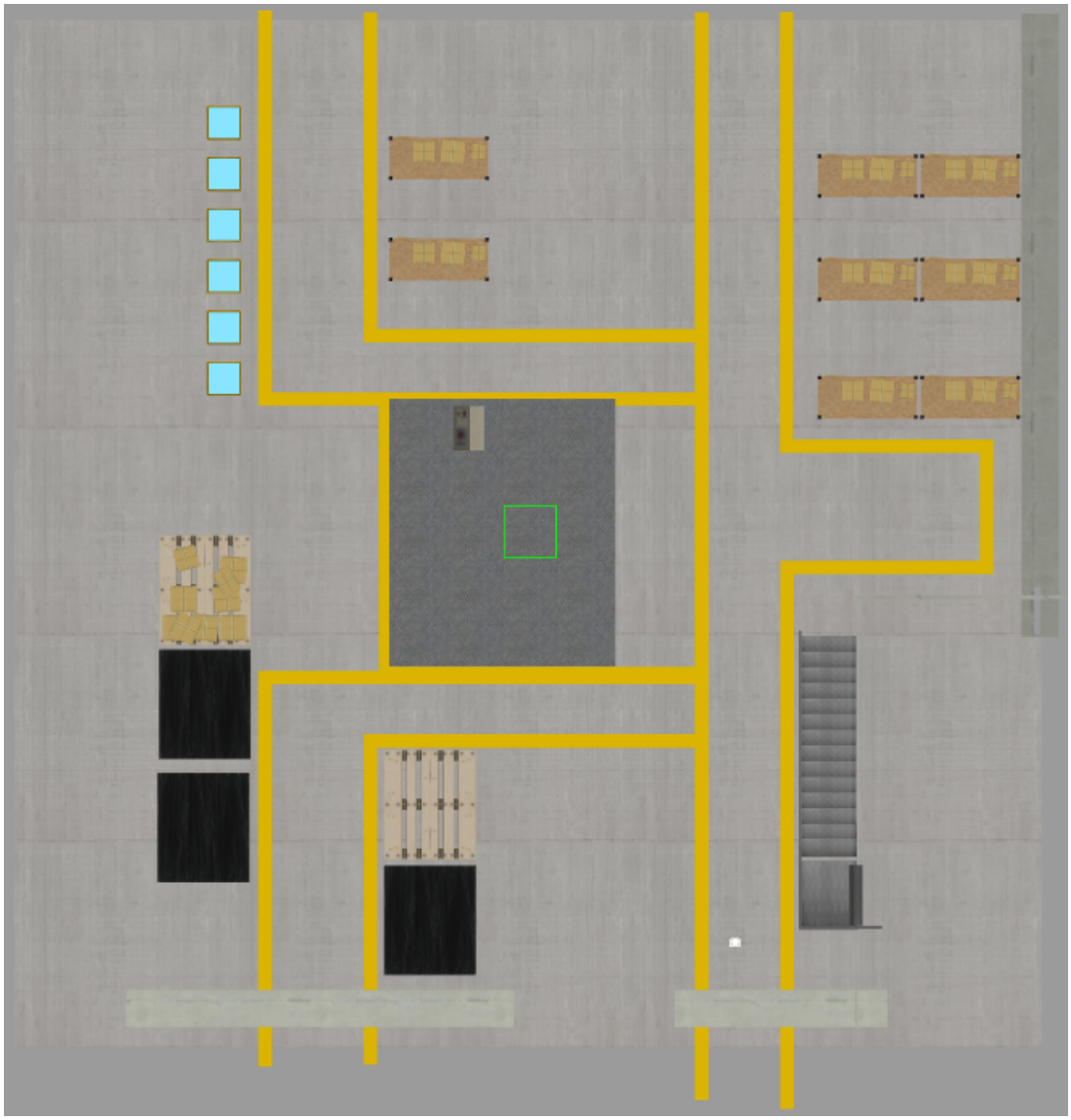


Figure 6.2: Factory world in top-down orthographic view

6.2.2 Robot Platform

The robot used in the simulation is a `turtlebot3`, which uses the `burger` robot model. The source code is available on the `turtlebot3` GitHub repository[W31], which also contains the SDF file for the robot.

The evaluation of the accuracy of the SLAM algorithm requires a ground truth pose, which is the accurate pose of the robot in the environment. To allow this, a ground truth plugin has been added to the robot's SDF, as described in Listing 6.1. The Gaussian noise is set to 0 to acquire an exact pose.

```
<plugin name="p3d_base_controller" filename="libgazebo_ros_p3d.so">
 <ros>
  <remapping>odom:=ground_truth/odometry</remapping>
  </ros>
<always_on>true</always_on>
<update_rate>50</update_rate>
<body_name>base_footprint</body_name>
<gaussian_noise>0.00</gaussian_noise>
<frame_name>world</frame_name>
<xyz_offset>0 0 0</xyz_offset>
<rpy_offset>0 0 0</rpy_offset>
</plugin>
```

Listing 6.1: Ground truth plugin for the `turtlebot3` robot

Actuators

The `burger` model of the `turtlebot3` has a cylindrical shape as base with two attached wheels. The motion of the wheels are simulated using the `turtlebot3_diff_drive` plugin. Thus, the robot has a differential drive. As described by Klančar et al.[106], this means that the robot is non-holonomic. A robot is holonomic when the degree of controllable and total degrees of freedom is equal. Due to its two wheels, the `burger` model is not able to rotate around the z-axis while driving in a specific direction. It also has a castor wheel to prevent tipping, although this is not visualized in the simulation.

Sensors

The most important sensor in this setup provided by the `turtlebot3` is the LIDAR sensor. It uses the `libgazebo_ros_ray_sensor` plugin and publishes ROS2 messages of type `sensor_msgs/LaserScan` on the `scan` topic. It uses rays of light to detect the distance to close objects in a 360 ° radius. The minimum range is 0.12 m and the maximum range is 3.5 m. Topics are published 5 times per second. There is a Gaussian noise on the values with a mean of 0 and a standard deviation of 0.01.

The second sensor of the `turtlebot3` is an IMU. As described by Norhafizan et al.[107], an IMU is used to measure velocity, orientation and gravitational force. This is used by the robot to estimate its current pose based on previous angular and linear velocities. This process tends to become more inaccurate over time, as the inaccuracy of each velocity adds up over time. It is published on the topic `imu` with type `sensor_msgs/msg/Imu` 200 times per second. The angular velocity measurements have a Gaussian noise with mean 0 and standard deviation $2 \cdot 10^{-4}$, whereas the standard deviation is $1.7 \cdot 10^{-2}$ for the linear velocity.

Topics and Robot Description

Based on the sensors and actuators, the `turtlebot3` is publishing several topics. Figure 6.3 shows all the ROS2 nodes and topics published when the simulation is running and

the `turtlebot3` has been spawned. An ellipsoid is a ROS2 node and a rectangle is a topic. The `turtlebot3_diff_drive` subscribes to the `cmd_vel` topic. This topic has the type `geometry_msgs/Twist` and contains a 3D vector for the linear and a 3D vector for the rotational speed. The robot cannot linearly move in the Z axis and it can only rotate around the Z axis. Figures 6.4a and 6.4b show the linear and rotational axes. The X axis is depicted in red, the Y axis in green and the Z axis in blue.

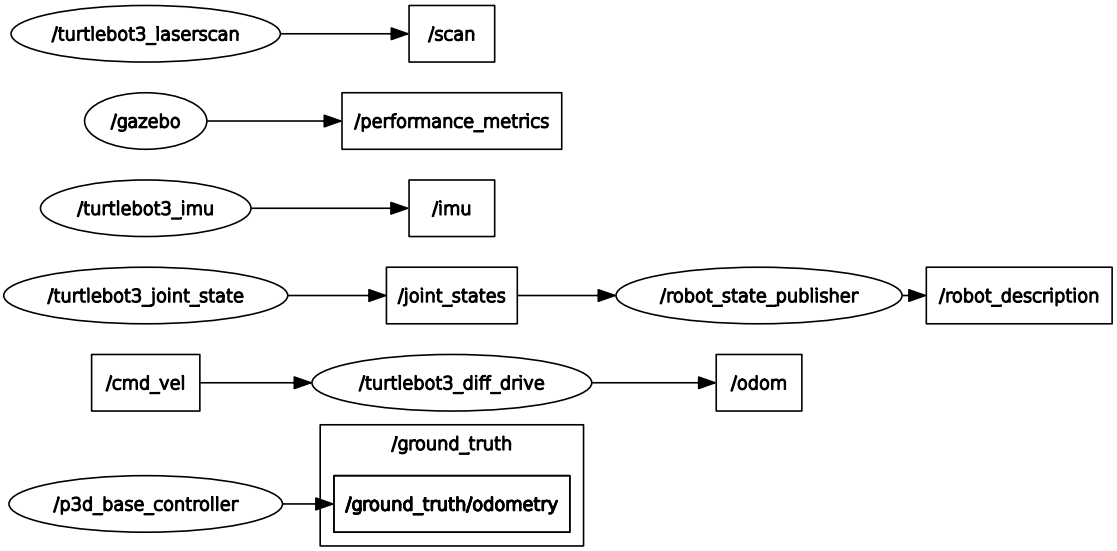
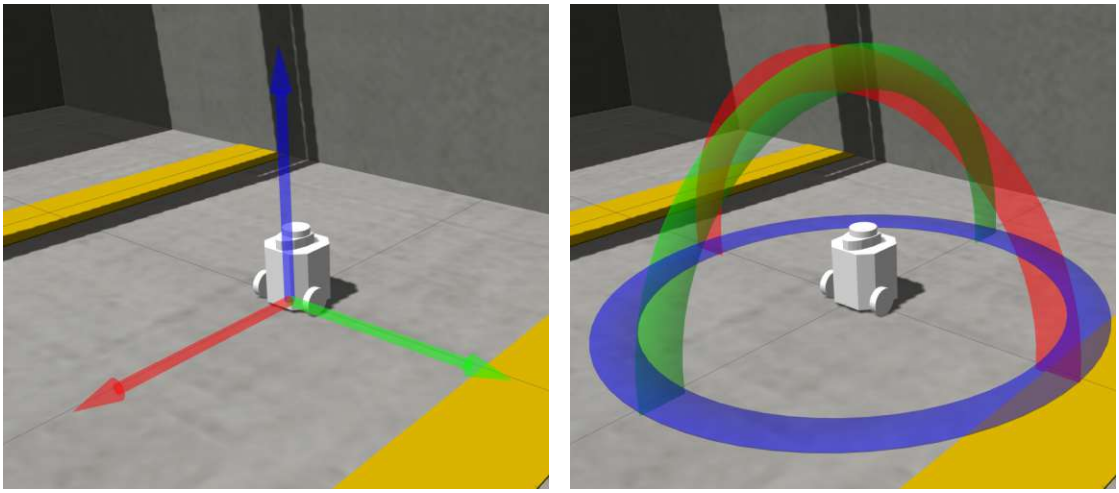


Figure 6.3: `turtlebot3` and Gazebo nodes and topics



(a) Linear axes (b) Rotational axes

Figure 6.4: Axes of the `turtlebot3` robot

6.3 SLAM Node

The SLAM algorithm used in the experiment is provided by the `cartographer-ros` node from Google and uses the `cartographer` software for localization and mapping. It supports 2D and 3D SLAM, although initially only 2D was supported, as described by Hess et al.[73]. The `cartographer-ros` package officially does not support ROS2, although binary packages are available for Ubuntu. However, the newest ROS2 distribution "Humble Hawksbill" – which supports Ubuntu 22.04 – does not contain the packages. Thus, the SLAM node is running in a Docker container which uses the ROS2 distribution "Foxy Fitzroy". This is the second latest LTS release and supports Ubuntu 20.04. Therefore, even in the local deployment option described in Chapter 7, the SLAM node is running in Docker. The `cartographer-ros` package is configured using a LUA script and the description of the parameters is in the official documentation[W15]. Listing 6.2 shows the cartographer configuration.

```
include "map_builder.lua"
include "trajectory_builder.lua"

options = {
  map_builder = MAP_BUILDER,
  trajectory_builder = TRAJECTORY_BUILDER,
  map_frame = "map",
  tracking_frame = "imu_link",
  published_frame = "base_footprint",
  odom_frame = "odom",
  provide_odom_frame = true,
  publish_frame_projected_to_2d = true,
  use_odometry = true,
  use_nav_sat = false,
  use_landmarks = false,
  num_laser_scans = 1,
  num_multi_echo_laser_scans = 0,
  num_subdivisions_per_laser_scan = 1,
  num_point_clouds = 0,
  lookup_transform_timeout_sec = 0.2,
  submap_publish_period_sec = 0.15,
  pose_publish_period_sec = 30e-3,
  trajectory_publish_period_sec = 30e-3,
  rangefinder_sampling_ratio = 1.,
  odometry_sampling_ratio = 1.,
  fixed_frame_pose_sampling_ratio = 1.,
  imu_sampling_ratio = 1.,
  landmarks_sampling_ratio = 1.,
}

MAP_BUILDER.use_trajectory_builder_2d = true

TRAJECTORY_BUILDER_2D.min_range = 0.12
```

```
TRAJECTORY_BUILDER_2D.max_range = 3.5
TRAJECTORY_BUILDER_2D.missing_data_ray_length = 3.
TRAJECTORY_BUILDER_2D.use_imu_data = false
TRAJECTORY_BUILDER_2D.use_online_correlative_scan_matching = true

return options
```

Listing 6.2: Cartographer configuration

The configured map frame is `map` and the odometry frame is `odom`. The `provide_odom_frame` parameter is set, as the robot does not provide an odometry frame. The cartographer also provides a TF frame from `odom` to `base_footprint`. Based on this configuration, the cartographer uses the laser scan and the robot odometry as input, but does not consider the IMU values. Also, the configurations of the simulated LIDAR scanner such as minimum and maximum range are set accordingly. The robot's pose is published in the TF tree with a period of $30 \cdot 10^{-3}$ s, thus the frequency is 33.3 Hz.

Figures 6.5 and 6.6 show the TF tree without and with SLAM node. The graph is created with the `view_frames` tool of the `tf2_tools` ROS package. The rate of the `map` and `odom` frames in Figure 6.6 is approximately 33 Hz, indicating that the desired frequency is met in the SLAM node.

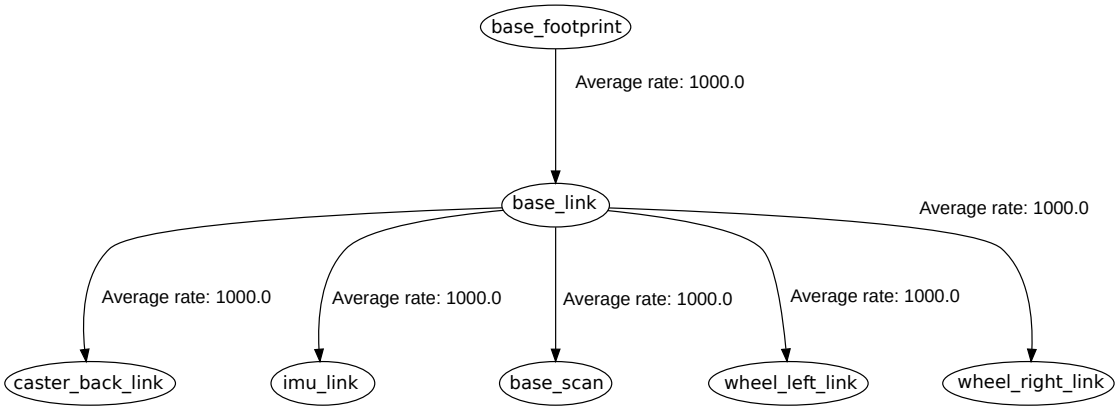


Figure 6.5: TF frames without SLAM

Figure 6.7 depicts the ROS nodes and topics after the SLAM node has been started, omitting the nodes and unused topics from the robot from Figure 6.3. The graph shows that the `scan` and `odom` topic are subscribed from the `cartographer_node`. The `cartographer_occupancy_grid_node` is using the input from the cartographer and publishes the global map of type `nav_msgs/msg/OccupancyGrid`.

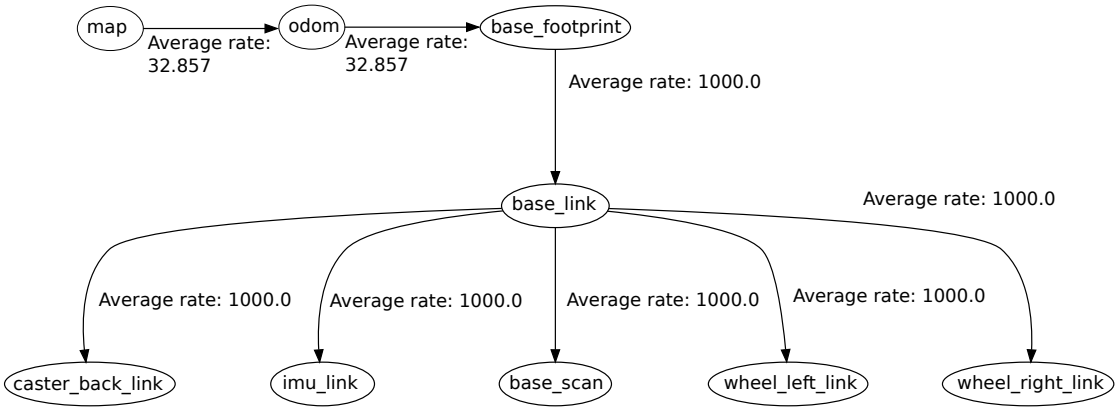


Figure 6.6: TF frames with SLAM

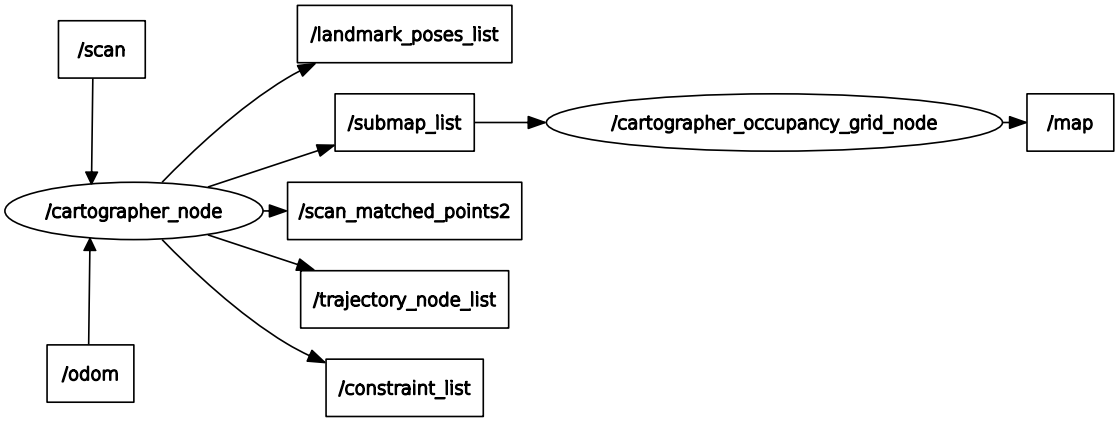


Figure 6.7: Cartographer nodes and topics

6.4 Communication of ROS2 Nodes

One of the most important differences between ROS and ROS2 is the way how the communication between nodes is handled. In the previous version – ROS – one node acts as ROS master and as a middleware for the publishers and subscribers. Thus, even though the communication of topics is decentralized in ROS, the centralized ROS node is mandatory for lookup between nodes.[63] ROS2 is based on the DDS, an open standard for decentralized distributed communication.[61] In the current ROS distributions, many DDS middleware implementations are available and can be exchanged. The default DDS middleware implementation in ROS2 is eProsima Fast DDS. Therefore, ROS uses its own protocol and communication middleware, whereas ROS2 builds upon existing DDS implementations. Profanter et al.[108] compared the performance of Fast DDS and the ROS communication protocol and found that Fast DDS has a significantly better performance.

By default, the discovery of nodes is done in a decentralized fashion. This is achieved by using multicast messages. However, as described in the Fast DDS documentation[W32], this approach has two drawbacks:

- It does not scale well, as the amount of messages grows rapidly when new nodes are introduced
- It may not work well in certain environment such as WiFi, the 5G use case or with virtualization

In the setup described in this thesis, the issue is that WiFi and 5G is used as communication layer between different ROS2 nodes. Additionally, some of the components are hosted in Docker containers. Thus, the multicast mechanism does not work and discovery fails. There are two solutions to this problem: (1) Use an overlay network such as a VPN or (2) use the Fast DDS discovery server. The discovery server acts in a similar fashion as the ROS master from the previous ROS version. It is hosted in a central location and assists the client in discovering the different publishers and subscribers. In this thesis, the discovery server approach has been chosen. The main reason is that the goal is to evaluate low latency solutions and a VPN may add additional latencies.

CHAPTER 7

Evaluation of Robotic Application and MEC Prototype

In this chapter the edge computing prototype is evaluated using the simulated robotics experiment described in Chapter 6. Herein, the different deployment options are defined, together with the evaluation criteria. Furthermore, the results of the latency measurements and of the SLAM accuracy are reported and discussed.

7.1 Setup of the Experiment

Research Question 3 (*RQ3*) asks how offloading a ROS2 SLAM node to the edge of a 5G network affects the latency and the quality and functionality of the produced output. To show this, the SLAM node is deployed in different physical locations. The local deployment is considered the baseline scenario, having negligible latency between the ROS nodes. To answer *RQ3*, the SLAM node is deployed in the edge using the edge computing prototype with the OAI, which has been created as part of this thesis (see Chapters 4 and 5). Furthermore, the performance is compared to a WiFi based deployment, as this is currently the most common used method of offloading resource-intensive tasks in ROS. Table 7.1 shows these deployment options.

Containerization of the SLAM node

As described in Table 7.1, the simulation nodes always run on a client, either on a WiFi terminal or on a 5G terminal. In fact, this can be the same hardware, therefore, the simulation setup does not need to be moved to other locations. The SLAM node on the other hand has to be deployed on different servers. Thus, it is containerized using Docker to ease the deployment.

Details about the Dockerfile are described in Section C.1 in Appendix C.

Deployment Option	Simulation nodes	SLAM nodes	Expected Latency [ms]
local, baseline	local	local	<1
5G edge	5G terminal	5G edge	<15
WiFi	WiFi terminal 1	WiFi terminal 2	<10

Table 7.1: Deployment options for the simulation nodes and SLAM nodes for the experiment

Hardware Specifications

Table 7.2 shows the hardware specifications of the different servers and computers used for all the deployment options.

Name	CPU model	Cores	Speed [MHz]	RAM [GB]
simulation_computer	AMD Ryzen 7 5800H	16	4460	16
gnb_server	Intel Xeon Gold 6154	36	3000	64
cn_server	Intel Xeon E5-2690	20	3000	64
wifi_server	AMD Ryzen 7 3700X	16	4400	16

Table 7.2: Hardware specifications of the computers used in the deployment options

Local Deployment

In the local deployment, all the ROS nodes are running on the same machine, the `simulation_computer` from Table 7.2. The Fast DDS discovery server and the ROS2 SLAM node are running in Docker containers.

Deployment in the 5G Edge

Executing the experiment in the edge of the OAI 5G has several pre-requisites:

1. Deploy the OAI 5G CN on a server
2. Deploy the OAI gNB
3. Establish a PDU session using a 5G terminal
4. Deploy the SLAM node in the edge

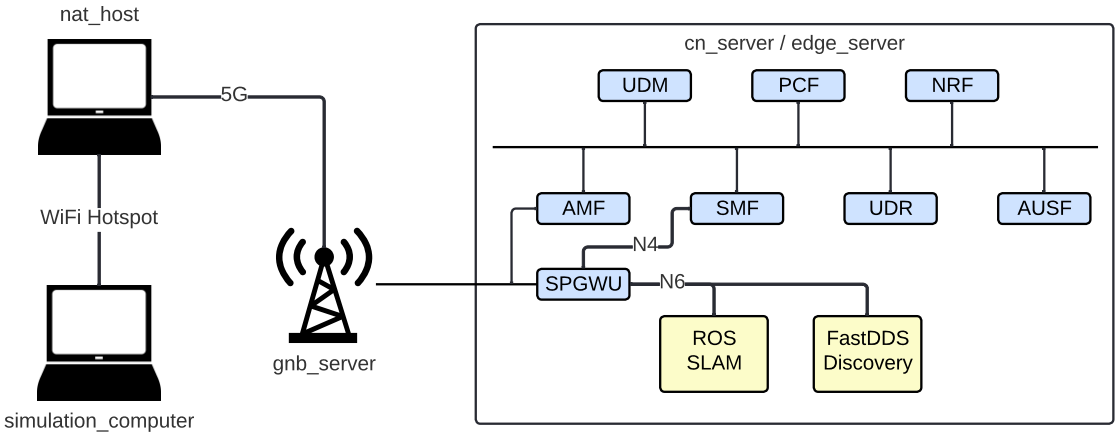


Figure 7.1: Robotics experiment 5G edge deployment setup

The setup is depicted in Figure 7.1. The CN consisting of SPGWU, AMF, SMF, UDR, AUSF, UDM, PCF and NRF is deployed on one single server in EURECOM's server network, the `cn_server`. It is deployed using `docker-compose`, as described in the OAI documentation[W33].

The OAI gNB is deployed in another server in the same network. The gNB uses band n78 in UL and DL with a 20 MHz channel bandwidth. SCS is set to 30 kHz, thus numerology 1 is used. Seven slots are reserved for DL and two slots for UL. As RU the AW2S[W19] is used. The gNB is running on the server `gnb_server` from Table 7.2 on bare-metal.

The 5G terminal used for the experiment is a Quectel RM500Q. The modem configuration has been changed to allow the configuration of custom PLMNs and DNNs. The driver used for the Quectel modem was not supported on Ubuntu 22.04. Thus, another computer with Ubuntu 20.04 is used as 5G UE. Both computers are connected using a WiFi hot-spot and configuring NAT and port forwarding on the 5G UE. This adds additional latencies, as reported in Section 7.4. The simulation is running on the computer `simulation_computer` from Table 7.2.

The SLAM node and the Fast DDS discovery server are deployed on the same server as the 5GC (`cn_server`) using `docker-compose`.

The scenario depicted in Figure 7.1 uses the distributed anchor point access model. There is one UPF – the SPGWU – deployed directly at the edge. The core network is also deployed at the edge, although this is not a requirement. The reason this deployment has been chosen for the experiment is a performance issue on the VPP-UPF. When the VPP-UPF is deployed as Docker container, the maximum DL bandwidth is around 1.5 Mbit/s. As this bandwidth is not sufficient for exchanging the ROS messages between the SLAM node and the `simulation_computer`, the VPP-UPF could not be used for this setup. As the SPGWU does not support the UL CL scenario, the setup only works with the distributed anchor point access model.

Nevertheless, the author's additions in the SMF and the PCF described in Chapter 5 are used for the deployment, although strictly speaking not necessary. In this setup, the UL CL is not required, as the whole 5GC is deployed at the edge. Thus, the latency t_{mno} is not a concern. Additionally, the ROS nodes are deployed on the same machine as well, resulting in a negligible t_{transit} and $t_{\text{datacenter}}$. While this scenario is sufficient to showcase the 5G edge computing use case and provides valid data, it does not cover all use cases. The setup depicted in Figure 7.1 can serve as an example for a private 5G network, where a company operates its own 5G network and deploys the gNB and the 5GC on-premise. The disadvantage of this approach is that it misses the ubiquity of edge computing. The setup may work for this specific setup, but as soon as a second production site is introduced to the company, the limits of this solution become apparent.

Even without the support of the UL CL on the UPF, the author's contributions on the PCF and the SMF allow to select different UPFs and thus different edge DNs by changing the PCC rules on the PCF, without the need to reconfigure the 5GC.

WiFi Deployment

The simulation is running on the computer `simulation_computer` with a WiFi 2.4GHz connection using the IEEE 802.11n (WiFi 4) standard. The SLAM node is running on the computer `wifi_server`. The `wifi_server` uses an Ethernet connection, but is located in the same LAN as the `simulation_computer`.

When deploying the Fast DDS discovery server and ROS2 nodes in a docker network, there is an issue with networking due to the used Real Time Publish Subscribe Protocol (RTPS). The ROS2 nodes advertise their IP addresses in the protocol towards the Fast DDS discovery server. When this is done inside a Docker container, the IP address of the Docker network is used. Thus, all the containers in this Docker network can communicate with each other. However, this IP range is not routed in the host network. Therefore, to have communication between the nodes on the `wifi_server` and with the nodes on the `simulation_computer`, the discovery server is not used. The Docker containers use the `host` network mode and the multicast discovery mechanism, as described in Section 6.4.

7.2 Considered Performance Metrics

For each of the different deployment options, the latency and available bandwidth has been measured. The latency is measured using the `ping` utility with an interval of 0.01 seconds and 1000 sent packets. The bandwidth is measured using the `iperf3` tool with a 20 seconds transmit interval and one second between periodic throughput reports.

The measured latency and bandwidth are just an indicator that the setup works as intended. The latency measurements with ping are not reliable enough, because it gives the latency when the network is not under load. Bandwidth measurements with `iperf3` on the other hand utilize the capacity of the network, but do not use real data. Thus, the

metrics of the robotic simulation need to be analyzed as well. For each of the experiment runs, ROS bags are recorded. They allow to store all the exchanged messages from all the topics in a ROS-specific format. As described in Section 7.3, the ground truth pose is published together with the estimated pose. The ground truth can be used as a baseline to evaluate how well the localization of the SLAM algorithm performs. Different results from the deployment options are evaluated towards the baseline and the performance is compared.

The results are reported in Section 7.4.

7.3 Evaluation with Robotics Experiment

Section 6.2 describes how the robot simulation is set up. This is sufficient to qualitatively evaluate the performance by analyzing the visualization provided by the `rviz` tool, as described by the `rviz` GitHub repository[W34]. This tool is capable of visualizing the created map together with the current robot position. However, this approach is an estimate and may only be used to evaluate if everything is working as expected.

One approach to measure the accuracy of the localization is by comparing the pose of the ground truth with the estimated pose. The ground truth is published by Gazebo as `nav_msg/Odometry` message, and the estimated pose is published in the TF tree, the transformation from the global map frame to the odom frame and to the base footprint frame. To correlate these, the author has written a ROS2 node that subscribes to the ground truth as well as the TF tree. Based on this information, the topics `ground_truth/pose` and `real_pose/pose` are published. Both have the message type `geometry_msg/PoseStamped`. This is a ROS2 message that contains a timestamp and a pose. This timestamp allows to align the different poses for the evaluation.

As the experiment shall be repeatable, the robot should follow the same path for each execution of the experiment. This is not a trivial task. Due to the inherent inaccuracies of the drive and the sensors, it cannot be guaranteed that the robot follows the exact same route. Usually, the map is created by manually steering the robot until the generated map "looks good" in `rviz`, meaning that all the obstacles are drawn in the map. Then, the ROS2 navigation stack can be used to navigate the robot in the world and avoid obstacles. At this stage, the localization and the quality of the map may be compared.

This approach is not feasible, as the robot should follow the exact same route. To accomplish this, two methods are available:

- Write a ROS2 steering node that follows a given path using the `cmd_vel` topic
- Use the ROS2 navigation stack and provide it with a fixed path

Both of these methods have advantages and disadvantages. The first method requires that a steering algorithm is implemented. As described by Macenski et al.[62] – who created the ROS2 navigation stack – robust robot steering requires handling several different use cases and scenarios. For example, it is not enough to give the robot a linear velocity and expect that it reaches a certain pose in the map eventually. The current pose of the robot has to be taken into account. Even then, when steering for example one meter straight ahead and the robot orientation is only some degrees off of the expected value, the robot might never reach the target. Therefore, steering algorithms need to have a subroutine to approach a target, which includes error correction.

The ROS2 navigation stack performs best on an existing map. Using it to steer in an unmapped environment may create significant challenges. The navigation stack uses local and global costmaps to generate a path to a target. However, when these costmaps do not exist yet and are continuously updated, the robot might get stuck and has to execute a strategy, such as returning to the last known position. These movements might be abrupt and include a lot of angular movements. This can negatively impact the quality of the produced map.

For the evaluation of the prototype, both approaches have been used, although only the first approach is used for a quantitative comparison. The reason is that the behavior of the navigation stack is too nondeterministic. In several runs of the experiment the robot might follow the intended path, but in some it may get stuck, resulting in unexpected behavior. Due to the complexity of implementing a full steering solution, the robot follows a very simple path for the quantitative analysis.

For each deployment option, the experiment is executed in a reproducible and comprehensible fashion. A ROS2 launch file is used to launch all the ROS2 nodes in the specified order. Additionally, a `rosbag` is generated. This file allows to store all exchanged ROS2 messages for the duration of the experiment. The tool `evo` is a tool to quantitatively analyze the accuracy of SLAM algorithms, as described by its documentation[W35]. It is used to print the graphs of the absolute pose error reported in Section 7.4. Section C.3 in Appendix C provides details how `evo` is used to calculate the absolute pose error and generate the graphs.

7.4 Results

In this section, the latency and bandwidth measurements are reported, together with the accuracy of the SLAM node in the different deployments.

7.4.1 Latency

Table 7.3 shows the latency measurements between different hosts. For the WiFi deployment option, only the first row is relevant. It can be seen that the WiFi deployment has a very low t_{mean} and σ_t.

The last three rows in Table 7.3 are all relevant for the 5G edge deployment. As described in Section 4.1, the overall response time consists of different transmission times, most importantly t_{access}, t_{mno} and t_{transit}. The latency between the `gnb_server` and the `cn_server` corresponds to t_{mno}. The latency for t_{transit} is not measured, as the edge application is running on the same physical machine as the core network and thus the UPF. The latency t_{mno} is negligible with t_{mean} of 0.1 ms. Thus, it can be concluded that the latency measurement between `simulation_computer` and `cn_server` is approximately t_{access}. As described, t_{access} includes the latency between `simulation_computer` and `nat_host`, the 5G terminal.

The latency t_{access} has a relatively high standard deviation σ_t. Also, the maximum latency of 23 ms is very high, even when considering that it also includes a WiFi connection. Dürre et al.[109] have reported a similar phenomenon in the OAI RAN, where the latency is relatively high for the first packet and then gradually decreases. At the lowest point, the latency abruptly increases to the starting value and decreases again. This pattern has also been observed during this experiment, explaining the comparably high σ_t.

Client	Server	t_{min} [ms]	t_{max} [ms]	t_{mean} [ms]	σ_t [ms]
simulation_computer	wifi_server	0.8	4.5	1.7	0.4
simulation_computer	cn_server	7.1	23.2	10.6	2.8
simulation_computer	nat_host	0.7	4.2	1.2	0.3
gnb_server	cn_server	0.1	0.1	0.1	0.002

Table 7.3: Latency measurements between hosts

The latency measurements for the 5G scenario are higher than expected. While t_{mean} is 10.6 ms and thus below the expected latency of 15 ms described in Table 2.1, t_{max} is 8.2 ms higher than the expected value. One of the reasons why this is the case may be a non-optimized gNB configuration. While the author configured the gNB to his best knowledge, there may be different configurations or experimental branches to further optimize the latency. Additionally, the gNB is in constant development and some URLLC features are not yet implemented. Furthermore, as the example of the VPP-UPF indicates, containerizing a UPF can result in unexpected performance losses. Initial, not yet published, experiments by the OAI RAN engineers have shown that a great source of jitter is the SPGWU and the Quectel UE. The results show that there is a lot of potential for further research and improvements in the OAI.

7.4.2 Bandwidth

The bandwidth is measured between the same hosts as the latency. There is one measurement in the UL and in the DL direction between each host. The reported bandwidth is the mean bandwidth over the measurement period of 20 seconds. The measurement results are described in Table 7.4. It can be seen that the DL bandwidth

in the WiFi and the 5G deployment is nearly identical. There is a crucial difference in the UL direction, though. While WiFi is synchronous, the gNB configuration allocates more resources in the DL than in the UL direction. Thus, the UL bandwidth in the 5G scenario is approximately 10 Mbit/s.

Client	Server	UL [Mbit/s]	DL [Mbit/s]
simulation_computer	wifi_server	57.8	54.4
simulation_computer	cn_server	9.8	53
simulation_computer	nat_host	105	106
gnb_server	cn_server	939	941

Table 7.4: Bandwidth measurements between hosts

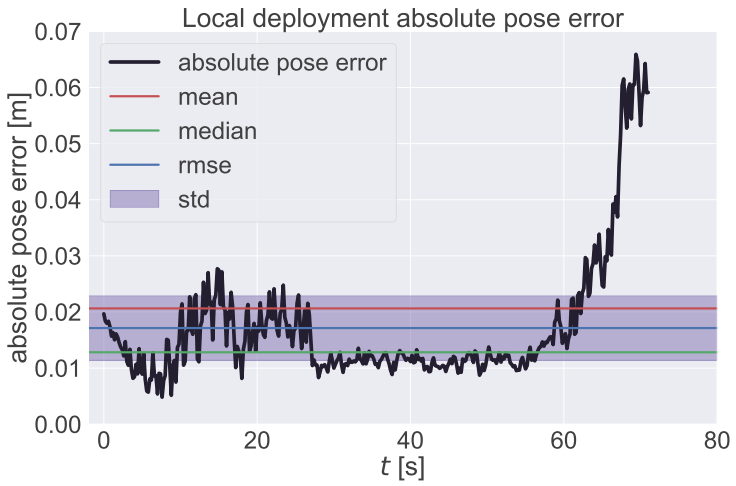
As the gNB is using a 20 MHz channel bandwidth, the bandwidth results for the 5G deployment are slightly lower than expected. According to the OpenAirInterface roadmap presentation from July 2022[W36], a DL bandwidth of 131 Mbit/s and a UL bandwidth of 21 Mbit/s has been achieved using a 40 MHz channel bandwidth with the same DL and UL slot configuration. One would expect that doubling the channel bandwidth would result in a 200% increase in bandwidth. While this is approximately the case for the UL direction (from 9.8 Mbit/s to 21 Mbit/s), the DL bandwidth measured in this experiment is slightly lower. The expected value is 65.5 Mbit/s. However, the different measurements use different RUs. While the AWS2 RU is used in this thesis, the reported 131 Mbit/s are achieved using a USRP N310. As the RU is a crucial factor in gNB performance, it may not be reasonable to expect the same linear correlation between channel bandwidth and bandwidth for different RUs.

The bandwidth requirements for the robot simulation have been measured using the local deployment. As the SLAM node and the Fast DDS discovery server are running in their own Docker network, the bandwidth of the docker bridge can be measured using the Linux `nload` utility[W37]. During the execution of the experiment, the UL bandwidth never exceeds 7 Mbit/s with the sole exception of the start of the simulation, before starting the SLAM node, where the UL bandwidth spikes at 10 Mbit/s. The DL bandwidth never exceeds 5 Mbit/s. Therefore, the measured bandwidths in the WiFi and 5G deployment option are no limiting factor for the experiment.

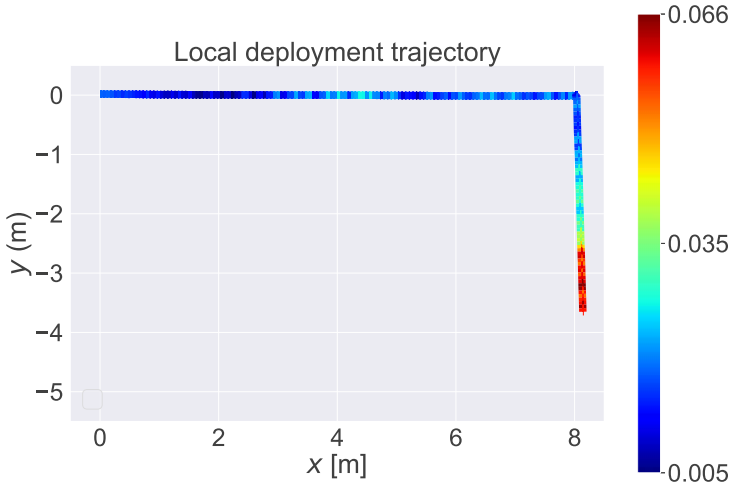
7.4.3 SLAM accuracy

The accuracy of the SLAM node is measured by the absolute pose error between the ground truth pose and the estimated pose. As described in Section 7.3, the simulated robot is programmed to follow the exact same path for each of the deployment options. Hence, the results are comparable.

Figure 7.2a shows the absolute pose error of the local deployment and Figure 7.2b shows the trajectory of the robot.



(a) Absolute pose error of the local deployment



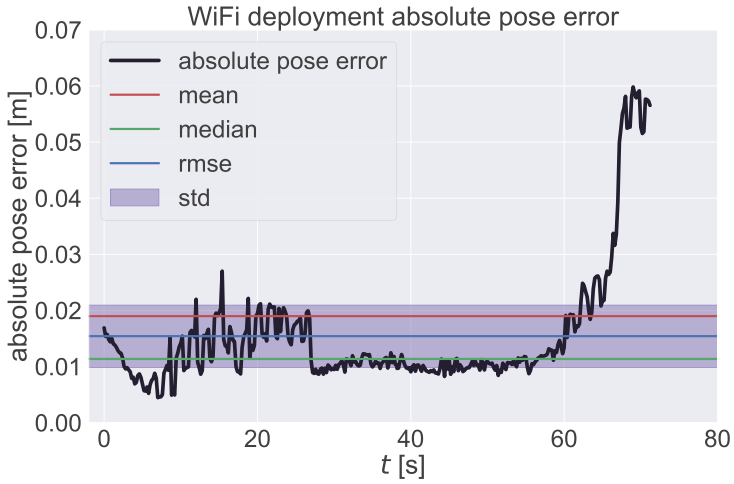
(b) Trajectory in the local deployment

Figure 7.2: Results of the local deployment

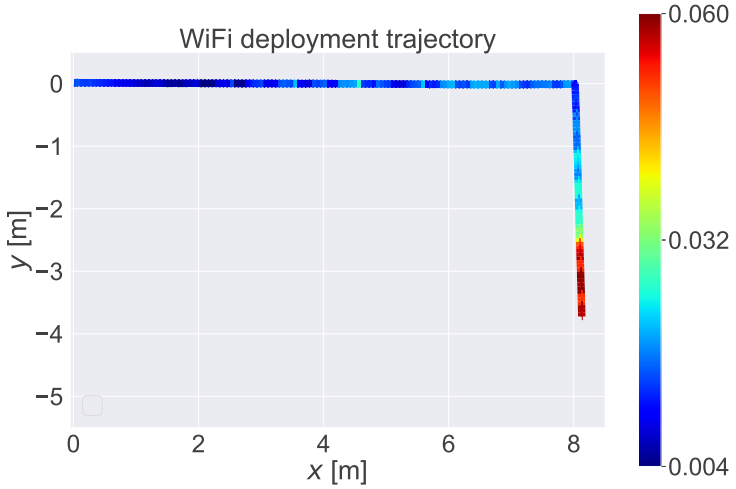
It can be seen in Figure 7.2a that the absolute pose error is minimal for the first minute of the experiment, whereas the pose error steeply increases in the last 10 seconds of the experiment. Figure 7.2b gives an indication where in the map this happens. The robot drives straight for 7 meters with a very accurate localization. At this point, the robot turns 90 °, and the accuracy rapidly declines. As the robot did not create a map in this unexplored part of the environment, the SLAM algorithm can solely rely on the robot's measurements, i.e., the odometry. This is an example why SLAM is such an important task in robotics, as the odometry can become highly inaccurate over time. As the odometry measures how far the robot's wheels have turned, this can become even more inaccurate after and during a rotational movement, as the wheels are also moving. As this is observed in the local deployment without any additional latencies, the

latency factor can be excluded in reasoning why this happened. Even though the results of the SLAM algorithm are not optimal, they can nevertheless be used to assess whether SLAM can be off-loaded to the 5G edge, one of the underlying research questions of this thesis. The Umeyama alignment is used to compare the similarity of the trajectories, as described by Umeyama[110].

Figures 7.3a and 7.3b show the absolute pose error and the trajectory for the WiFi deployment, respectively.



(a) Absolute pose error of the WiFi deployment



(b) Trajectory in the WiFi deployment

Figure 7.3: Results of the WiFi deployment

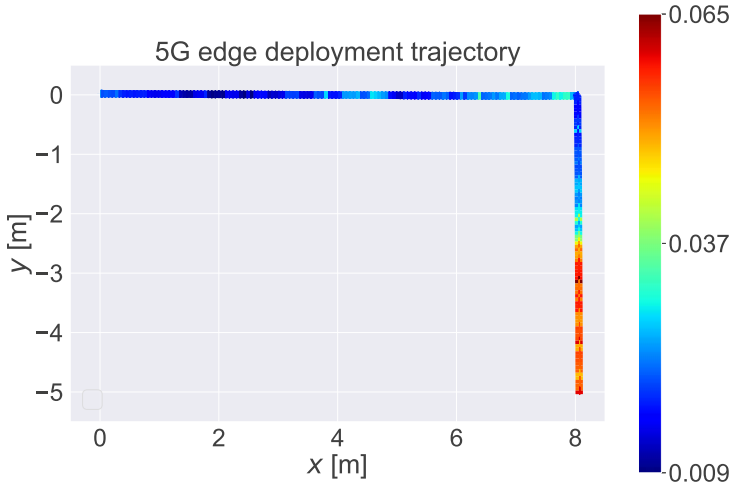
Comparing the local (Figure 7.2) and the WiFi deployment (Figure 7.3) shows that the accuracy of the localization is very similar. The absolute pose error as well as the trajectory follow the same pattern. In the first minute, the absolute pose error is between

0.5 cm and 3 cm. In both experiments there is a small spike in the pose error between 10 and 30 seconds, and then the pose error is stable around 1 cm. An absolute pose error of 1 cm is very accurate, especially considering that the robot itself has a radius of 10 cm. Thus, the minimum pose error is around 10% of the size of the robot. The comparison also shows that the WiFi deployment scenario performs slightly better. There is only one spike up to nearly 3 cm accuracy, whereas the local deployment has several spikes at around 15 seconds. The same is true for the last 10 seconds of the experiment, where the accuracy deteriorates rapidly in both scenarios. The maximum pose error in the local deployment is 6.6 cm and in the WiFi deployment it is 6 cm. Given that the same configuration is used for both scenarios, it cannot be concluded with certainty why the WiFi scenario performs slightly better. As the experiment is running on a non-deterministic simulation with a lot of built-in randomness, this might just be a result of this fact. On the other hand, the simulation and the SLAM node use a lot of processing power. Therefore, it may also be the case that the localization is more accurate, as the offloaded SLAM node is able to match its desired loop closure rate better when running on `wifi_server`. What can be concluded without doubt is that off-loading of the SLAM algorithm over WiFi did not have any significant negative impact on the functionality and the quality of the SLAM localization.

The results for the absolute pose error and the trajectory of the 5G edge deployment are shown in Figures 7.4a and 7.4b.

While the fact that the WiFi deployment does not have a negative impact on SLAM is an important finding, the goal of this thesis is to offload SLAM to the 5G edge. Thus, the local (Figure 7.2) and 5G edge deployment (Figure 7.4) are compared as well. It can be seen that the edge deployment follows the same trend as the local and WiFi deployment. However, there are some significant differences. First of all, the overall pose error is higher than in the local deployment. This is especially true for the stable pose error between 30 and 60 seconds. While the local deployment has an error of around 1 cm, in the 5G edge deployment this error is slightly under 2 cm. Also, the first spike between 10 and 30 seconds shows that the highest absolute pose is above 3 cm, which is never the case for the local deployment. The maximum pose error after the 90 ° turn is very similar to the local deployment and is around 6.6 cm. While a median absolute pose error of 1.8 cm may still be acceptable – depending on the scenario – there is no doubt that the offloading of the SLAM algorithm to the 5G edge has a negative impact on the functionality and the quality of the SLAM localization. There are two reasons why this is the case. First of all, when comparing the latencies described in Table 7.3, one can see that the mean latency of the 5G edge deployment is 10.6 ms, which is approximately 6 times higher than the mean latency of the WiFi deployment. Also, there is a considerable jitter, the deviation of the packets from the mean value. While the standard deviation is only 0.4 ms in the WiFi deployment, it is 2.8 ms in the 5G edge deployment. As discussed, this behavior has been observed before in the OAI 5G. This shows that the latency itself as well as the jitter has an impact on the accuracy of the SLAM localization.

(a) Absolute pose error of the 5G edge deployment

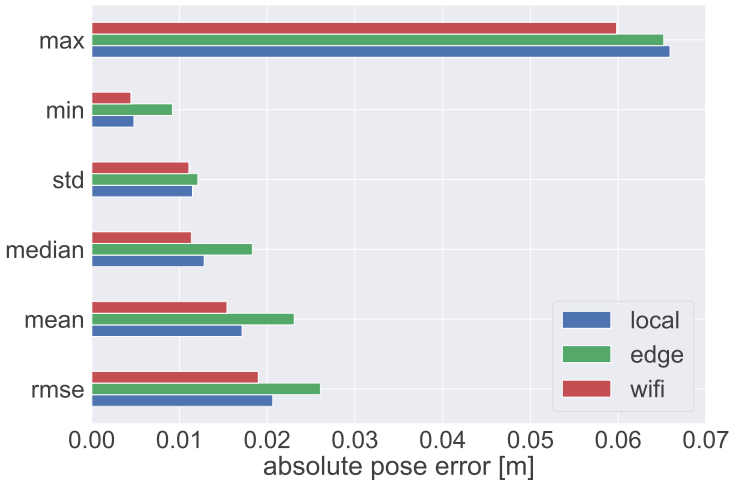


(b) Trajectory in the 5G edge deployment

Figure 7.4: Results of the edge deployment

Figure 7.5a shows the minimum, maximum, mean, median, standard deviation and root-mean-square error of the different deployments and Figure 7.5b compares the distribution of the absolute pose error. Figure 7.5b shows very well that the absolute pose error is higher in the edge deployment.

Table 7.5 shows the numeric values for the absolute pose error depicted in Figure 7.5a.



(a) Absolute pose error of all deployments

(b) Distribution of the absolute pose errors of all deployments

Figure 7.5: Comparison between the deployment options

	rmse [cm]	mean [cm]	median [cm]	σ[cm]	min [cm]	max [cm]
local	2.1	1.7	1.3	1.1	0.5	6.6
edge	2.6	2.3	1.8	1.2	0.9	6.5
wifi	1.9	1.5	1.1	1.1	0.4	6

Table 7.5: Numerical values for the absolute pose error

As depicted in the trajectories in Figures 7.2b, 7.3b and 7.4b, the experiment does not end at exactly the same position. This is because the inaccuracy of the robot position after the turn made it impossible to reach the target pose with the simplified steering

algorithm. Thus, the experiment had to be manually stopped. As depicted in Figure 7.4a, in the edge experiment the trajectory where the localization is not accurate anymore continued longer than for the other experiments. While this has a negative impact on the mean accuracy, the figures clearly show that the pose error is larger compared to the local deployment before the turn.

One of the reasons why the accuracy degrades is that the rotational movements are not correctly aligned with the real robot's position. It can be seen in the trajectories that the linear movement did not cause large pose errors. The increase of the pose error started after the 90 ° turn. There may be several reasons why this is the case. First of all, the naive steering implementation does not take the robot's non-holonomic drive into account. Further, the SLAM algorithm does not consider the robot's IMU. However, even enabling reading the IMU measurements did not increase the accuracy. This may be due to a high inaccuracy of the IMU sensor. In that case, the SLAM algorithm can be configured to weight the IMU measurements less than other sensors such as LIDAR. In any case, as the robot is driving through unmapped territory, the SLAM algorithm can only localize based on the robot's sensors. In case the robot drives back to the original starting point, the SLAM algorithm is able to localize the robot based on the created map and correct the pose offset.

Even considering the inaccuracy of the last part of the experiment and the slightly worse performance of the edge computing deployment, it can be concluded that the offloading of the SLAM node worked and the results are comparable with a local or a WiFi deployment. However, the hypothesis that the quality and the functionality of the SLAM algorithm is not affected is refuted by this results. Even though the slight offset may be considered acceptable, it is clear that the WiFi deployment option performed better. Whether the drop in accuracy is acceptable or not depends on the robot platform, the environment and the requirements. The median absolute pose error is 63% higher in the edge deployment w.r.t. the WiFi deployment. Although this increase is substantial, in absolute numbers it is 0.7 cm, a relatively small value. What this experiment has shown without doubt is that an increase in latency and high jitter negatively affects the quality of a ROS2 SLAM node.

Created Map

Figure 7.6 shows the robot visualization including the map created in the 5G edge deployment option shortly before the turn. The red arrow is the ground truth pose and the violet arrow is the estimated pose. Figure 7.7 is the visualization from the same experiment, shortly after the turn.

While the map itself is not quantitatively compared between the different deployment options, it gives a good qualitative estimate on how well the SLAM algorithm performs. In the representation from the `rviz` tool in Figures 7.6 and 7.7, the black edges represent obstacles and the grey area is explored and marked as obstacle-free. The red dots are the visualized LIDAR scans. However, the confidence is not great (indicated by a darker

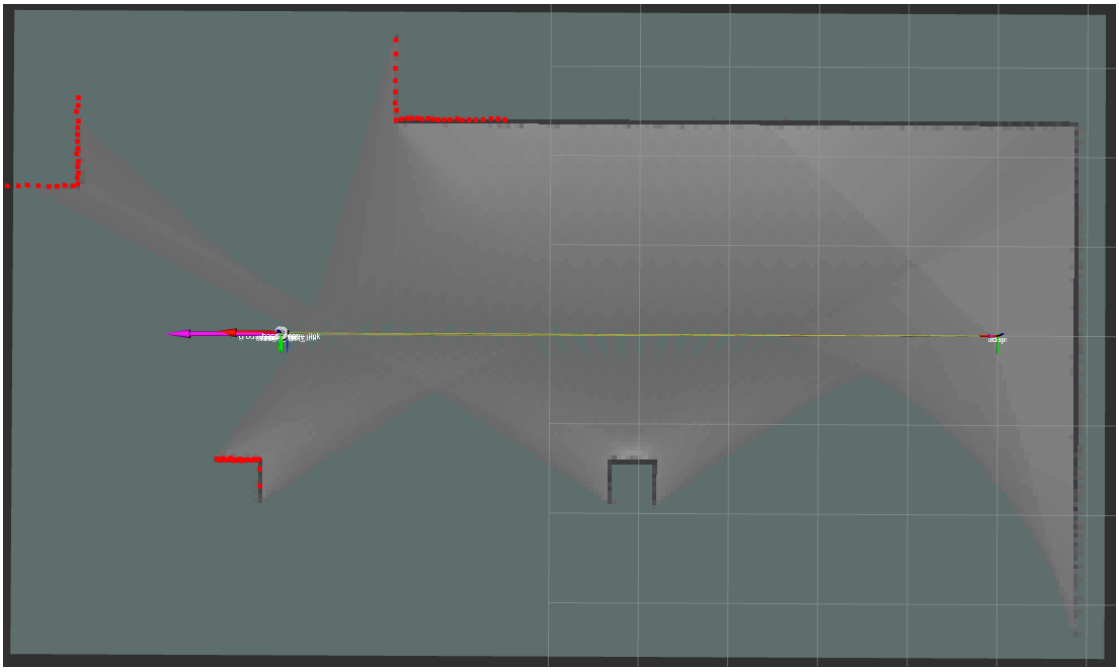


Figure 7.6: `rviz` visualization before the turn

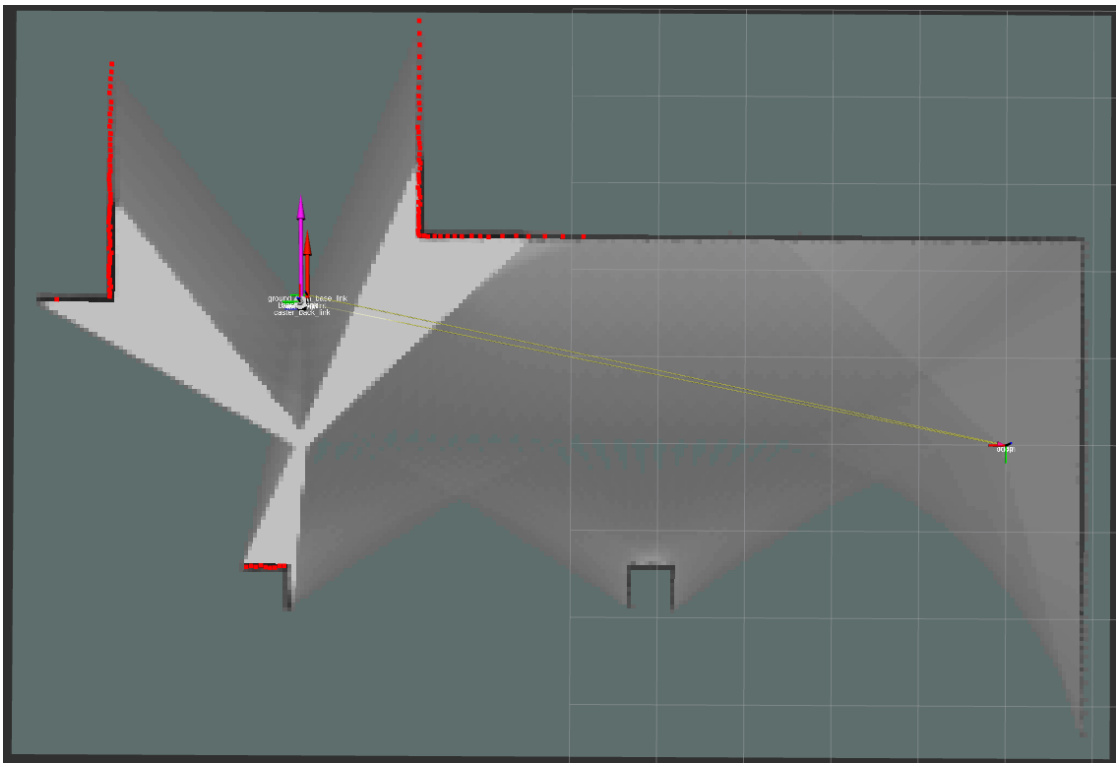


Figure 7.7: `rviz` visualization after the turn

gray), as the robot only moved once in a straight line. Figure 7.7 shows that the certainty of the map increases when the robot rotates. This behavior is one of the reasons why mapping is often done manually. The robot is manually steered through the world until the map "looks good". However, the map in Figure 7.6 is relatively accurate. This is indicated by the fact that there are no distortions of the edges, i.e., all the edges are parallel and straight. In case the robot's localization becomes too inaccurate, the SLAM node publishes a new transformation from the `map` frame to the `odom` frame in the `tf` tree. This may result in a shift of the already explored map and the mentioned distortions. This is slightly visible in Figure 7.8, although the map is still of an arguably high quality, given the fact that robot moved only through unmapped territory so far.

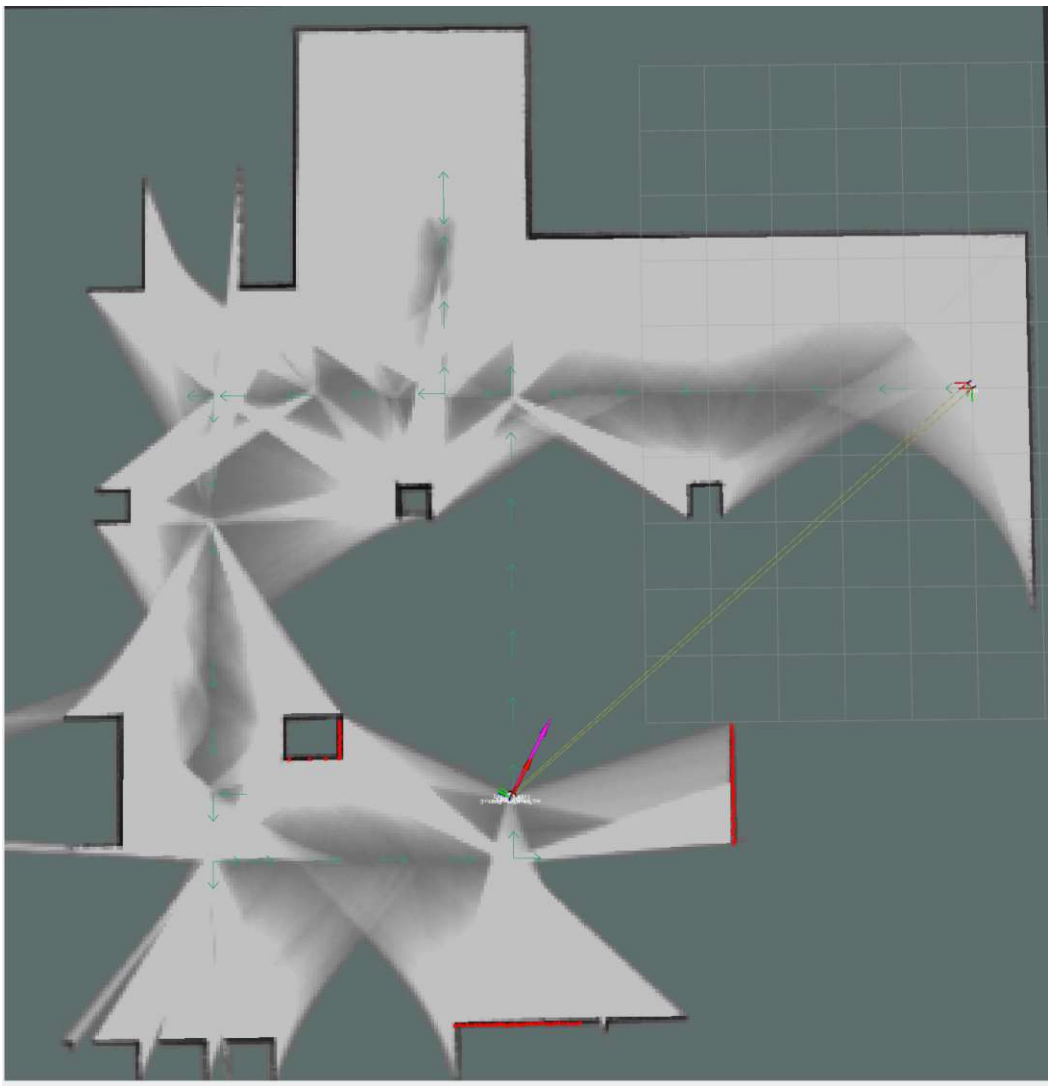


Figure 7.8: `rviz` visualization of experiment using the navigation stack

Furthermore, it can be seen that the ground truth pose and the estimated pose coincide very well in Figure 7.6, while Figure 7.7 shows an offset between the two arrows. While the offset does not look significant in this visualization, it relates to the 6 cm offset depicted in Figures 7.2, 7.3 and 7.4. These figures visually support the author's claim why the worsened quality of the 5G edge deployment can still be considered as functional. The increased median absolute pose error of 0.7 cm in the edge deployment is barely visible in this visualization. This does not mean that there is no negative effect on the overall quality of the map and the localization, but in a real-world robotics scenario inaccurate localization is a common issue and most steering algorithms and navigation stacks are capable of circumventing this issue to a certain extent. Nevertheless, the simulated environment is large compared to the robot and there are no particularly difficult spots such as tight space. As described, it depends on the actual field of operation of the robot whether the increase in pose error is acceptable or not.

Figure 7.8 shows the created map in a scenario where the navigation stack has been used to navigate through the map in the local deployment. The small green arrows in the map are the target poses of the navigation stack. It proved to be more stable when the next goals are not further apart than one meter while the map is not yet available.

CHAPTER 8

Discussion

The author of this thesis has provided an analysis of edge computing architectures in 5G in Chapter 4. Although the topic itself has gained a lot of traction, the different standards and white papers provided by ETSI and the 3GPP do not give a clear indication how an edge computing solution should look like. As discussed, the ETSI MEC framework uses an unspecified interface between the MEC platform and the 5G core network and thus there may be several different approaches in solving this challenge. The novel 3GPP EDGEAPP architecture on the other hand specifies this interface. However, the related standards are part of Release 17 and have been released during the time of writing of this thesis. Thus, it is expected that they may lack some maturity. Based on the available standards, the author has proposed different options in creating an end-to-end edge computing solution.

Nevertheless, the expected adaptations and challenges are great. On the one hand, the existing 5G implementations need to be adapted, be it in the RAN or the 5GC. In the 5GC, at least the SMF, the UPF and the PCF must be updated to support this specific use case. On the other hand – on the edge enabler side – either the MEC framework or the 3GPP EDGEAPP may be used. The MEC framework has a level of maturity and is already adopted. Still, the weak standardization of the interworking between MEC and 3GPP may result in several incompatible implementations by vendors. The 3GPP EDGEAPP is a novel approach and there is the need of a prototype implementation to showcase the usability of the solution. Another challenge is bringing application developers on board. Due to the edge-awareness of the devices in the EDGEAPP architecture, application developers who want to utilize edge capabilities need to consider this in the design and implementation. In the author's opinion, there is the need for a library for iOS and Android, but also for embedded systems, to support developers in using the edge. This allows to hide the complexity of the edge computing solutions and give developers a straight-forward interface. As the interface between the edge enabler client and the application itself is not specified, there is the need for the open source community and

the industry to come together to create a robust library which interacts with an edge enabler client.

It remains to be seen which architecture and technical solution for 5G edge computing will prevail. As discussed in Section 2.1.6, network slicing may be a competitive solution to the one presented in this thesis. However, in the author's opinion, the network slicing solution is not flexible enough to support all edge computing use cases. Also, creating a network slice for each campus network of each customer may turn out to be overly complicated and resource inefficient. Furthermore, although edge computing has many great applications, it is not decided yet if the commercial incentives are high enough for edge computing as discussed in this thesis to be broadly adopted in the future. It may also be the case that industry leaders prefer a private 5G solution with a private cloud, as it is simpler to deploy and it gives them a certain freedom w.r.t. MNOs. Nevertheless, there are still open issues in these scenarios such as a robust support for mobility.

As discussed in Chapter 4, there are different approaches in enabling edge computing within the 5GC. First and foremost, there are three architectures how an edge computing AF can interact with the 5GC: (1) Directly with the PCF, (2) Using the NEF, which interacts with the PCF and (3) Utilizing the UDR. While option (1) is the simplest, it is likely that option (2) will be adopted in commercial networks. The main reason is that MNOs do not trust 3rd-party equipment. Thus, the NEF is used as a security gateway. Option (3) seems overly complex, but it has the advantage that the AF is able to persist its decisions in the database of the UDR. This again depends on the use case and the MNO network at hand. 5G edge computing vendors will most likely have to support all options, further increasing the complexity and development effort of an already difficult topic. Also, in the author's opinion, the decision of the 3GPP to have options (1) and (2) support individual UEs and option (3) multiple UEs is not justified. There is no technical reason why this is necessary. This exemplifies one of the major issues with 3GPP mobile networks: The standardization is unnecessarily strict at certain points, while it gives too much freedom at other places. Although this freedom seems beneficial, it can easily lead to incompatible implementations by different vendors, resulting in a vendor lock-in for MNOs. This is one of the reason why open source approaches such as the OAI are so valuable. Application developers have the possibility to use a 5G network "as-is", without the need to read thousands of pages of standards just to understand in theory how a 5G network shall operate. Although there are different ways for an AF-5GC interaction, the PCF plays a vital role in all of these. While this design decision is understandable, it breaks the micro-services approach. The PCF has one large API used for QoS, edge computing, TSN and other policies. There is the danger that the PCF is being implemented as a monolithic policy component. It may be a good approach to split the different policies into micro-services and let an API gateway act as a monolithic PCF towards the 5GC. However, as there is only one API towards the SMF, this approach requires substantial engineering effort to split the large `PolicyDecision` model into several parts. For readers with knowledge of the 4G equivalent – the Policy and Charging Rules Function (PCRF) – it becomes apparent that the PCF is an incremental update of

the PCRF, whereas REST is used instead of Diameter. In the author's opinion, this is a missed chance for a true cloud-native 5G NF.

Another example where the standard gives much freedom are the different UPF edge computing deployment options, as described in Table 4.2. The distributed anchor point is the simplest option, but it has some drawbacks. The most crucial is that all the user traffic is routed to the edge DN. This is especially apparent when looking at the example of a Content Delivery Network (CDN). One can imagine a scenario where, e.g., Netflix uses edge computing in 5G to pre-fetch and buffer their video streams. This means, that Netflix has to deploy a proxy at the edge locations. Solving this using distributed anchor point would mean that all the UE's traffic is routed to Netflix's edge DN. This is an issue in privacy as well as in performance. The same applies to a certain extent to the session breakout mode, using the UL CL. While the UL CL is able to route only Netflix's traffic to their DN, the UL CL itself receives all the traffic. Thus, it should not be operated by Netflix, but by a trusted entity, preferably the MNO itself. However, the latency requirements mandate that the UL CL is as close as possible to the UPF in the edge DN. One can see that this can create substantial operational difficulties. The option to use multiple PDU sessions is most probably the most straight-forward, as a similar approach is already used for VoNR. However, it requires the UE to classify the traffic. It must possess rules when to use the Netflix PDU session and when to use the common internet PDU session. While this is hard-wired by the device manufacturers for voice, this may become a challenge for application developers. Android already offers a way for operators to configure network slicing parameters and traffic routing, as described by Android[W38]. A similar approach for more fine-grained edge computing rules may be a promising solution. Nevertheless, another potential drawback is that each PDU session increases the control plane traffic. A solution for the privacy issue for the distributed anchor point and the session breakout option is to have a trusted entity such as a cloud or edge computing provider operate the edge data center. In this case, the edge provider operates the edge UPF and uses IP routing within the edge data center to route the Netflix traffic to the Netflix proxies and forward the other traffic to the internet. However, this requires much trust in the edge provider and it is also questionable whether MNOs are willing or legally able to give away the control of the user plane. As one can see, there are still many open questions which have to be answered when applying the concepts presented in this thesis in a productive system.

The author has contributed to several important parts of the OAI code base, especially in creating the PCF and adapting the SMF. However, the performance issue on the VPP-UPF shows that the solution requires additional engineering effort to achieve the desired stability. The approach chosen in the SMF to use a graph of UPF associations might have been too generic for the use case of a UL CL. Other implementation methods could have been to fixate the scenario to one or more specific UPF deployment options. Nevertheless, the current solution in the SMF provides a flexibility and is thus more future proof. As the graph is already implemented, other graph search algorithms such as the Dijkstra algorithm or A* may be integrated to find the shortest path in the graph.

The work done on the PCF is a solid foundation for future improvements. The lack of an API between the PCF and the AF breaks the automation of an edge computing scenario and should be added to the PCF, so that the OAI CN is able to handle standardized edge computing enabler layer implementations.

The implementation described in Section 5.3 is not trivial. While the creation of the UPF graph is fairly straight-forward, selecting the subgraph based on the DNAIs was a complex task to solve. The main reason is that during development, the author has identified several situations where the data structures given by the specifications did not fit well to the task at hand. This required to implement additional abstraction layers. This is especially true for the `PolicyDecision` model class. As discussed, it contains many fields which are not relevant and the relevant fields are hidden in the object's hierarchy. During the implementation it becomes apparent that the API has not been designed with this specific task in mind. Sometimes there are also inconsistencies in the naming. For example, an important field for the presented solution is the `dnaiNwInstanceList` from the UPF information in the NRF. However, this field is not a list, but a map. While this is just a small inconvenience, it exemplifies that the 3GPP did not make it easy for software developers to follow their specifications.

The PFCP procedures and call flows have to follow a strict pre-defined order. There is one reason why this is the case: The PFCP standard, described in 3GPP TS 29.244[100], specifies that the F-TEID is generated by the UPF. While there is no reasoning present for this decision, it may be to simplify the F-TEID handling on the SMF. However, this prevents a more elegant solution for creating PFCP sessions with multiple UPFs. If the SMF would be allowed to generate the F-TEID, it could send the PFCP messages in any order. Even more important, the procedures could be executed in parallel. While it is not possible to change this for 5G, it may be used as a consideration for the upcoming specification of 6G. Another example of – in the author's opinion – unnecessary complexity is the presence of the NGAP protocol in the SMF. While it is reasonable that the UE uses a binary protocol for communication with the 5GC, there is no reason why this protocol should not terminate on the AMF, the first point of contact for the UE. In its current state, the AMF forwards the content of the NGAP protocol over HTTP to the SMF. There is no reason why the AMF cannot act as a gateway and translate the required fields from NGAP to JSON and vice versa. Even more astounding, the AMF communicates with the SMF using JSON as well as NGAP. Further, in the author's opinion the UL and DL procedures could be combined. The CN could receive a preliminary F-TEID from the gNB in the PDU session request, which is used to create the UL and DL GTP tunnels in the same PFCP message. It has to be analyzed whether this approach would contradict any requirements in the RAN or the UE. Again, this may be considered for the 6G specification.

The work described in Chapter 5 has also shown how important open source reference implementations of 5G networks are. The OAI – but also other open source networking equipment – gives researchers the possibility to conduct their experiments with a low entry barrier. Also, and this is arguably more important, the software itself can be

adapted to one's use without requiring the (commercial) support of a third party. Given the complexities of implementing a standard-compliant 5GC, it can also be used to explore different, more efficient and more innovate, solutions. However, as a reference implementation, an open source 5GC must also be compliant with the 3GPP standards.

As described in Chapter 7, the robotics scenario does successfully execute on the edge of the 5G network, but the quality and functionality of the produced output is affected. Therefore, the answer to *RQ3* has to be that – at least currently – offloading a SLAM algorithm in the edge of a 5G network can impact the usability of the solution. In the end, it has to be considered by robotic application developers whether a deployment where the median pose error is 7 mm worse is feasible. Given that current robot localization software is often inaccurate, this may be acceptable.

The results presented in Section 7.4 show that the Google cartographer is relatively robust w.r.t. latency and jitter. While it is apparent that the SLAM node performed worse in the 5G edge deployment than in the WiFi deployment, the difference is less then expected. As reported in Table 7.3, the mean latency is more than six times higher in the 5G edge deployment. Then, the median absolute pose error is 63% higher, approximately 10% of the latency increase.

The results presented in Section 7.4 show that there is still room for improvement in the OAI. While it may be unrealistic to expect an open source reference implementation to perform as well as a commercial solution, the latency measurements are higher than expected. The end-to-end latency also includes the WiFI connection between the `simulation_computer` and the `nat_host`, which adds at least 0.7 ms and up to 4.2 ms latency. While the results are not yet published, the engineers at the OAI RAN project are working on reducing the end-to-end latency and have already discovered that the UE as well as the SPGWU contribute to the jitter and the high latency. It is expected that the situation improves in future versions of the OAI. Furthermore, there is engineering effort necessary to make the Quectel RM500Q reliably compatible with Ubuntu 22.04. In the author's estimation, these efforts promise to reduce the latency significantly and a t_{mean} of 6 ms with a σ_t below 1 ms is realistic. When these performance milestones have been reached, the conducted experiment should be repeated to compare the absolute pose error. Nevertheless, there is still a lot of work necessary to reach the desired latency of 1 ms.

The solution used in the experiment does not use a real-life edge computing scenario. As the core network is deployed in the same machine as the SLAM node, the edge enabler layer is not used at all. Further improvements and implementations in the OAI are necessary to repeat the experiment in a more realistic edge computing scenario. However, this basic deployment can still be taken as a reference for private 5G networks. A private 5G network does neither need a UL CL, nor the AF influence on traffic routing, as most likely all the CN components are already deployed at the edge. Nevertheless, the OAI should cover private 5G as well as edge computing in a public 5G network.

The accuracy of the estimated pose after the turn of the robot is too bad for a real-

life robotics use case. However, this does not relate to the edge deployment, as this issue also persists in the local deployment. One of the reasons might be that ROS2 is used. Although ROS2 offers great improvements compared to its previous versions, not all ROS2 packages have yet reached the maturity of ROS packages. One example is `cartographer-ros`, as it is not officially supporting ROS2 distributions. Another is the new navigation stack. In the binary packages for the humble release is a bug at the time of writing of this thesis. The costmaps are not published on the topics, which leads to collisions. The author chose ROS2 instead of ROS because of the new approach to networking. ROS uses random ports to communicate between nodes, making the deployment in Docker or over firewalls difficult. DDS uses a dynamic, but predictable port range. This eases the deployment over network boundaries. However, the multicast mechanism does not always work reliable. The mitigation for this issue, the discovery server, does not work together with NAT, as the IP address is written in the application layer and thus not exchanged by a NAT gateway. The deployment of ROS2 nodes over network boundaries remains a significant challenge and the easiest solution may be to use an overlay VPN network. While this solves the issue of node communication, a VPN can introduce additional latencies, and a decentralized solution should be chosen.

The presented 5G approach has the advantage that it can support ROS2 networking without using an overlay VPN network. While many MNOs use NAT at the border of their core network (i.e., at the UPF), this is not mandatory. 5G uses GTP tunnels to route traffic to and from the UE, hence the UPF is able to route the UE IP address. If the edge DN configures an IP route to forward the UE IP address to the UPF, direct communication is possible. In the other direction, all the traffic is routed to the UPF in any case. It is also possible that the servers in the edge DN use the same IP subnet as the UEs. However, the SMF has to be configured accordingly to not assign UE IP addresses which are used in the edge DN to avoid conflicts. An option is that the SMF does not assign the addresses itself, but uses an external Dynamic Host Configuration Protocol (DHCP) server. While this is not standardized, one could imagine an SMF feature which uses different DHCP servers for different network slices. While this example is inspired by the ROS2 communication challenges, it inherently describes a VPN. If the UPF is deployed in the network of a company and the edge DN is the company-internal network, the 5G UEs can easily be integrated into the internal network. In this example, the company is using a public 5G network, but the IP packets between the 5G UEs and other hosts never leaves the company network. While the robotics use case is definitely a very interesting and important driver of 5G networks, this example shows that the flexibility of 5G can enable many more use cases.

CHAPTER 9

Conclusion and Future Work

This thesis discusses the topics of edge computing in 5G and how it relates to the fields of robotics and Industry 4.0. Thus, the author formulates three research questions in Section 1.3:

- **Research Question 1 (RQ1)**: What is a suitable edge computing architecture for offloading latency-critical applications to the edge of a 5G network?
- **Research Question 2 (RQ2)**: What is a feasible approach to implement a MEC prototype and integrate it into an existing 5G system such as the OAI?
- **Research Question 3 (RQ3)**: Given a MEC prototype and a ROS2 SLAM node, how does offloading the node to the edge of a 5G network affect the latency and the quality and functionality of the produced output?

Before *RQ1* can be answered, the broader concepts of 5G, edge computing and robotics have to be defined. The author has provided this in Chapter 2. While this is sufficient for the presented topic of edge computing in 5G, much more can be written about 5G and edge computing. However, there is enough literature available with the sole purpose of explaining these concepts. As this work relies heavily on these, the foundations chapter serves as an introduction to the reader.

The answer to *RQ1* is presented in Chapter 4. It defines the requirements for edge computing in 5G on the RAN, but especially on the 5GC. The author describes the two standardized edge computing frameworks, ETSI MEC and 3GPP EDGEAPP. While ETSI MEC is the older, more mature, standard, its adoption is not great in current 5G networks. The author argues that this is mainly due to the fact that the interworking with the underlying mobile network is not specified, resulting in different, incompatible implementations. 3GPP EDGEAPP tries to bridge the gap between the orchestration

and management aspects of ETSI MEC and the internal procedures of the 5GC. However, as the standard is very novel, reliable prototypes and experiments are not yet available. Therefore, the author describes a solution which incorporates the mature ETSI MEC architecture within the tightly-coupled 3GPP EDGEAPP architecture, which is able to utilize the advantages of both approaches. Furthermore, in Chapter 3, non-standardized approaches and prototypes of edge computing in 5G NSA are described.

Additionally, the possibilities how an external edge enabler layer such as the 3GPP EDGEAPP architecture can influence the 5GC are discussed. Hereby, the author focuses on a standard-compliant solution, which should prevent the aforementioned incompatibilities. Therefore, the EES from 3GPP EDGEAPP or the MEP from ETSI MEC are acting as a 5G AF. The APIs how the AF can influence the UPF selection and traffic steering are discussed in great details. In situations where the standards does not specify the details, the author has proposed different solution options. These options are devised in a manner as to not contradict or violate the specifications, while having an added value. An example thereof are the different UPF selection options in the SMF. Each of this option utilizes the available standardized APIs, but their possible implementations have different prerequisites and require different 5GC deployments. The comparison of these proposed options also considers impacts on a cloud-native 5GC.

Altogether, the author provides two different suitable edge computing architectures and describes different options how these may be integrated into the 5GC. Each approach has its own field of use. The author chose one of the presented solutions to answer *RQ2*.

The findings in Chapter 4 also reveal much potential for future work. Foremost, the need for a prototype implementation of the 3GPP EDGEAPP architecture becomes apparent. Further, the analysis of the standardized APIs and specifications has shown that there are many different approaches. While this work explores and compares these theoretically, there is the potential to perform experiments on each option together with an edge enabler layer to, e.g., assess which approach is best suited in combination with an NFV MANO.

RQ2 is explored in Chapter 5. It describes a concrete implementation of the procedures discussed while answering *RQ1*. The author contributed to the OAI 5GC to support the edge computing use case. There are significant changes in the OAI SMF. Previously to these, the SMF supported only one UPF per PDU session and the UPF selection was based on the network slicing information and the DNN. The author replaced the list of available UPFs with a graph, which embeds the inherit hierarchy of different UPF deployments. Additionally, the flexibility of the graph allows to support many cases of deployment options. This is reflected in the novel UPF selection method. It allows to select a UPF based on information received by the PCF, an essential feature for edge computing. Further, the author has proven that the time complexity of the UPF graph creation and selection is linear w.r.t. the amount of UPFs and thus scales well in a large network.

In addition to the changes in the SMF, the PCF has been created. While it does not

support the interface to the AF, it supports influencing the UPF selection and traffic steering on the SMF. The software architecture of the PCF is designed with future use cases in mind, especially adding support of the AF interface. Other important features of the policy frameworks such as QoS are considered in the design. Furthermore, the author contributed to the VPP-UPF to support the different UPF deployment configurations and to the NRF.

This thesis shows that standard-compliant edge computing can be implemented in an open source 5G CN. The additions to the SMF code base prove that graph data structures and algorithms are a good design pattern for the edge computing requirements in the SMF. However, the lack of the AF interface on the PCF prevents an end-to-end use case. The dynamic provisioning of the policies using the file system is not sufficient for a realistic edge computing solution. The implementation lays the foundation for such a use case, but it is not explored further herein. Thus, a potential for future work is to take a 3GPP EDGEAPP prototype and integrate it with the OAI 5GC. This would allow to showcase the ubiquity of edge computing and show that the presented work has been designed with this in mind.

While the answers to *RQ1* and *RQ2* give valuable insights to 5G and edge computing, one may ask what the real-world implications are. As described in Chapter 1, the robotics use case is an important driver for 5G edge computing. Also, the increased automation of factories poses new challenges in the fields of robotics. The experiment described in Chapter 6 to answer *RQ3* shows that 5G edge computing is a viable solution to some of these challenges.

The conducted experiment showed that offloading a ROS2 SLAM node to the 5G edge is possible. Although the reported results from Chapter 7 show that the quality of the localization is negatively affected, the functionality itself was not impacted. Nevertheless, the underlying hypothesis of this work is that offloading the SLAM algorithm to the edge does not negatively impact the quality w.r.t. a WiFi deployment. Therefore, this hypothesis is refuted. Whether the reported deterioration of the accuracy is acceptable depends on the use case.

As described in Chapter 8, there are ongoing efforts to reduce the latency and the jitter in the OAI. Thus, the experiment presented in this thesis should be reproduced after this improvements will be implemented in the OAI. The author expects that the SLAM accuracy will be closer or even equal to the WiFi deployment in an improved version of the OAI.

Further, the results show that the used Google cartographer SLAM node performed well given the unexpected high latency and jitter. While SLAM and robotics is often classified as latency-sensitive applications in research, the author was unable to find reliable studies examining the real limits. Also, to the author's best knowledge, there is no study which explores how the pose error correlates with the latency. A potential future work is to conduct such an experiment with popular ROS2 SLAM nodes. The findings given by such a study could greatly benefit the understanding of the latency

requirements of different SLAM nodes.

ROS2 is inherently a distributed system and the offloading of the SLAM algorithm served as a latency-critical example. Thus, there are many possibilities for offloading other parts of the robot, such as the navigation stack, path planning and pose estimation. Also, a 3D-SLAM instead of a 2D-SLAM may be offloaded to the edge and compared based on its accuracy. After the underlying 5G implementation has reached a certain maturity, components with even higher latency requirements such as steering and obstacle avoidance may be offloaded to the edge as well. The presented findings in this thesis provide a guideline how this potential future work can be conducted.

Ultimately, the author provides an analysis how edge computing can be enabled in a 5G network and also a prototype implementation thereof. The conducted experiment shows that is possible to move a latency-critical ROS2 node to the 5G edge. Thus, this work lays the foundation for further research in 5G-enabled open source mobile robotics.

APPENDIX A

API Details

This appendix contains details and examples about the APIs described in this thesis. When URIs are used in this context, the host name is always the name of the network function (e.g., "pcf" for a PCF).

A.1 PCF SM Policy Control API

This chapter provides examples for the PCF SM Policy Control API.

A.1.1 Example Body of a SM Policy Association CREATE Request

Listing A.1 shows the JSON body of a default SM Policy Control CREATE Request. The corresponding URI of the request is: `http://pcf:port/npcf-smpolicycontrol/v1/sm-policies/` and the method is `POST`.

```
{
    "supi": "imsi-208950000000032",
    "pduSessionType": "IPV4",
    "pduSessionId": 42,
    "dnn": "default",
    "notificationUri": "http://smf:port/12345",
    "sliceInfo": {
        "sst": 222,
        "sd": "123"
    }
}
```

Listing A.1: Example body of SMPolicyControl CREATE request

The request in Listing A.1 does only contain the mandatory fields, hence the PCF can only decide based on these. This means that either a policy for SUPI, DNN or slice should be available on the PCF.

A.1.2 Example Body of a SM Policy Association CREATE Response

Listing A.2 shows an example of the body of a response to the SMF when the SM Policy creation was successful. The PCF creates a unique ID for each SM policy association. This ID is not communicated in the body, but is located in the Location header of the response: `http://pcf:port/npcf-smpolicycontrol/v1/sm-policies/<id>`. This URL is used to query, update and delete policy associations.

The response contains two different PCC rules. The rule "supi-rule-edge" is used for edge computing, whereas the "supi-rule-internet" is used for internet traffic. This example shows that it is possible to have different routes for the same subscriber and even the PDU session. This may be implemented on the SMF by enforcing a UPF to be a UL CL. The flow description of the edge rule targets one specific IP address (8.8.8.8) and refers to the traffic rule "edge-traffic". The precedence 9 indicates that it has a higher precedence than the other rule. Therefore, the internet rule is applied whenever the edge rule does not match. Its flow description matches any IP traffic.

There are two traffic control descriptions: "default-traffic" and "edge-traffic". In this example, each has its dedicated DNAIs and an associated routing profile ID, which is enforced by the SMF.

```
1  {
2    "pccRules": {
3      "supi-rule-edge": {
4        "flowInfos": [
5          {i
6            "flowDescription": "permit out ip from 8.8.8.8 to
                 assigned"
7          }
8        ],
9        "pccRuleId": "supi-rule-edge",
10       "precedence": 9,
11       "refTcData": [
12         "edge-traffic"
13       ]
14     },
15     "supi-rule-internet": {
16       "flowInfos": [
```

```
        {
          "flowDescription": "permit out ip from any to
             assigned"
        }
      ],
      "pccRuleId": "supi-rule-internet",
      "precedence": 10,
      "refTcData": [
        "default-traffic"
      ]
    }
  },
  "traffContDecs": {
    "default-traffic": {
      "routeToLocs": [
        {
          "dnai": "internet-dn",
          "routeProfId": "route-internet"
        },
        {
          "dnai": "ulcl",
          "routeProfId": "route-internet"
        },
        {
          "dnai": "aupf1",
          "routeProfId": "route-internet"
        },
        {
          "dnai": "access",
          "routeProfId": "route-internet"
        }
      ],
      "tcId": "default-traffic"
    },
    "edge-traffic": {
      "routeToLocs": [
        {
          "dnai": "edge-dn",
          "routeProfId": "route-edge"
        },
        {
          "dnai": "ulcl",
          "routeProfId": "route-edge"
```

```
        },
        {
          "dnai": "aupf2",
          "routeProfId": "route-edge"
        },
        {
          "dnai": "access",
          "routeProfId": "route-edge"
        }
      ],
      "tcId": "edge-traffic"
    }
  }
}
```

Listing A.2: Example body of SMPolicyControl CREATE response

A.2 NRF NF Discovery Service

This chapter provides an example of the UPF information stored by and exchanged through the NRF used in this thesis.

A.2.1 Example of UPF Info in the NRF

The example described in Listing A.3 show how the mapping between DNAI from an AF or PCF and network instance locally configured on the UPF can be achieved through the NRF UPF info.

```
{
  "sNssaiUpfInfoList": [
    {
      "sNssai": {
        "sst": 222,
        "sd": "123"
      },
      "dnnUpfInfoList": [
        {
          "dnn": "default",
          "dnaiList": [
            "edge_dn_1"
          ],
          "dnaiNwInstanceList": {
            "edge_dn_1": "DN_1_N9"
          }
```

```
17          }
18        ]
19      }
20    ]
21  }
```

Listing A.3: UPF info example

A.3 UPF Configuration

The Listing A.4 from this chapter shows the VPP-UPF configuration that is used to configure the UPF as a UL CL with an `N3`, an `N4` and two `N9` interfaces.

```
1  {
2  ip table add 1
3  ip table add 2
4  ip table add 3
5
6  create host-interface name n9-1
7  set interface mtu 1500 host-n9-1
8  set interface ip table host-n9-1 3
9  set interface ip address host-n9-1 @N9_1_IPV4_ADDRESS_LOCAL@/24
10 set interface state host-n9-1 up
11
12 create host-interface name n9-2
13 set interface mtu 1500 host-n9-2
14 set interface ip table host-n9-2 1
15 set interface ip address host-n9-2 @N9_2_IPV4_ADDRESS_LOCAL@/24
16 set interface state host-n9-2 up
17
18 create host-interface name n4
19 set interface mtu 1500 host-n4
20 set interface ip table host-n4 0
21 set interface ip address host-n4 @N4_IPV4_ADDRESS_LOCAL@/24
22 set interface state host-n4 up
23
24 create host-interface name n3
25 set interface mtu 1500 host-n3
26 set interface ip table host-n3 2
27 set interface ip address host-n3 @N3_IPV4_ADDRESS_LOCAL@/24
28 set interface state host-n3 up
29
30 ip route add 0.0.0.0/0 table 2 via @N3_IPV4_ADDRESS_REMOTE@
      host-n3
```

```
ip route add 0.0.0.0/0 table 0 via @N4_IPV4_ADDRESS_REMOTE@
   host-n4
ip route add 0.0.0.0/0 table 1 via @N9_2_IPV4_ADDRESS_REMOTE@
   host-n9-2
ip route add 0.0.0.0/0 table 3 via @N9_1_IPV4_ADDRESS_REMOTE@
   host-n9-1

upf pfcp endpoint ip @N4_IPV4_ADDRESS_LOCAL@ vrf 0
upf node-id fqdn gw@GW_ID@.vppupf.node.5gcn.mnc@MNC03@.mcc@MCC@
   .@REALM@

upf nwi name @NWI_N3@ vrf 2
upf nwi name @NWI_N9_1@ vrf 3
upf nwi name @NWI_N9_2@ vrf 1

upf specification release 16

upf gtpu endpoint ip @N3_IPV4_ADDRESS_LOCAL@ nwi @NWI_N3@ teid
   0x000004d2/2
upf gtpu endpoint ip @N9_1_IPV4_ADDRESS_LOCAL@ nwi @NWI_N9_1@
   teid 0x000004d2/1
upf gtpu endpoint ip @N9_2_IPV4_ADDRESS_LOCAL@ nwi @NWI_N9_2@
   teid 0x000004d2/3
}
```

Listing A.4: VPP-UPF configuration for a UL CL scenario

APPENDIX B

Prototype Configuration Details

This appendix contains examples on how to configure the edge computing prototype described in Chapter 5.

B.1 PCF Policy Provisioning

As described in Section 5.2, each of the three policy control types needs to be stored in a separate directory. These directories are configured in the `pcf.conf` file as follows:

```
1 {
2 PCC_RULES_DIRECTORY ="/openair-pcf/policies/pcc_rules";
3 TRAFFIC_RULES_DIRECTORY ="/openair-pcf/policies/traffic_rules";
4 POLICY_DECISIONS_DIRECTORY ="/openair-pcf/policies/
     policy_decisions";
5 }
```

Listing B.1: Example of PCF policy directories configuration

When the PCF is started using `docker-compose`, there are four environment variables to set:

- `POLICY_BASE_DIR`: e.g., `/openair-pcf/policies/`
- `PCC_RULES_DIR`: e.g., `pcc_rules`
- `TRAFFIC_RULES_DIR`: e.g., `traffic_rules`
- `POLICY_DECISIONS_DIR`: e.g., `policy_decisions`

These environment variables create the same configuration as outlined in Listing B.1.

In the `TRAFFIC_RULES_DIRECTORY`, one or more files of the type `TrafficControlData` can be stored. Listing B.2 shows an example:

```
internet-scenario1:
  routeToLocs:
  - dnai: access
  - dnai: iupf1
  - dnai: aupf1
  - dnai: internet

edge-scenario1:
  routeToLocs:
  - dnai: access
  - dnai: aupf2
  - dnai: edge
```

Listing B.2: Example of PCF traffic rules

The `PCC_RULES_DIRECTORY` must contain files of the type `PccRule`. Listing B.3 shows an example:

```
internet-rule:
  flowInfos:
  - flowDescription: permit out ip from any to assigned
  precedence: 10
  refTcData:
  - internet-scenario1

edge-rule:
  flowInfos:
  - flowDescription: permit out ip from 8.8.8.8 to assigned
  precedence: 9
  refTcData:
  - edge-scenario1
```

Listing B.3: Example of PCF PCC rules

The policy decisions follow the syntax as described in Section 5.2. Listing B.4 shows an example:

```
decision_supi:
  supi_imsi: "208950000000031"
  pcc_rules:
  - internet-rule
  - edge-rule
```

```
decision_dnn:
  dnn: default
  pcc_rules:
  - internet-rule

decision_default:
  default: true
  pcc_rules:
  - internet-rule

decision_slice:
  slice:
    sst: 222
    sd: "123"
  pcc_rules:
  - internet-rule
```

Listing B.4: Example of PCF policy decisions

B.2 UL CL Tutorial

A tutorial describing how the UL CL solution presented in Chapter 5 can be used in the OAI is available on the OAI GitLab repository located at:

https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed

It contains a tutorial and a `docker-compose` file for the UL CL scenario.

APPENDIX C

Simulation and Experiment Details

This appendix contains details on how the robotic simulation and the experiment is set up and executed.

C.1 Dockerfile for the SLAM node

The SLAM node Docker image uses the base image `ros:foxy-ros-base-focal`. Then, the package `ros-foxy-cartographer-ros` is installed and the local ROS workspace is copied into the image and built. The local workspace contains the launch files for the SLAM node as well as additional configuration such as the `lua` file used by the cartographer. The image used for the experiment is available on the Docker Hub with the tag `stespe/ros-cartographer-node:latest`.

It is worth to note that the simulation uses the ROS2 humble distribution, as the local operating system is Ubuntu 22.04 and the ROS2 foxy distribution does not support this Ubuntu release.

The Dockerfile for the image is shown in Listing C.1.

```
FROM ros:foxy-ros-base-focal
ARG DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install -q -y \
    --no-install-recommends ros-foxy-cartographer-ros
COPY ./edge_ws /workspace
WORKDIR /workspace
RUN . "/opt/ros/$ROS_DISTRO/setup.sh" && colcon build
COPY ./ros_entrypoint.sh /ros_entrypoint.sh
ENTRYPOINT [ "/ros_entrypoint.sh" ]
```

Listing C.1: Dockerfile for the SLAM node

The entrypoint is the generic ROS2 entrypoint. Thus, to start the SLAM node, the command `ros launch edge_slam_node slam.launch.py` needs to be executed. Also, the environment variable `ROS_DISCOVERY_SERVER` has to be set to the IP address and port of the Fast DDS discovery server. To ease the deployment, a `docker-compose` file is used, as described in Section C.2.

C.2 Docker-compose File Used in the Experiment

Listing C.2 describes the `docker-compose` file used to deploy the SLAM and Fast DDS discovery server nodes.

```
version: '3.8'
services:
  ros-slam-node:
    container_name: ros-slam-node
    image: stespe/ros-cartographer-node:latest
    environment:
      - ROS_DISCOVERY_SERVER=192.168.130.4:11811
    command: ["ros2","launch","edge_slam_node","slam.launch.py"]
    networks:
      ros-network:
        ipv4_address: 192.168.130.2

  fastdds-discovery:
    container_name: fastdds-discovery
    image: stespe/ros-cartographer-node:latest
    command: ["fastdds", "discovery", "-i", "0"]
    ports:
      - "11811:11811/udp"
    networks:
      ros-network:
        ipv4_address: 192.168.130.4

networks:
  ros-network:
    name: ros-network
    ipam:
      config:
        - subnet: 192.168.130.0/24
```

Listing C.2: `docker-compose` file to deploy the SLAM and Fast DDS nodes

C.3 Comparing the Absolute Pose Error using Evo

The `evo` tool supports reading files in the `rosbag` format. As the experiment is running on ROS2, the format of the ROS bags changed. To prepare the input for `evo`, first the topics of interest need to be extracted from the ROS2 bag. Then, the ROS2 bag is converted into a ROS bag format. The `ros2 bag convert` tool is used to extract the pose topics. The `convert` tool is configured using a `yaml` file. This configuration is shown in Listing C.3.

```
output_bags:
- uri: pose_bags  # required
  storage_id: sqlite3  # required
  max_bagfile_size: 0
  max_bagfile_duration: 0
  storage_preset_profile: ""
  storage_config_uri: ""
  all: false
  topics: ["/ground_truth/pose", "/real_pose/pose"]
```

Listing C.3: YAML configuration to extract the poses from a ROS2 bag file

The tool to convert from ROS2 bags to ROS bags is `rosbags-convert`. Then, the `evo` tool is used to generate the plots based on the converted bag, as shown in Listing C.4

```
ros2 bag convert -i <ros2bag> -o convert.yaml
rosbags-convert pose_bags

evo_ape bag pose_bags.bag -va --plot /ground_truth/pose \
              /real_pose/pose --save_results result.zip
```

Listing C.4: Converting the ROS bags and plotting the absolute pose error

List of Figures

List of Tables

List of Algorithms

List of Listings

Acronyms

3GPP 3rd Generation Partnership Project.

5GC 5G Core.

5GS 5G System.

A-UPF Anchor UPF.

AC Application Client.

AF Application Function.

AMF Access and Mobility Management Function.

AUSF Authentication Server Function.

BP Branching Point.

CDN Content Delivery Network.

COTS Common-Off-The-Shelf.

CU Centralized Unit.

CUPS Control and User Plane Separation.

DDS Data Distribution Service.

DHCP Dynamic Host Configuration Protocol.

DL Downlink.

DN Data Network.

DNAI Data Network Access Identifier.

DNN Data Network Name.

DTO Data Transfer Object.

DU Distributed Unit.

EAS Edge Application Server.

ECS Edge Configuration Server.

EEC Edge Enabler Client.

EES Edge Enabler Server.

eMBB Enhanced Mobile Broadband.

eNB eNodeB.

EPC Evolved Packet Core.

ETSI European Telecommunications Standardization Institute.

FAR Forwarding Action Rule.

F-TEID Fully Qualified Tunnel Endpoint Identifier.

FAPI Femto Application Programming Interface.

gNB gNodeB.

GPRS General Packet Radio Service.

GTP GPRS Tunneling Protocol.

HARQ Hybrid Automatic Repeat Request.

HSS Home Subscriber Server.

I-UPF Intermediate UPF.

IMSI International Mobile Subscriber Identity.

IMU Inertial Measurement Unit.

IoT Internet of Things.

ITU International Telecommunication Union.

LIDAR Light Detection and Ranging.

LTE Long Term Evolution.

MAC Medium Access Control.

MANO Management and Orchestration.

MCC Mobile Country Code.

MEC Multi-Access Edge Computing.

MEO MEC Orchestrator.

MEP MEC Platform.

MIMO Multiple Input Multiple Output.

MME Mobility Management Entity.

mMTC Massive Machine-type Communications.

MNC Mobile Network Code.

MNO Mobile Network Operator.

NAS Non-Access Stratum.

NEF Network Exposure Function.

NF Network Function.

NFV Network Function Virtualization.

NFVI Network Functions Virtualization Infrastructure.

NGAP NG Application Protocol.

NR New Radio.

NRF Network Repository Function.

NSA Non Stand-Alone.

NSSF Network Slice Selection Function.

NWI Network Instance.

OAI OpenAirInterface.

OFDM Orthogonal Frequency-Division Multiplexing.

OS Operating System.

OSS Operations Support System.

OTT Over-the-Top.

OVS Open vSwitch.

PBCH Physical Broadcast Channel.

PCC Policy and Charging Control.

PCF Policy Control Function.

PCRF Policy and Charging Rules Function.

PDCCH Physical Downlink Control Channel.

PDCP Packet Data Convergence Protocol.

PDR Packet Detection Rule.

PDSCH Physical Downlink Shared Channel.

PDU Protocol Data Unit.

PFCP Packet Forwarding Control Protocol.

PGW Packet Data Network Gateway.

PLMN Public Land Mobile Network.

PRACH Physical Random Access Channel.

PSA PDU Session Anchor.

PUCCH Physical Uplink Control Channel.

PUSCH Physical Uplink Shared Channel.

QoS Quality of Service.

RAN Radio Access Network.

RAT Radio Access Type.

RLC Radio Link Control.

RNIS Radio Network Information Service.

ROS2 Robot Operating System 2.

RRC Radio Resource Control.

RTPS Real Time Publish Subscribe Protocol.

RU Radio Unit.

S1AP S1 Application Protocol.

SA Stand-Alone.

SBA Service Based Architecture.

SBI Service Based Interface.

SCC Session and Service Continuity.

SCS Sub-Carrier Spacing.

SDF Simulation Description Format.

SDN Software Defined Networking.

SDR Software Defined Radio.

SGW Serving Gateway.

SIM Subscriber Identity Module.

SLAM Simultaneous Localization and Mapping.

SMF Session Management Function.

SUPI Subscription Permanent Identifier.

TAC Tracking Area Code.

TSN Time-Sensitive Networking.

UDM Unified Data Management.

UDR Unified Data Repository.

UDSF Unstructured Data Storage Function.

UE User Equipment.

UL Uplink.

UL CL Uplink Classifier.

UPF User Plane Function.

URLLC Ultra-Reliable Low Latency Communications.

USRP Universal Software Radio Peripheral.

VIM Virtualized Infrastructure Manager.

VM Virtual Machine.

VNF Virtualized Network Function.

VoNR Voice over NR.

VPP Vector Packet Processing.

Bibliography

[1] Quoc-Viet Pham et al. „A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art“. In: *IEEE Access* 8 (2020), pp. 116974–117017. DOI: 10.1109/ACCESS.2020.3001277.

[2] Najmul Hassan, Kok-Lim Alvin Yau, and Celimuge Wu. „Edge Computing in 5G: A Review“. In: *IEEE Access* 7 (2019), pp. 127276–127289. DOI: 10.1109/ACCESS.2019.2938534.

[3] Weisong Shi and Schahram Dustdar. „The Promise of Edge Computing“. In: *Computer* 49.5 (2016), pp. 78–81. DOI: 10.1109/MC.2016.145.

[4] European Telecommunications Standards Institute (ETSI). *MEC Framework and Reference Architecture*. Group Specification (GS) MEC 003. Version V3.1.1. 2022.

[5] 3GPP. *Architecture for enabling Edge Applications*. Technical Specification (TS) 23.558. Version 17.3.0. 3rd Generation Partnership Project (3GPP), 2022.

[6] Tariq Masood and Paul Sonntag. „Industry 4.0: Adoption challenges and benefits for SMEs“. In: *Computers in Industry* 121 (2020), p. 103261. DOI: 10.1016/j.compind.2020.103261.

[7] Jin Yang. „5G Network for a Variety of Vertical Services“. In: *5G Verticals: Customizing Applications, Technologies and Deployment Techniques*. 2020, pp. 25–56. DOI: 10.1002/9781119514848.ch2.

[8] Muhammad Farooq and Valliappan Raju. „Impact of Over-the-Top (OTT) Services on the Telecom Companies in the Era of Transformative Marketing“. In: *Global Journal of Flexible Systems Management* 20.2 (2019), pp. 177–188. DOI: 10.1007/s40171-019-00209-6.

[9] Florian Kaltenberger et al. „OpenAirInterface: Democratizing innovation in the 5G Era“. In: *Computer Networks* 176 (2020), p. 107284. DOI: 10.1016/j.comnet.2020.107284.

[10] Sriganesh K. Rao and Ramjee Prasad. „Impact of 5G Technologies on Industry 4.0“. In: *Wireless Personal Communications* 100.1 (2018), pp. 145–159. DOI: 10.1007/s11277-018-5615-7.

[11] Christopher Cox. „Introduction“. In: *An Introduction to 5G*. Ed. by Christopher Cox. John Wiley & Sons, Ltd, 2021. Chap. 1, pp. 1–27. ISBN: 9781119602682. DOI: 10.1002/9781119602682.ch1.

[12] Stefan Rommer et al. „Introduction“. In: *5G Core Networks*. Ed. by Stefan Rommer et al. Academic Press, 2020. Chap. 1, pp. 1–5. ISBN: 9780081030097. DOI: 10.1016/B978-0-08-103009-7.00001-6.

[13] Peter F. Cowhey. „The international telecommunications regime: the political roots of regimes for high technology“. In: *International Organization* 44.2 (1990), pp. 169–199. DOI: 10.1017/S0020818300035244.

[14] ITU. *IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond*. ITU Recommendation (ITU-R) M.2083-0. International Telecommunications Union (ITU), 2015.

[15] I Chih-Lin et al. „New paradigm of 5G wireless internet“. In: *IEEE Journal on Selected Areas in Communications* 34.3 (2016), pp. 474–482. DOI: 10.1109/JSAC.2016.2525739.

[16] Christopher Cox. „Architecture of the Radio Access Network“. In: *An Introduction to 5G*. Ed. by Christopher Cox. John Wiley & Sons, Ltd, 2021. Chap. 3, pp. 55–71. ISBN: 9781119602682. DOI: 10.1002/9781119602682.ch3.

[17] Christopher Cox. „Architecture of the 5G New Radio“. In: *An Introduction to 5G*. Ed. by Christopher Cox. John Wiley & Sons, Ltd, 2021. Chap. 7, pp. 145–171. ISBN: 9781119602682. DOI: 10.1002/9781119602682.ch7.

[18] Tero Henttonen et al. „5G Radio Protocols“. In: *5G Technology: 3GPP New Radio*. Ed. by Harri Holma, Antti Toskala, and Takehiro Nakamura. John Wiley & Sons, Ltd, 2020. Chap. 7, pp. 149–186. ISBN: 9781119236306. DOI: 10.1002/9781119236306.ch7.

[19] Magnus Olsson et al. „EPC and 4G Packet Networks: Driving the Mobile Broadband Revolution“. In: *EPC and 4G Packet Networks*. Ed. by Magnus Olsson et al. Second Edition. Academic Press, 2013. Chap. 2, pp. 17–64. ISBN: 9780123945952. DOI: 10.1016/B978-0-12-394595-2.00002-5.

[20] Stefan Rommer et al. „Architecture overview“. In: *5G Core Networks*. Ed. by Stefan Rommer et al. Academic Press, 2020. Chap. 3, pp. 15–72. ISBN: 9780081030097. DOI: 10.1016/B978-0-08-103009-7.00003-X.

[21] Stefan Rommer et al. „Session management“. In: *5G Core Networks*. Ed. by Stefan Rommer et al. Academic Press, 2020. Chap. 6, pp. 111–136. ISBN: 9780081030097. DOI: 10.1016/B978-0-08-103009-7.00006-5.

[22] 3GPP. *System architecture for the 5G System (5GS)*. Technical Specification (TS) 23.501. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[23] Merlin Chlosta, David Rupprecht, and Thorsten Holz. „On the Challenges of Automata Reconstruction in LTE Networks“. In: *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. WiSec '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 164–174. ISBN: 9781450383493. DOI: 10.1145/3448300.3469133.

[24] 3GPP. *Non-Access-Stratum (NAS) protocol for 5G System (5GS)*. Technical Specification (TS) 24.501. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[25] 3GPP. *NG-RAN; NG Application Protocol (NGAP)*. Technical Specification (TS) 38.413. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[26] 3GPP. *General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)*. Technical Specification (TS) 29.281. Version 16.2.0. 3rd Generation Partnership Project (3GPP), 2021.

[27] 3GPP. *5G System; Session Management Services*. Technical Specification (TS) 29.502. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[28] 3GPP. *5G System; Access and Mobility Management Services*. Technical Specification (TS) 29.518. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[29] 3GPP. *5G System; Unified Data Repository Services*. Technical Specification (TS) 29.504. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[30] 3GPP. *5G System; Unified Data Management Services*. Technical Specification (TS) 29.503. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[31] 3GPP. *5G System; Authentication Server Services*. Technical Specification (TS) 29.509. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[32] Roger Piqueras Jover. „Security and protocol exploit analysis of the 5G specifications“. In: *IEEE Access* 7 (2019), pp. 24956–24963. DOI: 10.1109/ACCESS.2019.2899254.

[33] Li Sun and Qinghe Du. „Physical layer security with its applications in 5G networks: A review“. In: *China communications* 14.12 (2017), pp. 1–14. DOI: 10.1109/CC.2017.8246328.

[34] 3GPP. *Network function repository services*. Technical Specification (TS) 28.538. Version 16.8.0. 3rd Generation Partnership Project (3GPP), 2021.

[35] 3GPP. *5G; Unstructured data storage services*. Technical Specification (TS) 29.598. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2021.

[36] 3GPP. *Policy and charging control framework for the 5G System (5GS)*. Technical Specification (TS) 29.512. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[37] 3GPP. *Policy Authorization Service; Stage 3*. Technical Specification (TS) 29.514. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[38] 3GPP. *5G System; Network Slice Selection Services*. Technical Specification (TS) 29.531. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[39] 3GPP. *5G System; Network Exposure Function Northbound APIs*. Technical Specification (TS) 29.522. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[40] 3GPP. *5G System; Packet Flow Description Management Service*. Technical Specification (TS) 29.551. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[41] 3GPP. *5G System; Network Exposure (NE) function services for Non-IP Data Delivery (NIDD) and Short Message Services (SMS)*. Technical Specification (TS) 29.541. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[42] 3GPP. *5G System; Network Exposure Function Southbound Services*. Technical Specification (TS) 29.591. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[43] Tianzhu Zhang et al. „Comparing the performance of state-of-the-art software switches for NFV“. In: *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*. 2019, pp. 68–81. DOI: 10.1145/3359989.3365415.

[44] Christopher Cox. „Spectrum, Antennas and Propagation“. In: *An Introduction to 5G*. Ed. by Christopher Cox. John Wiley & Sons, Ltd, 2021. Chap. 4, pp. 73–89. ISBN: 9781119602682. DOI: 10.1002/9781119602682.ch4.

[45] 3GPP. *NR; NR and NG-RAN Overall description*. Technical Specification (TS) 38.300. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2021.

[46] Ali Zaidi et al. „NR Physical Layer: Overview“. In: *5G Physical Layer*. Ed. by Ali Zaidi et al. Academic Press, 2018. Chap. 2, pp. 21–34. ISBN: 9780128145784. DOI: 10.1016/B978-0-12-814578-4.00007-2.

[47] Anand R. Prasad et al. „3GPP 5G security“. In: *Journal of ICT Standardization* 6.1 (2018), pp. 137–158. DOI: 10.13052/jicts2245-800X.619.

[48] 3GPP. *Security architecture and procedures for 5G System*. Technical Specification (TS) 33.501. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[49] Shunliang Zhang. „An Overview of Network Slicing for 5G“. In: *IEEE Wireless Communications* 26.3 (2019), pp. 111–117. DOI: 10.1109/MWC.2019.1800234.

[50] Justus Rischke et al. „5G Campus Networks: A First Measurement Study“. In: *IEEE Access* 9 (2021), pp. 121786–121803. DOI: 10.1109/ACCESS.2021.3108423.

[51] Adlen Ksentini and Pantelis A. Frangoudis. „Toward Slicing-Enabled Multi-Access Edge Computing in 5G“. In: *IEEE Network* 34.2 (2020), pp. 99–105. DOI: 10.1109/MNET.001.1900261.

[52] Jitender Grover and Rama M. Garimella. „Optimization in Edge Computing and Small-Cell Networks“. In: *Edge Computing: From Hype to Reality*. Ed. by Fadi Al-Turjman. Springer International Publishing, 2019, pp. 17–31. ISBN: 9783319990613. DOI: 10.1007/978-3-319-99061-3_2.

[53] Mohammad R. Mesbahi, Amir M. Rahmani, and Mehdi Hosseinzadeh. „Reliability and high availability in cloud computing environments: a reference roadmap“. In: *Human-centric Computing and Information Sciences* 8.1 (July 2018), p. 20. DOI: 10.1186/s13673-018-0143-8.

[54] Wazir Z. Khan et al. „Edge computing: A survey“. In: *Future Generation Computer Systems* 97 (2019), pp. 219–235. DOI: 10.1016/j.future.2019.02.050.

[55] Nasir Abbas et al. „Mobile Edge Computing: A Survey“. In: *IEEE Internet of Things Journal* 5.1 (2018), pp. 450–465. DOI: 10.1109/JIOT.2017.2750180.

[56] Srinivasan A. Bragadeesh and Uimamakeswari Arumugam. „A Conceptual Framework for Security and Privacy in Edge Computing“. In: *Edge Computing: From Hype to Reality*. Ed. by Fadi Al-Turjman. Springer International Publishing, 2019, pp. 173–186. ISBN: 9783319990613. DOI: 10.1007/978-3-319-99061-3_10.

[57] Breno Costa et al. „Orchestration in Fog Computing: A Comprehensive Survey“. In: *ACM Comput. Surv.* 55.2 (Jan. 2022). DOI: 10.1145/3486221.

[58] Prateek Sharma et al. „Containers and Virtual Machines at Scale: A Comparative Study“. In: *Proceedings of the 17th International Middleware Conference*. Middleware '16. Trento, Italy: Association for Computing Machinery, 2016. ISBN: 9781450343008. DOI: 10.1145/2988336.2988337.

[59] European Telecommunications Standards Institute (ETSI). *Network Functions Virtualisation (NFV); Architectural Framework*. Group Specification (GS) NFV 002. Version V1.2.1. 2014.

[60] Lusani Mamushiane et al. „Overview of 9 Open-Source Resource Orchestrating ETSI MANO Compliant Implementations: A Brief Survey“. In: *2019 IEEE 2nd Wireless Africa Conference (WAC)*. 2019, pp. 1–7. DOI: 10.1109/AFRICA.2019.8843421.

[61] Steven Macenski et al. „Robot Operating System 2: Design, architecture, and uses in the wild“. In: *Science Robotics* 7.66 (2022). DOI: 10.1126/scirobotics.abm6074.

[62] Steven Macenski et al. „The Marathon 2: A Navigation System“. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020. DOI: 10.1109/IROS45743.2020.9341207.

[63] Morgan Quigley et al. „ROS: an open-source Robot Operating System“. In: *ICRA workshop on open source software*. Vol. 3. 2009.

[64] Xenofon Foukas et al. „FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks“. In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '16. Irvine, California, USA: Association for Computing Machinery, 2016, pp. 427–441. ISBN: 9781450342926. DOI: 10.1145/2999572.2999599.

[65] Ben Pfaff et al. „The Design and Implementation of Open vSwitch“. In: *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 117–130. ISBN: 9781931971218. URL: https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff (visited on 01/19/2023).

[66] Nick McKeown et al. „OpenFlow: Enabling Innovation in Campus Networks“. In: *SIGCOMM Comput. Commun. Rev.* 38.2 (Mar. 2008), pp. 69–74. DOI: 10.1145/1355734.1355746.

[67] Farhad Rezazadeh et al. „Actor-Critic-Based Learning for Zero-touch Joint Resource and Energy Control in Network Slicing“. In: *ICC 2021 - IEEE International Conference on Communications*. 2021, pp. 1–6. DOI: 10.1109/ICC42927.2021.9500265.

[68] Navid Nikaein, Xenofon Vasilakos, and Anta Huang. „LL-MEC: Enabling Low Latency Edge Applications“. In: *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. 2018, pp. 1–7. DOI: 10.1109/CloudNet.2018.8549500.

[69] Ilkka Harjula et al. „Smart Manufacturing Multi-Site Testbed with 5G and Beyond Connectivity“. In: *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. 2021, pp. 1–6. DOI: 10.1109/PIMRC50174.2021.9569284.

[70] Esa Piri et al. „5GTN: A test network for 5G application development and testing“. In: *2016 European Conference on Networks and Communications (EuCNC)*. 2016, pp. 313–318. DOI: 10.1109/EuCNC.2016.7561054.

[71] Norman Finn. „Introduction to Time-Sensitive Networking“. In: *IEEE Communications Standards Magazine* 2.2 (2018), pp. 22–28. DOI: 10.1109/MCOMSTD.2018.1700076.

[72] 5G-ACIA. *Integration of 5G with Time-Sensitive Networking for Industrial Communications*. White Paper. 5G Alliance for Connected Industries and Automation (5G-ACIA), 2021.

[73] Wolfgang Hess et al. „Real-time loop closure in 2D LIDAR SLAM“. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278. DOI: 10.1109/ICRA.2016.7487258.

[74] Stefano Aldegheri et al. „Late Breaking Results: Enabling Containerized Computing and Orchestration of ROS-based Robotic SW Applications on Cloud-Server-Edge Architectures“. In: *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 2020, pp. 1–2. DOI: 10.1109/DAC18072.2020.9218659.

[75] Shunki Shibuya et al. „Seamless Rapid Prototyping with Docker Container for Mobile Robot Development“. In: *2022 61st Annual Conference of the Society of Instrument and Control Engineers (SICE)*. IEEE. 2022, pp. 1063–1068. DOI: 10.23919/SICE56594.2022.9905781.

[76] Anton Filatov et al. „2D SLAM quality evaluation methods“. In: *2017 21st Conference of Open Innovations Association (FRUCT)*. 2017, pp. 120–126. DOI: 10.23919/FRUCT.2017.8250173.

[77] Arthur Huletski, Dmitriy Kartashov, and Kirill Krinkin. „Evaluation of the modern visual SLAM methods“. In: *2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)*. 2015, pp. 19–25. DOI: 10.1109/AINL-ISMW-FRUCT.2015.7382963.

[78] Philipp Schulz et al. „Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture“. In: *IEEE Communications Magazine* 55.2 (2017), pp. 70–78. DOI: 10.1109/MCOM.2017.1600435CM.

[79] Florian Voigtländer et al. „5G for Robotics: Ultra-Low Latency Control of Distributed Robotic Systems“. In: *2017 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*. 2017, pp. 69–72. DOI: 10.1109/ISCSIC.2017.27.

[80] Sharon Choy et al. „The brewing storm in cloud gaming: A measurement study on cloud to end-user latency“. In: *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*. 2012, pp. 1–6. DOI: 10.1109/NetGames.2012.6404024.

[81] Batyr Charyyev, Engin Arslan, and Mehmet H. Gunes. „Latency Comparison of Cloud Datacenters and Edge Servers“. In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9322406.

[82] Trung-Kien Le, Umer Salim, and Florian Kaltenberger. „An Overview of Physical Layer Design for Ultra-Reliable Low-Latency Communications in 3GPP Releases 15, 16, and 17“. In: *IEEE Access* 9 (2021), pp. 433–444. DOI: 10.1109/ACCESS.2020.3046773.

[83] 3GPP. *NR;Physical channels and modulation*. Technical Specification (TS) 38.211. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2021.

[84] Dario Sabella. „MEC: Standards and Industry Associations Around Edge Computing“. In: *Multi-access Edge Computing: Software Development at the Network Edge*. Ed. by Dario Sabella. Springer International Publishing, 2021. Chap. 2, pp. 19–56. ISBN: 9783030796181. DOI: 10.1007/978-3-030-79618-1_2.

[85] Dario Sabella. „MEC Standards on Edge Platforms“. In: *Multi-access Edge Computing: Software Development at the Network Edge*. Ed. by Dario Sabella. Springer International Publishing, 2021. Chap. 3, pp. 59–87. ISBN: 9783030796181. DOI: 10.1007/978-3-030-79618-1_3.

[86] Dario Sabella. „MEC Standards on Edge Services“. In: *Multi-access Edge Computing: Software Development at the Network Edge*. Ed. by Dario Sabella. Springer International Publishing, 2021. Chap. 4, pp. 89–114. ISBN: 9783030796181. DOI: 10.1007/978-3-030-79618-1_4.

[87] Dario Sabella. „Edge Computing in 5G Networks“. In: *Multi-access Edge Computing: Software Development at the Network Edge*. Ed. by Dario Sabella. Springer International Publishing, 2021. Chap. 7, pp. 201–243. ISBN: 9783030796181. DOI: 10.1007/978-3-030-79618-1_7.

[88] Yun C. Hu et al. *Mobile Edge Computing A key technology towards 5G*. White Paper 11. European Telecommunications Standards Institute (ETSI), 2015.

[89] Sami Kekki et al. *MEC in 5G networks*. White Paper 28. European Telecommunications Standards Institute (ETSI), 2018.

[90] 3GPP. *Edge Computing Management*. Technical Specification (TS) 28.538. Version 17.0.0. 3rd Generation Partnership Project (3GPP), 2022.

[91] Nurit Sprecher et al. *Harmonizing standards for edge computing*. White Paper 36. European Telecommunications Standards Institute (ETSI), 2020.

[92] 3GPP. *Procedures for the 5G System (5GS)*. Technical Specification (TS) 23.502. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[93] 3GPP. *Policy and charging control framework for the 5G System (5GS)*. Technical Specification (TS) 23.503. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[94] 3GPP. *Study on enhancement of support for Edge Computing in 5G Core network (5GC)*. Technical Report (TR) 28.538. Version 17.0.0. 3rd Generation Partnership Project (3GPP), 2020.

[95] 3GPP. *NG-RAN; Architecture description*. Technical Specification (TS) 38.401. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2021.

[96] 3GPP. *Principles and Guidelines for Services Definition*. Technical Specification (TS) 29.500. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[97] Akeem O. Mufutau et al. „Software-Defined Radio Enabled Cloud Radio Access Network Implementation Using OpenAirInterface“. In: *Wireless Personal Communications* 121.2 (Nov. 2021), pp. 1233–1253. DOI: 10.1007/s11277-021-09064-0.

[98] 3GPP. *Policy and Charging Control signalling flows and QoS parameter mapping; Stage 3*. Technical Specification (TS) 29.513. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[99] Thomas H. Cormen et al. „Graph Algorithms“. In: *Introduction to Algorithms*. Fourth Edition. The MIT Press, 2022. Chap. 6, pp. 547–744. ISBN: 9780262367509.

[100] 3GPP. *Interface between the Control Plane and the User Plane nodes*. Technical Specification (TS) 24.244. Version 16.5.0. 3rd Generation Partnership Project (3GPP), 2020.

[101] 3GPP. *Policy and Charging Control (PCC); Reference points*. Technical Specification (TS) 29.212. Version 16.4.0. 3rd Generation Partnership Project (3GPP), 2020.

[102] IETF. *Diameter Base Protocol*. Request for Comments (RFC) 6733. Internet Engineering Task Force (IETF), 2012.

[103] Danilo Cerović et al. „Fast Packet Processing: A Survey“. In: *IEEE Communications Surveys & Tutorials* 20.4 (2018), pp. 3645–3676. DOI: 10.1109/COMST.2018.2851072.

[104] Jesus Sanchez-Gomez, Ramon Sanchez-Iborra, and Antonio Skarmeta. „Transmission Technologies Comparison for IoT Communications in Smart-Cities“. In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 2017, pp. 1–6. DOI: 10.1109/GLOCOM.2017.8254530.

[105] Nathan Koenig and Andrew Howard. „Design and use paradigms for Gazebo, an open-source multi-robot simulator“. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[106] Gregor Klančar et al. „Motion Modeling for Mobile Robots“. In: *Wheeled Mobile Robotics*. Ed. by Gregor Klančar et al. Butterworth-Heinemann, 2017. Chap. 2, pp. 13–59. ISBN: 9780128042045. DOI: 10.1016/B978-0-12-804204-5.00002-0.

[107] Norhafizan Ahmad et al. „Reviews on various inertial measurement unit (IMU) sensor applications“. In: *International Journal of Signal Processing Systems* 1.2 (2013), pp. 256–262.

[108] Stefan Profanter et al. „OPC UA versus ROS, DDS, and MQTT: Performance Evaluation of Industry 4.0 Protocols“. In: *2019 IEEE International Conference on Industrial Technology (ICIT)*. 2019, pp. 955–962. DOI: 10.1109/ICIT.2019.8755050.

[109] Jan Dürre et al. „A Disaggregated 5G Testbed for Professional Live Audio Production“. In: *2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. 2022, pp. 1–6. DOI: 10.1109/BMSB55706.2022.9828613.

[110] Shinji Umeyama. „Least-Squares Estimation of Transformation Parameters Between Two Point Patterns“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.04 (Apr. 1991), pp. 376–380. DOI: 10.1109/34.88573.

Web References

[W1] *Announcements of commercial launches.* URL: https://5gobservatory.eu/market-developments/5g-services (visited on 01/19/2023).

[W2] *Why do we need 5G?* URL: https://www.etsi.org/technologies/5G (visited on 01/19/2023).

[W3] ENEA. *Survey: One Third of Mobile Operators Will Deploy 5G Standalone within Two Years.* 2020. URL: https://www.enea.com/news/press-releases/survey-one-third-of-mobile-operators-will-deploy-5g-standalone-within-two-years (visited on 01/19/2023).

[W4] *5G Edge Computing Infrastructure - AWS Wavelength - Amazon Web Services.* URL: https://aws.amazon.com/wavelength (visited on 01/19/2023).

[W5] *Setting up a robot simulation (Gazebo) – ROS 2 Documentation: Humble documentation.* URL: https://docs.ros.org/en/humble/Tutorials/Advanced/Simulators/Gazebo.html (visited on 01/19/2023).

[W6] *Control and User Plane Separation of EPC nodes (CUPS).* URL: https://www.3gpp.org/cups (visited on 01/19/2023).

[W7] *Release 15.* URL: https://www.3gpp.org/specifications-technologies/releases/release-15 (visited on 01/19/2023).

[W8] RTR. *Frequenzbereiche | RTR.* URL: https://www.rtr.at/TKP/was_wir_tun/telekommunikation/spectrum/bands/FRQ_spectrum.de.htm (visited on 01/19/2023).

[W9] VMware. *VMware - Delivering a Digital Foundation For Businesses.* URL: https://www.vmware.com (visited on 01/19/2023).

[W10] Oracle. *Oracle VM VirtualBox.* URL: https://www.virtualbox.org (visited on 01/19/2023).

[W11] Docker Inc. *Docker: Accelerated, Containerized Application Development.* URL: https://www.docker.com (visited on 01/19/2023).

[W12] Kubernetes. *Kubernetes.* URL: https://kubernetes.io (visited on 01/19/2023).

[W13] Red Hat. *Red Hat OpenShift makes container orchestration easier.* URL: https://www.redhat.com/en/technologies/cloud-computing/openshift (visited on 01/19/2023).

[W14] *Introducing tf2 - ROS 2 Documentation: Humble documentation.* URL: https://docs.ros.org/en/humble/Tutorials/Intermediate/Tf2/Introduction-To-Tf2.html (visited on 01/19/2023).

[W15] *Lua configuration reference documentation - Cartographer ROS documentation.* URL: https://google-cartographer-ros.readthedocs.io/en/latest/configuration.html (visited on 01/19/2023).

[W16] *Release 17.* URL: https://www.3gpp.org/specifications-technologies/releases/release-17 (visited on 01/19/2023).

[W17] *Ettus Research - The leader in Software Defined Radio (SDR).* URL: https://www.ettus.com (visited on 01/19/2023).

[W18] *Benetel - OpenRAN Radio Units. Opening Possibilities.* URL: https://benetel.com (visited on 01/19/2023).

[W19] *RRH/RRU.* URL: https://www.aw2s.com/electronic-engineering/engineering-services (visited on 01/19/2023).

[W20] *5G CORE NETWORK.* URL: https://openairinterface.org/oai-5g-core-network-project (visited on 01/19/2023).

[W21] *OpenAPI Specification.* URL: https://swagger.io/specification (visited on 01/19/2023).

[W22] *Pistache.* URL: https://pistacheio.github.io/pistache (visited on 01/19/2023).

[W23] *oaisoftwarealliance's Profile | Docker Hub.* URL: https://hub.docker.com/u/oaisoftwarealliance (visited on 01/19/2023).

[W24] *GitHub - jbeder/yaml-cpp: A YAML parser and emitter in C++.* URL: https://github.com/jbeder/yaml-cpp (visited on 01/19/2023).

[W25] *GitHub - travelping/upg-vpp: User Plane Gateway (UPG) based on VPP.* URL: https://github.com/travelping/upg-vpp (visited on 01/19/2023).

[W26] *Gazebo.* URL: https://gazebosim.org/home (visited on 01/19/2023).

[W27] *Gazebo.* URL: https://classic.gazebosim.org (visited on 01/19/2023).

[W28] *SDFormat Home.* URL: http://sdformat.org (visited on 01/19/2023).

[W29] *Useful World Files for Gazebo and ROS 2 Simulations - Automatic Addison.* URL: https://automaticaddison.com/useful-world-files-for-gazebo-and-ros-2-simulations/#Factory_World (visited on 01/19/2023).

[W30] *GitHub - wh200720041/warehouse_simulation_toolkit.* URL: https://github.com/wh200720041/warehouse_simulation_toolkit (visited on 01/19/2023).

[W31] *GitHub - ROBOTIS-GIT/turtlebot3: ROS packages for Turtlebot3.* URL: https://github.com/ROBOTIS-GIT/turtlebot3 (visited on 01/19/2023).

[W32] *16.2. Use ROS 2 with Fast-DDS Discovery Server - Fast DDS 2.7.1 documentation.* URL: https://fast-dds.docs.eprosima.com/en/latest/fastdds/ros2/discovery_server/ros2_discovery_server.html (visited on 01/19/2023).

[W33] *docs/DEPLOY_HOME.md · master · oai / cn5g / oai-cn5g-fed · GitLab.* URL: https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_HOME.md (visited on 01/19/2023).

[W34] *ros2/rviz: ROS 3D Robot Visualizer.* URL: https://github.com/ros2/rviz (visited on 01/19/2023).

[W35] *GitHub - MichaelGrupp/evo: Python package for the evaluation of odometry and SLAM.* URL: https://github.com/MichaelGrupp/evo (visited on 01/19/2023).

[W36] *OpenAirInterface RAN Roadmap.* URL: https://openairinterface.org/wp-content/uploads/2022/07/2022-07-12-EURECOM-RAN-SLIDES.pdf (visited on 01/19/2023).

[W37] *nload(1): displays current network usage - Linux man page.* URL: https://linux.die.net/man/1/nload (visited on 01/19/2023).

[W38] *5G Network Slicing | Android Open Source Project.* URL: https://source.android.com/docs/core/connect/5g-slicing (visited on 01/19/2023).