Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Telecommunication Engineering

Development of Testbed for Vehicular Edge Computing

Bachelor Thesis

Vojtěch Hašpl

Prague, May 2023

Bachelor Programme: Electronics and Communications

Supervisor: Ing. Jan Plachý, Ph.D.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Hašpl Vojtěch**	Personal ID number:	**499277**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Telecommunications Engineering**		
Study program:	**Electronics and Communications**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Development of Testbed for Vehicular Edge Computing

Bachelor’s thesis title in Czech:

Vývoj testbedu pro Vehicular Edge Computing

Guidelines:

Familiarize yourself with 5G mobile networks, platforms for emulation of mobile networks and the control of radio access network. Study the concept of edge computing with primary focus on vehicular edge computing, that enables computation heavy demanding processing to be done in the edge computing servers. Evaluate existing solutions for vehicular edge computing offloading, management of communication and computation resources and propose possible enhancements. Based on the existing works setup a working testbed for computation offloading from vehicles via 5G mobile network, evaluate its performance and outline possible future development towards 6G mobile networks.

Bibliography / sources:

[1] L. Bréhon–Grataloup, R. Kacimi, and A.L. Beylot,“Mobile edge computing for V2X architectures and applications: A survey.“, Computer Networks, 206, p.108797, 2022.
[2] S. Baidya, K. Yu-Jen, Z. Hengyu, Z. Jishen, and D. Sujit, "Vehicular and edge computing for emerging connected and autonomous vehicle applications.", ACM/IEEE Design Automation Conference (DAC), pp. 1-6. IEEE, 2020.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Plachý, Ph.D. Department of Telecommunications Engineering FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **12.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Jan Plachý, Ph.D. Supervisor’s signature	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

. Date of assignment receipt	Student’s signature

I hereby declare I have written this bachelor thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

Prague, May, 2023

. . . .
Signature

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 25. května 2023

. . . .

Podpis

Acknowledgements

Thank you

Abstract

Recent advancements in Autonomous Vehicles (AVs) research are bringing AVs closer to widespread adoption. However, the computational requirements of AVs demand robust hardware resources, resulting in substantial power consumption for the vehicles. This has led to the emergence of Vehicular Edge Computing (VEC), a research area focused on utilization of processing resources at mobile networks' edge for offloading computationally demanding tasks from vehicles at low latency. To enable authentic simulation for evaluating VEC algorithms, a testbed leveraging Multi-Access Edge Computing (MEC) is proposed in this thesis. The set up testbed is based on MEC platform by OpenAirInterface (OAI) and the underlying OAI 5G mobile network, which facilitates accurate and scalable simulations. In evaluation against cloud deployed 400 kilometers away, the testbed achieves approximately 59 % lower Round Trip Time (RTT) demonstrating its viability for realistic evaluation of VEC algorithms.

Keywords: vehicular edge computing, multi-access edge computing, mobile networks, 5G, task offloading, autonomous vehicles

Abstrakt

Klíčová slova: vehicular edge computing, multi-access edge computing, mobilní sítě, 5G, přesun výpočtů, autonomní vozidla

Contents

List of Abbreviations

3GPP Third Generation Partnership Project.

AF Application Funciton.

AMF Access and Mobility Management Function.

API Application Programmable Interface.

AUSF Authentication Server Function.

AVs Autonomous Vehicles.

B2B Business-to-Business.

CC Cloud Computing.

CN Core Network.

COTS Customer Off-The-Shelf.

CUPS Control and User Plane Separation.

CVB Connected Vehicles Blueprint.

DN Data Network.

EALTEdge Enterprise Application on Lightweight Telco Edge.

EC Edge Computing.

eMBB Enhanced Mobile Broadband.

eNB evolved NodeB.

EPC Evolved Packet Core.

ETSI Eurupean Telecommunication Standard Institution.

FDD Frequency Division Duplex.

FEs Functional Entities.

FlexCN	Flexible Core Controller.
FlexRIC	Flexible RAN Intelligent Controller.
gNB	new generation NodeB.
HTTP	Hypertext Transfer Protocol.
IoT	Internet of Things.
IP	Internet Protocol.
ISG	Industry Specification Group.
k8s	Kubernetes.
LL-MEC	Low-Latency MEC.
MAC	Media Access Control.
MEC	Multi-Access Edge Computing.
MEC apps	MEC Applications.
MEO	MEC Orchestrator.
MEP	MEC Platform.
MEPM	MEC Platform Manager.
Mm	MEC Management reference point.
mMTC	Massive Machine Type Communication.
Mp	MEC platform functionality reference point.
Mx	Reference point connecting to external entities.
NAS	Non-Access Stratum.
NEF	Network Exposure Funciton.
NF	Network Functions.
nFAPI	Network Functional API.
NFV	Network Function Virtualization.
NR	New Radio.
NRF	Network Repository Function.
NSSF	Network Slice Selection Function.

O-RAN	Open Radio Access Network.
OAI	OpenAirInterface.
OpenNESS	Open Network Edge Services Software.
OSA	OpenAirInterface Software Alliance.
OSS	Operations Support System.
PCEI	Public Cloud Edge Interface.
PCF	Policy Control Function.
PDCP	Packet Data Convergence Protocol.
PDU	Packet Data Unit.
PHY	Physical Layer.
QoS	Quality of Service.
RAN	Radio Access Network.
REST	Representational State Transfer.
RLC	Radio Link Control.
RNIS	Radio Network Information Service.
RRC	Radio Resource Control.
RTT	Rount Trip Time.
SBA	Service Based Architecture.
SDAP	Service Data Application Protocol.
SDN	Software Defined Networking.
SDO	Standard Defining Organization.
SMF	Session Management Function.
SMS	Short Message Service.
TCP	Transmission Control Protocol.
TDD	Time Division Duplex.
UALCMP	User Application Lifecycle Management Proxy.
UDM	Unified Data Management.

UDR Unified Data Repository.

UE User Equipment.

UL CL Uplink Classifier.

UPF User Plane Function.

URLLC Ultra Reliable Low Latency Communication.

VEC Vehicular Edge Computing.

VIM Virtualization Infrastructure Manager.

VM Virtual Machine.

VR Virtual Reality.

1. Introduction

Mobile networks provide connectivity to mobile users but through their recent development they are seen as a key enabler for emerging services and applications in areas such as Internet of Things (IoT) or automotive industry, whose requirements on communication data rate and latency are ever increasing. This drives the need for innovation and development of mobile networks. The development of mobile networks follows standards defined by multiple standards defining organizations (SDO). One of the SDOs is the Third Generation Partnership Project (3GPP), which unites seven telecommunications standard development organizations and their members, that define exact standards for all aspects of mobile networks such as architecture, protocols, interfaces, modulation techniques etc. The European SDO taking part in 3GPP is European Telecommunications Standard Institution (ETSI). Another noteworthy body involved in mobile network development is the Open Radio Access Network (O-RAN) whose aim is to promote development of open specifications for RAN components [**?**].

Development of mobile networks had started in 1980s with the first generation of mobile networks (1G) that focused on analog communication and enabled voice calls [1]. The 1G was followed by 2G which replaced the analog communication with digital communication and introduced short messages service (SMS) [1]. 3G, as a successor, has mainly focused on enhancing data rates to enable multimedia applications such as video streaming [1]. 4G then disrupted mobile networks design by switching to fully IP based architecture and further enhancing data rates and decreasing communication latency [1,2]. The 4G is followed by the 5G which is focused on increasing the number of communicating devices, as well as further reducing communication latency.

The initial deployment of 5G started in 2019 and is still ongoing. 5G has been developed with the aim of providing support not only for fast internet access, but also massive machine-type communication and mission critical communication [3]. The 5G is a flexible mobile network, capable of supporting novel applications and services, such as virtual reality (VR), IoT or connected vehicles. The development of the mobile network from 4G is seen both in the Radio Access Network (RAN) with introduction of 5G New Radio (NR) interface and in the core network (CN) by evolving the 4G CN into a Service Based Architecture (SBA) that provides as a flexible platform, capable of integrating novel applications and techniques like edge computing [4].

Edge computing (EC) can be described as an extension of Cloud Computing (CC) to the network edge [4]. CC is a computing paradigm that provides mobile network users' access to a pool of resources, e.g. (storage, computing, applications and services), through the network in similar fashion to the wired networks [5]. The EC brings the resources closer to the user, to the so-called network edge to offer services with low latency. The low latency is critical for applications such as autonomous vehicles (Avs). An EC implemented in the mobile networks is called Multi-Access Edge Computing (MEC), which has been formerly known as Mobile Edge Computing. MEC is a RAN access agnostic, supporting mobile as well as Wi-Fi networks. The leading international SDO focused on MEC is ETSI that develops ETSI MEC standards within Industry Specification Group (ISG). The aim of ETSI MEC is to create an open standardized environment allowing for seamless integration of applications from vendors, service providers as well as third parties [**?**]. In this thesis ETSI MEC standards developed are followed to achieve full compliance with standardized networks.

Through offering sizeable computational resources available at low latency, MEC enhances various use cases such as IoT, augmented reality, location services or Avs by providing high powered computing capabilities to computation limited devices [**?**]. AVs require processing of large amounts of data in order to recognize the surrounding environment and decide the next best action (slow down, speed up, turn etc.). This is a computationally demanding process, posing a computation challenge to the local (vehicle) limited computation capacity. As shown in recent research [7], the computation required for level-4 and level-5 Avs requires high performance server grade hardware. This is where MEC can be leveraged to offload some of the data processing from the AV to the network edge. This enables to drive down the vehicles' cost and power consumption, while keeping the communication latency low. The utilization of MEC for vehicular applications is referred to as vehicular edge computing (VEC).

As of today, VEC development is still in progress. To exploit computation offloading by vehicles with sufficiently low latency, algorithms for allocation of computation and communication resources need to be developed and evaluated. Therefore, this thesis focuses on developing a functional testbed for VEC. The structure of the thesis is as follows: first, integration of MEC in 5G networks is explained. Then, in Chapter 3, several existing open-source solutions for emulating network edge computing are examined and a suitable platform is selected for integration with emulated 5G mobile network. Chapter 4 provides a description of OpenAirInterface for emulation of mobile networks. Chapter 5 is dedicated towards setting up the VEC testbed and its evaluation against reasonably distant cloud computing. Finally, in the last chapter, the thesis is concluded, and future work outlined.

2. MEC & 5G

The aim of this chapter is to provide a detailed description of MEC integration into 5G mobile networks. It is possible to integrate MEC into 4G, as described by ETSI [8], but as MEC has been standardized when 4G networks have been already deployed, it is mostly deployed with 5G enabling seamless integration of MEC [9] and providing lower latency [3], which is a crucial parameter for edge applications such as AVs. In this chapter, both 5G and MEC system architectures are described, followed by a framework for integration of MEC system into 5Gmobile network.

2.1 Key concepts in 5G

The evolution of mobile networks from 4G to 5G has focused not only on providing improved experience for end users, through enhancing communication data rates and decreasing communication latency, but also on introducing novel industry business opportunities, including Business to Business (B2B) [10]. To achieve that, two new industry-oriented communication scenarios have been introduced to complement the end user oriented one – Enhanced Mobile Broadband (eMBB). The newly introduced usage scenarios are Massive Machine-Type Communication (mMTC) and Ultra-Reliable Low Latency Communication (URLLC) [10]. Requirements of the three mentioned communication services are illustrated in Figure 2.1.

As shown in Figure 2.1, three illustrated usage scenarios for 5G largely vary in requirements, such as user experienced data rate, connection density or latency. Trying to cater for all three at the same time would be very inefficient because in serving such varying use cases, it is difficult to accurately predict service requirements which may lead to underutilization of network resources. Therefore, the concept of network slicing has been introduced [11]. Network slicing works by creating multiple, separated logical end-to-end networks on the same physical infrastructure [11]. Each of these logical networks can then be orchestrated individually, according to the specific service requirements [11]. In the context of AVs, this would allow for operation of AVs tailored network slice, providing low latency-oriented service.

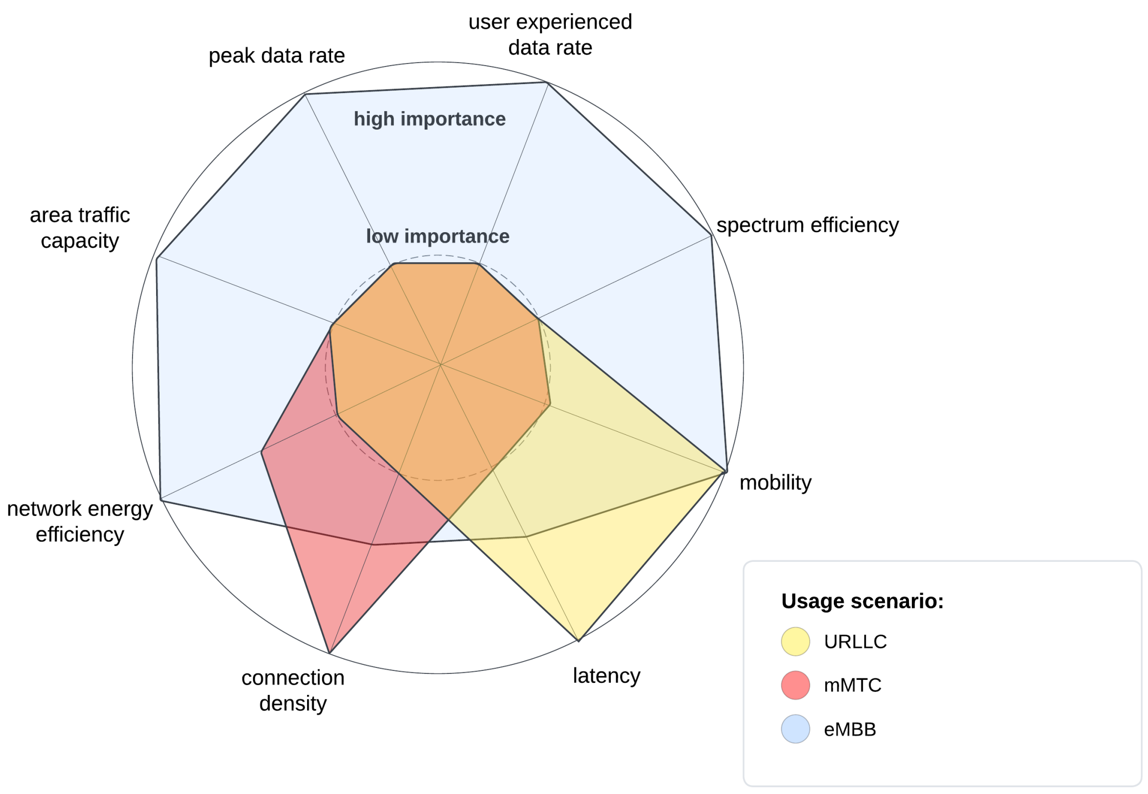


Figure 2.1. Usage scenarios in 5G, based on [3]

The concept of network slicing requires new approaches for managing network resources. The technologies identified as enablers for network slicing are Network Function Virtualization (NFV) and Software Defined Networking (SDN) [11]. NFV aims to leverage virtualization techniques to improve mobile networks' flexibility, agility and scalability [11]. SDN, on the other hand, aims to provide programmability to mobile networks' services and software-based flow control of user data [11].

Another important concept, although not new in 5G, is Control and User Plane Separation (CUPS). CUPS allows for distributed CN deployments, where the user plane can be deployed closer to the base station (and also end user) for reduced latency and saving core network resources. While the user plane solution for both 5G CN and 4G CN, the Evolved Packet Core (EPC), is still quite similar, the control plane underwent a revolution through introduction of SBA [10]. The 5G CN Network Functions (NFs) are not connected with point-to-point interfaces, but rather on a common bus, while exposing services to other NFs via Application Programmable Interfaces (APIs) [10]. In practice, this can be achieved through providing HTTP REST APIs [4, 10]. HTTP (Hyper Text Transfer Protocol) is a request response communication model protocol, while REST (Representational State Transfer) is an architecture style for creating flexible systems that can communicate with each other, using HTTP methods GET, POST, PUT and DELETE.

2.2 5G system architecture

This section aims to introduce and describe the 5G system architecture with focus on NFs playing key role in MEC integration. As described earlier, 5G CN control plane NFs utilize SBA, which means that they are all logically interconnected. The user plane, however, is still connected to the rest of the network using point-to-point interfaces. The 5G system architecture depicted in Figure 2.2 is described in [12].

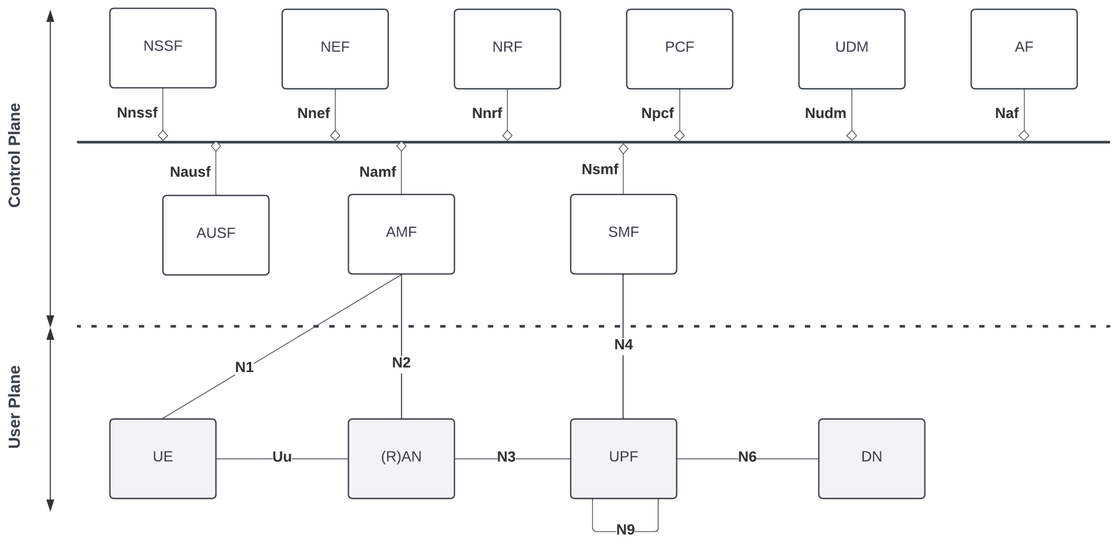


Figure 2.2. 5G CN, based on [12]

The user plane consists of User Equipment (UE), Radio Access Network (RAN), User Plane Function (UPF) and Data Networks Function (DN). The brackets around 'R' in RAN in Figure 2.2 indicate that 5G core network (CN) is not designed exclusively for radio access technologies, but instead aims to be access agnostic. UE is connected to RAN via the Uu interface, while RAN itself is connected to UPF via the N2 interface. The main role of UPF is to process and forward user data. It also connects UEs with IP networks. UPF can be deployed closer to the end users, which reduces communication latency. This is crucial for MEC deployment, because UPF can be deployed at the network edge with the edge server and UPF steering traffic directly to MEC, significantly reducing latency in comparison to 4G, where the traffic always has to be steered through the CN. [10]

The control plane consists of Network Slice Selection Function (NSSF), Network Exposure Function (NEF), Authentication Server Function (AUSF), Network Repository Function (NRF), Access and Mobility Management Function (AMF), Policy Control Function (PCF), Session Management Function (SMF), Unified Data Management (UDM) and Application Function (AF). The control plane NFs directly interacting with user plane are AMF and SMF. These NFs together are largely responsible for Packet Data Unit (PDU) session management, such as initial attach of UE, mobility management or setting up radio bearers. [10]

The AMF deserves special notice, as it relays all control signaling messages from other CN NFs towards RAN and UE via the N2 and N1 interfaces. AMF allows UEs to register, authenticate themselves and, in coordination with RAN and other NFs, ensures smooth handover between cells [10]. Although RAN is not extensively discussed in this thesis, it is beneficial to establish how RAN and CN interact to set up communication channels. AMF provides the RAN with Quality of Service (QoS) parameters, which are used by the Radio Resource Control (RRC) protocol to set up appropriate radio bearers, i.e., logical radio connections between UEs and RAN [3].

To understand how MEC can be integrated to 5G system, it is necessary to describe PDU session management. The PDU management provides connectivity between UEs and DNs, mostly the Internet. To set up this connection, a UE sends a PDU session establishment request to an AMF, which instructs a SMF. What follows is a series of control plane signaling messages between the SMF and various other NFs to properly register the PDU session, set up (R)AN and create a UPF tunnel towards the DN. The simplified PDU session establishment process is illustrated in Figure 2.3. 5G system architecture allows for the UE to be to be connected to multiple DNs at a time. One PDU session can connect the UE to the Internet, while another one can be set up to connect it with, e.g., local edge server. To determine the correct endpoint DN for a given traffic, UPF uses an Uplink Classifier (Ul CL), which enables steering traffic towards appropriate DNs based on, e.g., uplink IP address. Crucially, configuring UL CL can be requested by AF as a SMF service. [10]

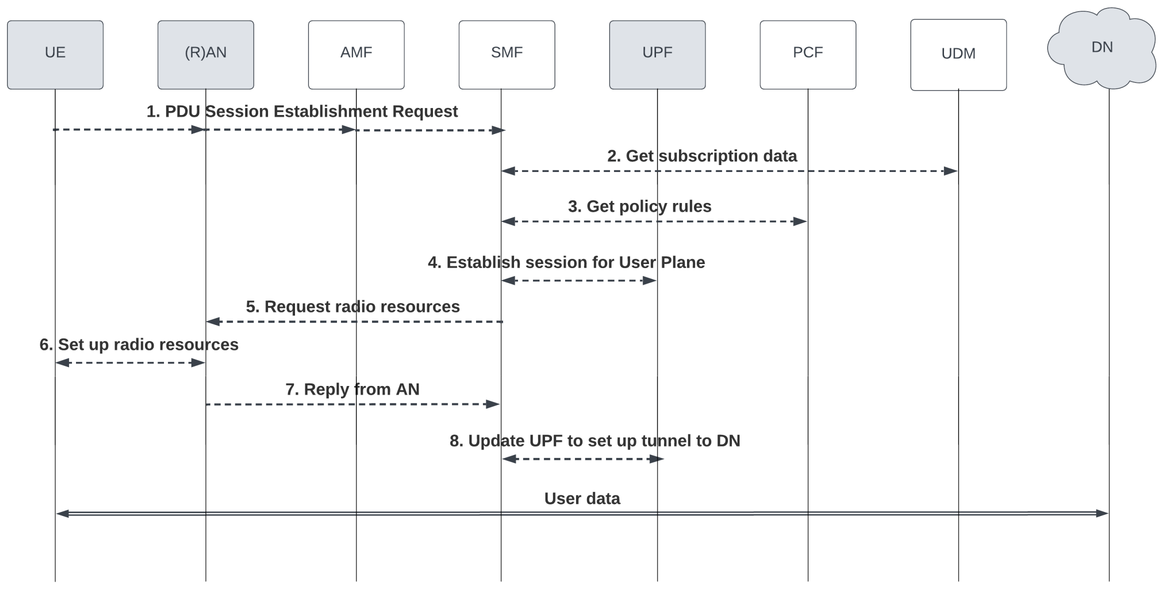

UE
(R)AN
AMF
SMF
UPF
PCF
UDM
DN
1. PDU Session Establishment Request
2. Get subscription data
3. Get policy rules
4. Establish session for User Plane
5. Request radio resources
6. Set up radio resources
7. Reply from AN
8. Update UPF to set up tunnel to DN
User data

Figure 2.3. PDU session establishment, based on [10]

The PDU session management was slightly innovated in 5G, as it introduces completely new workflows: service registration & discovery and capabilities exposure [10]. These workflows are crucial for understanding how the 5G system interacts with external applications such as MEC. Service registration & discovery, illustrated in Figure 2.4, describes how individual NFs within 5G system utilize the SBA to provide and consume services. Put simply, every service producer needs to register its services with NRF. When a service consumer wants to request services of a particular NF, it first requests NRF for a list of available NFs offering those services. The other workflow, capabilities exposure, describes specifically how external applications can interact with and influence the 5G system. The NFs crucial for this workflow are AF, which basically represents the external application, and NEF, which acts as a gateway between external applications and 5G system. Provided the external application is properly registered, NEF can, e.g., allow AF to influence traffic steering, managed by SMF, to forward traffic to a specific DN. This is of particular interest to MEC integration, since this workflow can allow MEC to influence traffic steering from UE to edge server. This flow is illustrated in Figure 2.5.

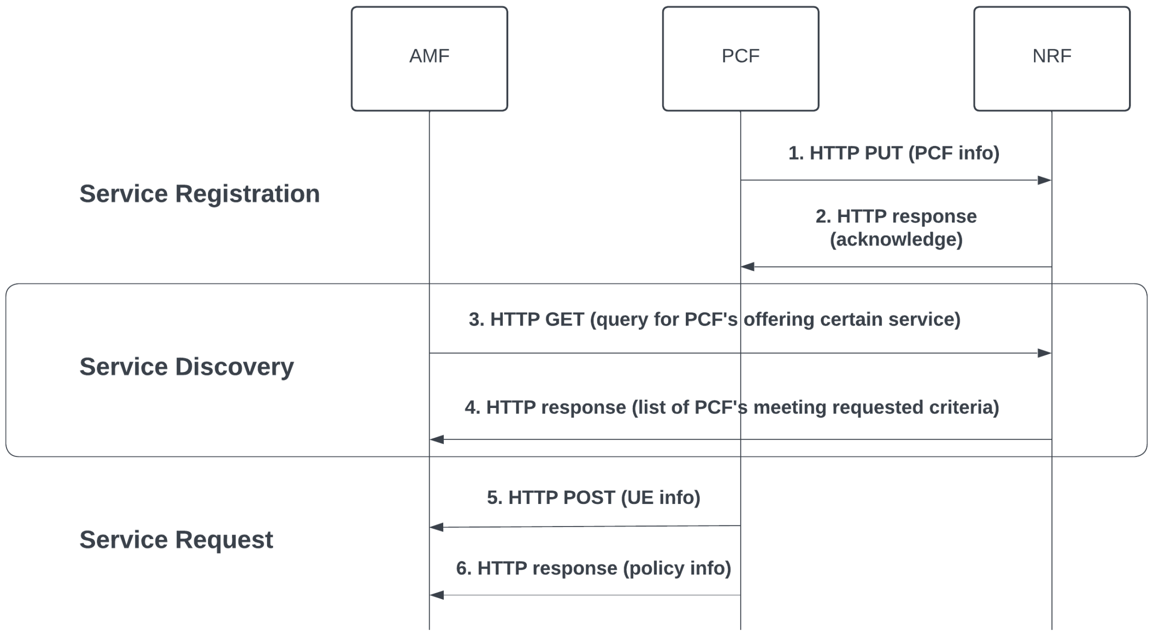

AMF
PCF
NRF
Service Registration
1. HTTP PUT (PCF info)
2. HTTP response (acknowledge)
Service Discovery
3. HTTP GET (query for PCF's offering certain service)
4. HTTP response (list of PCF's meeting requested criteria)
Service Request
5. HTTP POST (UE info)
6. HTTP response (policy info)

Figure 2.4. Service registration in discovery in 5G, based on [10]

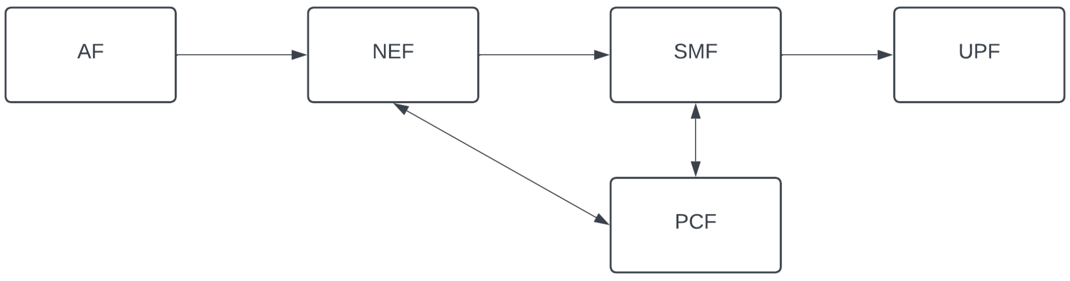


Figure 2.5. AF influenced traffic control, based on [10]

2.3 MEC system concept and architecture

Previous sections of this chapter focused on providing a general overview of the 5G system, as well as identifying individual workflows supporting MEC integration. This section provides a general overview of key functional entities (FEs) and workflows that play a role in connecting the MEC system to a 5G mobile network.

The general idea of MEC is to bring CC capabilities to the network edge to provide low latency access to computing resources. The low latency improves experience in existing cloud applications and enables emerging applications in industrial IoT or automotive sectors [13]. Thus, opening the 5G ecosystem to new vertical sectors and enabling exploitation of new business opportunities [4]. However, having multiple stakeholders in the development chain of end-to-end MEC solutions implies the need for standardization [13]. MEC standards play an important role in creating a common platform for Mobile Network Operators (MNOs) and MEC application developers from various verticals to aid the development of end-to-end MEC solutions. This section presents and describes the MEC system framework and reference architecture developed by ETSI ISG MEC, the leading standardization body in MEC.

The MEC framework, defined in [14], is illustrated in Figure 2.6, which shows the general entities involved on the system, host and network levels. The network level entities ensure connectivity to local area networks, cellular networks and external networks, e.g., the Internet. The host level, constituted by MEC host and its management, is the core of MEC system, because it represents the IT infrastructure on which MEC applications (MEC apps) are run. The MEC host can be further dissected into MEC platform, virtualization infrastructure and MEC applications [14]. The system level is responsible for management and orchestration of the whole MEC system, which is typically comprised of multiple MEC hosts [4, 14].

The reference architecture, illustrated in Figure 2.7, offers a more detailed insight into the general framework from Figure 2.6 and identifies reference points for connecting individual FEs of the MEC system. The reference points can be divided into three groups based on [14]:

- MEC platform functionality reference points (Mp)
- Management reference points (Mm)
- Reference points connecting to external entities (Mx)

Not all the illustrated reference points are specified by ETSI ISG MEC [14]. An example of ETSI ISG MEC specified interface is the Mp1, connecting MEC Platform and MEC apps. This is desirable since it is expected that MEC platform and MEC apps will be developed

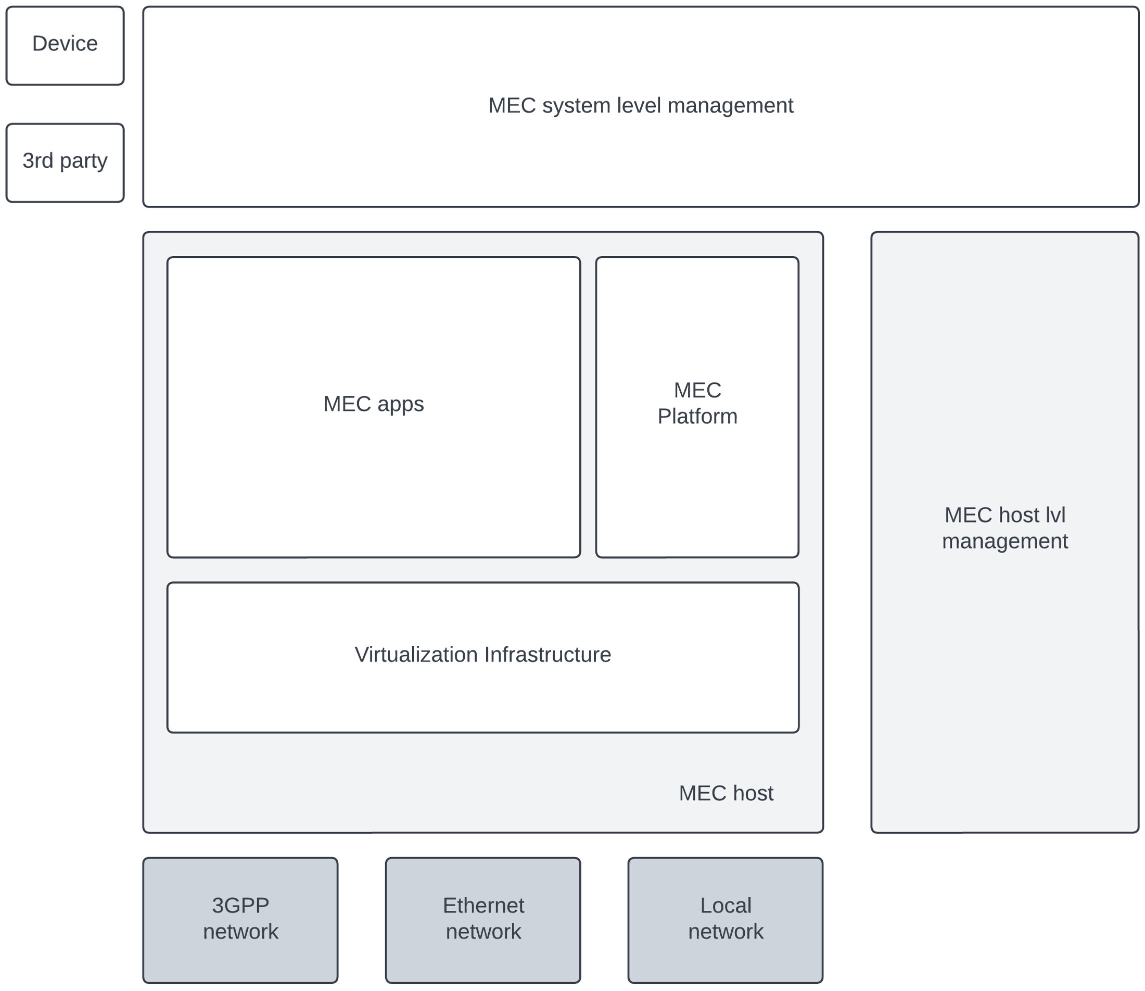


Figure 2.6. ETSI MEC framework, [14]

by different parties. An example of a closed (internal) interface is the Mm5, connecting MEC platform and its management entity, which are both expected to be developed by the same organization and are therefore not specified but left as an implementation choice [4, 14].

The MEC platform is a crucial MEC system FE, as it acts as an intermediary between MEC apps and the system level management. Through the Mp1 interface, MEC platform offers an environment where MEC apps may discover, advertise, consume and offer MEC services [15]. This is very similar to the SBA based workflows in 5G system. MEC platform offers the MEC apps similar functionality to what NEF and NRF offer to the 5G system. On top of that, the MEC platform has another very important responsibility. Upon request from a MEC app, MEC platform can activate, deactivate or update traffic rules and then configure the data plane accordingly [14]. This in turn is analogous to AF functionality within 5G system. These analogies are elaborated upon in the next section focusing on integration of MEC system to 5G mobile network.

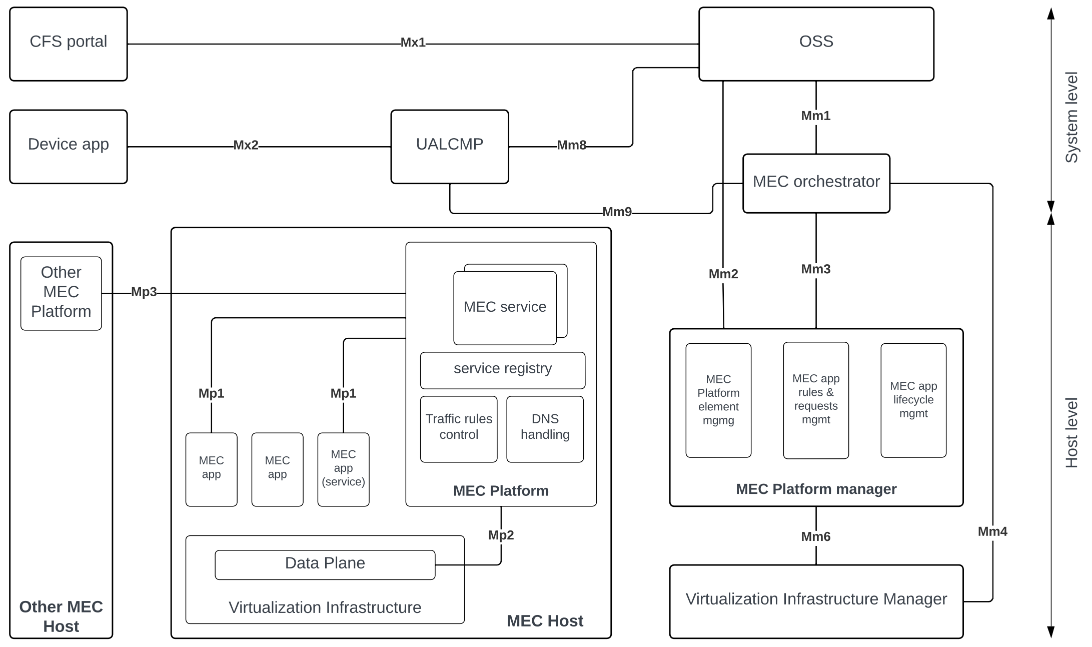


Figure 2.7. ETSI MEC reference architecture, [14]

Another important part of the MEC system is its management, split into host level and system level, which is illustrated in Figure 2.6. The individual FEs taking part in MEC management are illustrated in Figure 2.7. System level management is comprised of Operations Support System (OSS), User Application Life Cycle Management Proxy (UALCMP) and MEC orchestrator (MEO). MEO is a key FE overseeing the whole MEC system in terms of deployed MEC hosts, available resources and services etc. It also plays a key role in application onboarding or mobility handling. Host level management responsibilities are shared between two entities: MEC Platform Manager (MEPM) and Virtualization Infrastructure Manager (VIM). MEPM manages the lifecycle of applications and can inform MEO about MEC app related events, while VIM manages the virtualized resources of the virtualization infrastructure. [14]

Thus far, it has been described how MEC apps and MEC platform interact with each other to provide and consume services and how the whole MEC system is managed on both system and host levels. The last important workflow to be described is how the MEC system interacts with the UE, so that the UE can discover and exploit available MEC servers. There are two recognized approaches for edge discovery from the UE perspective: DNS based and device based [13]. The first option is called edge unaware, since the client application, i.e., the application instance in UE communicating with MEC app, has no information about the edge server location. Instead, the client application is connected to the MEC app via a DNS server, whose queries may be enriched by the 5G CN with the device location information [13,16]. The second approach, edge aware, requires a new entity on the client side, the device application, to communicate with the MEC system to receive the MEC application address and offer it to client applications [16]. The edge aware approach is more suited for highly distributed edge cloud exposed to UE mobility, because it allows for informing the device application about change of edge server [13]. In stateful applications, this requires the MEO to trigger context migration to another server [13, 14]

2.4 Integration of MEC into 5G system

The previous sections of this chapter provided a brief overview of both 5G system and MEC system respectively with focus on techniques and concepts that can be used for their seamless integration. This section aims to deliver an end-to-end framework for MEC and 5G integration based on standards by 3GPP and ETSI MEC. This is of vital importance for the rest of the thesis, but it can also be valuable for developers of MEC solutions.

The lifecycle of a MEC app can be divided into four stages, based on [16]:

- MEC app packaging and onboarding
- MEC app instantiation
- Client-side app and MEC app communication
- Usage of the MEC platform and services

MEC apps are expected to be packaged in Virtual Machines (VMs) or containers and to run on the virtualized infrastructure of MEC hosts. The onboarding of MEC apps is left as an implementation choice for the MEC operator [16]. The second stage of a MEC app lifecycle describes how it is instantiated on MEC system and how it is made available to clients. There are two options for MEC app initialization and, eventually, for establishment of communication between the client-side app and the MEC app. One option is via communication between the device application and the MEC system through a UALCMP, and the other is via the OSS without the need for any client-side action. The first option is related to edge-aware discovery, described in previous section. Device application contacts the UALCMP via the Mx2 interface and informs it which MEC app it would like to instantiate, the UALCMP then informs the MEO, which is responsible for choosing the appropriate MEC host and configuring the MEPM to initialize the given application. In response to successful instantiation, the device app receives the IP address of the running MEC app instance [16]. In the other option, the initialization is done without the client-side being required to participate in the process. For the client app, to leverage MEC, it simply accesses the required application via its domain name and the 5G system DNS is responsible for connecting the client to the nearest running MEC app instance [13, 16]. Once instantiated, MEC apps can leverage SBA within the MEC system, consuming services provided by the MEC platform or other applications and also offering their services to other applications.

The EC can be exploited in two ways, both of which end in UE accessing the desired application in an edge server to experience low latency. The first option, relying on DNS servers connecting clients to the nearest edge server, requires lower implementation effort at a cost of several limitations. It is less suited for highly mobile scenarios and highly distributed cloud [13]. The second option relies on tight integration of the MEC system and the underlying mobile network. As a result, it offers more support for highly mobile scenarios, even allowing context migration between MEC app instances in stateful applications [13]. Furthermore, MEC offers a framework for MEC apps, through which they can feed traffic rules to the mobile network to help satisfy QoS requirements. This is achieved through establishment of a dedicated PDU session steering traffic from the client app to the MEC host.

The resulting framework, utilizing edge aware app initialization and dedicated PDU session establishment is as follows:

1. Device app requests initialization of a MEC app from UALCMP
2. MEO chooses the optimal MEC host and sends traffic rules to the selected MEC platform
3. MEC platform, acting as an AF, requests traffic rules control from SMF, via NEF and then PCF
4. SMF configures the UPF with UL CL to steer desired traffic to MEC host

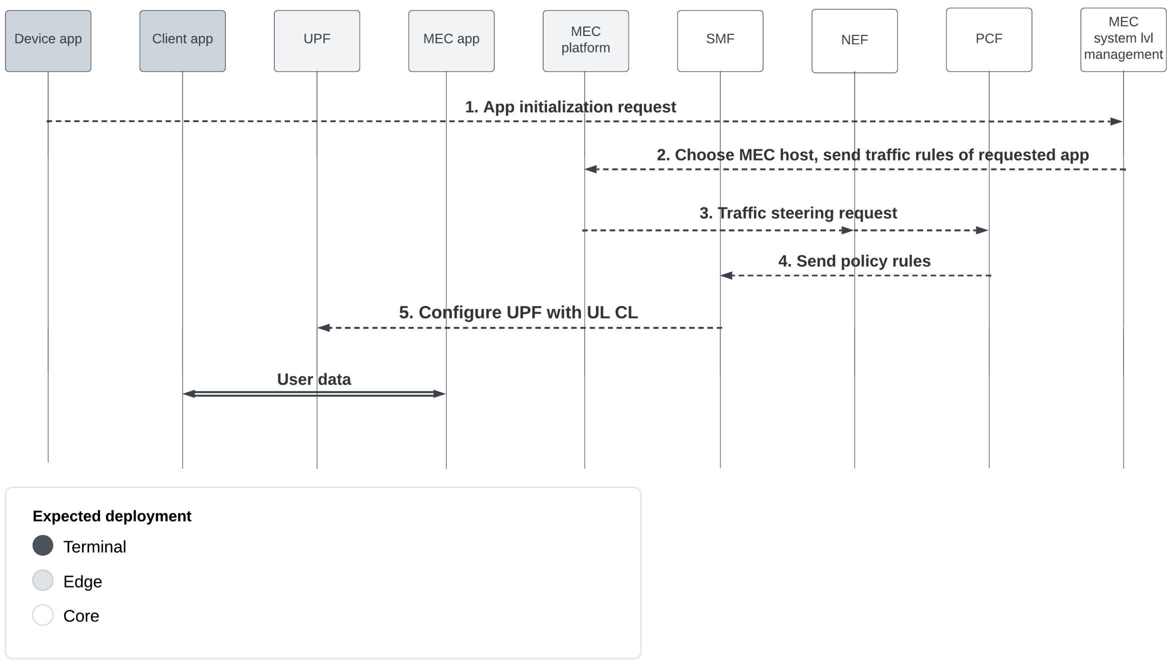

Device app
Client app
UPF
MEC app
MEC platform
SMF
NEF
PCF
MEC system lvl management
1. App initialization request
2. Choose MEC host, send traffic rules of requested app
3. Traffic steering request
4. Send policy rules
5. Configure UPF with UL CL
User data
Expected deployment
Terminal
Edge
Core

Figure 2.8. MEC integration in 5G

3. Open-source MEC Solutions

This chapter provides a comprehensive overview of the expanding ecosystem of open-source MEC solutions with the primary aim of selecting the most suitable one to leverage in the development of VEC testbed. The emergence of open-source MEC solutions stems from the necessity to accelerate the development of MEC applications and promote the adoption of MEC as a critical technology in 5G mobile networks. Consequently, this environment comprises of various MEC projects, developed by a range of stakeholders, addressing different challenges within MEC. This chapter intends to describe the open-source MEC ecosystem by introducing the main bodies taking part in MEC development, their respective projects and the challenges these projects aim to address.

The main selected criteria for evaluating MEC projects are which part of the ETSI MEC system they aim to constitute and what underlying technologies they use. Another important factor is interoperability support, i.e., which interfaces are provided in the project to enable communication with modules developed by different stakeholders. An end-to-end MEC solution is expected to be developed by three stakeholders: MEC app developer, mobile network operator and MEC operator. The evaluated open-source projects play a role of a MEC operator. Therefore, it is crucial that they provide means of integrating it with both MEC apps and mobile networks. Furthermore, all the evaluated projects are a work in progress, as they not yet provide all the desired functionalities or are fully operational. Therefore, some promising projects were tested and their state of operationality will be discussed.

Before evaluating the individual projects, it is desirable to mention the most often used software techniques leveraged in these projects. The most important one is virtualization. In the previous chapter, it was described how the architecture of both 5G network and MEC system consists of several building blocks called NEs or FEs. These building blocks are not implemented by specific hardware parts, instead, the trend is moving towards virtualization, i.e., managing multiple separated software-defined entities on general hardware. The way the separation is achieved sets apart different virtualization techniques. One way is to introduce a new abstraction layer on top of the hardware – the hypervisor, then the separation is created by having multiple VMs with their own operating system running on top of the hypervisor. Another way is to package the desired application and all of its dependencies in blocks called containers. All running containers share the same underlying operating system and are thus a significantly more lightweight solution than VMs.

Some popular hypervisors are VirtualBox or VMware and the most common platform for container virtualization is Docker. Orchestrating multiple containers on a single node is often handled by Kubernetes (k8s), a popular open-source container orchestration tool. Another key software, Open vSwitch, provides virtual networking layer for VMs or containers and enables their connectivity to each other and to data networks.

3.1 Akraino projects

The leader in development of open-source edge software is LF Edge, a Linux Foundation umbrella organization, which aims to unite industry leaders to introduce edge computing framework. LF Edge identifies three levels of maturity within their projects. Only two projects are, at the time of writing, listed in the top category: Akraino and EdgeX Foundry [?]. The latter, EdgeX Foundry, has established itself as the leading open-source industrial IoT edge framework. Akraino, on the other hand, spans over a broad variety of use cases, including but not limited to MEC. Akraino is not one common platform, but rather a whole set of edge stack solutions, called blueprints. Furthermore, in 2019 LF Edge signed a cooperation agreement with ETSI, which lead to an increasing number of Akraino blueprints tightly following ETSI ISG MEC specifications [4]. Under the newest – sixth release, Akraino constitutes of over 30 blueprints, such as: Connected Vehicles Blueprint (CVB), Enterprise Applications on Lightweight 5G Telco Edge (EALTEdge) or Public Cloud Edge Interface (PCEI) [?].

3.1.1 Connected Vehicles Blueprint

The CVB blueprint is developed by Tencent, Arm, Intel and Nokia. It aims to create a MEC platform offering services to connected vehicles. Some applications running on top of the provided platform include traffic situation monitoring, road event perception or high precision positioning. It references the ETSI MEC Mp1 and Mm5 interfaces, which are supposed to handle communication between MEC platform, MEC platform management and MEC apps. It is not uncommon among MEC solutions to leverage some related open-source software, CVB uses StarlingX as a virtualization platform, TARS as a microservices management tool and OpenNESS toolkit for managing and onboarding edge services. The setup of CVB requires three nodes: Jenkins master, TARS master and TARS node with the vehicle applications. An illustration of the scope of CVB is shown in Figure 3.1. [?]

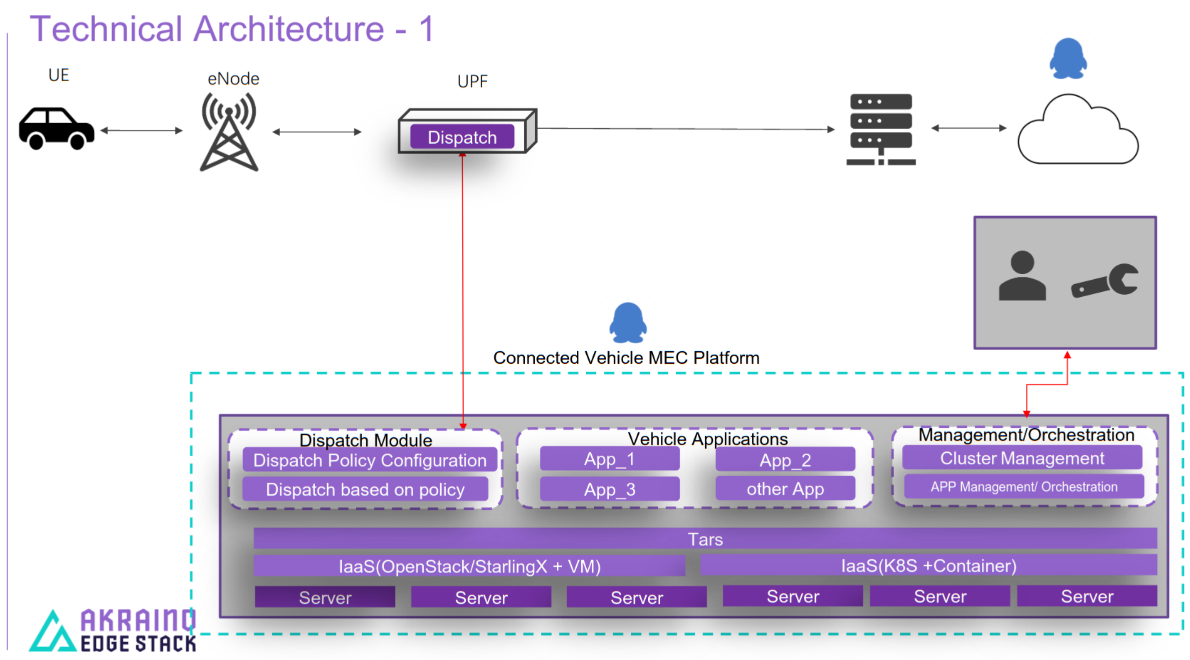


Figure 3.1. Technical overview of CVB, [?]

3.1.2 Enterprise Applications on Lightweight 5G Telco Edge

EALTEdge is a blueprint under 5G/MEC system blueprint family within the Akraino project. It is intended to play the role of a MEC operator through offering MEC platform as well as its management and orchestration to host enterprise applications on lightweight 5G telco edge. EALTEdge uses EdgeGallery as its upstream project, which provides edge platform, application management and platform for application developers. On top of EdgeGallery, EALTEdge intends to integrate other Akraino blueprints, such as Edge-Connector and EdgeGateway, to create a richer environment with even more capabilities. EdgeGallery provides several ETSI MEC APIs such as application enablement API as defined in ETSI MEC 011, it is also one of the projects listed in [s] as relevant MEC solutions. The general architecture of EALTEdge project, leveraging EdgeGallery is provided in Figure ??. [?]

The deployment of EALTEdge is quite flexible, allowing to deploy everything as containers in one node (bare metal or virtual) or it can be deployed in multiple, typically two nodes. Furthermore, the deployment is managed through provided ansible playbooks to simplify the whole process.

Because EALTEdge provides a general MEC system, which closely follows ETSI MEC specifications and also offers a development environment for MEC apps, it seems like an ideal platform for the testbed development. However, upon testing in lab environment, it was discovered that EdgeGallery servers forbid requests of downloading the tarballs containing the software. Therefore, while EALTEdge or even EdgeGallery alone seem like very promising projects, they could not be leveraged in testbed development.

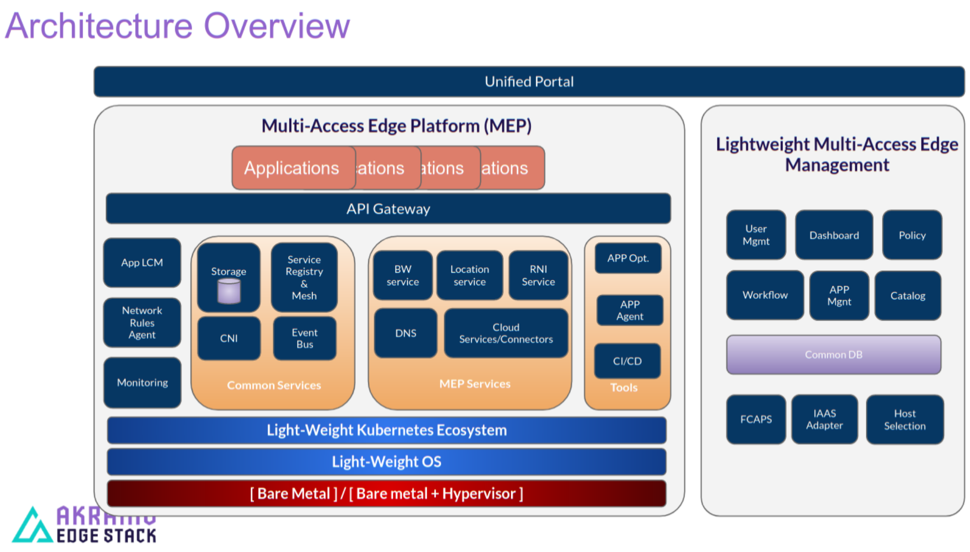
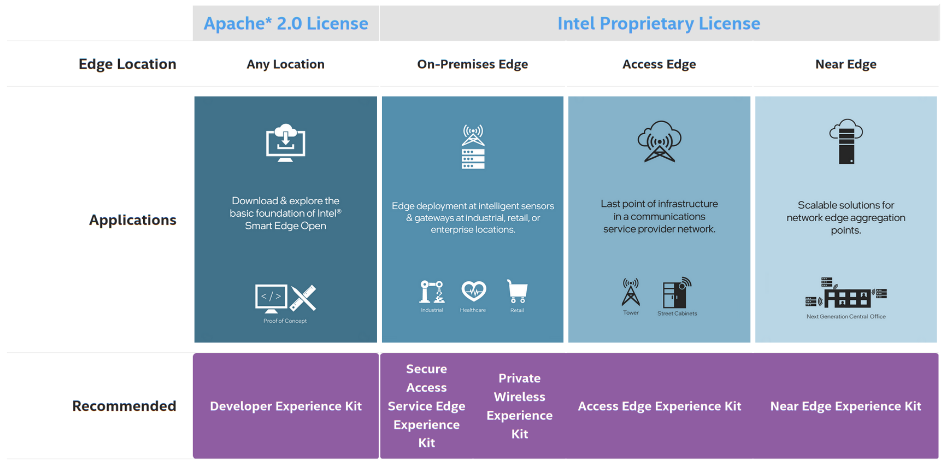


Figure 3.2. Technical overview of EALTEdge, [?]

3.2 Intel Edge Projects

Beside the Akraino project, another primary mover within edge software development is Intel. Intel develops commercial end-to-end edge solutions for their customers, but they also offer open-source variants for developers to prototype on. The range of edge-oriented products currently offered by Intel and the licenses under which they are released are shown in Figure 3.3. A successful open-source solution by Intel was Open Network Edge Services Software (OpenNESS), which at some point gained a lot of momentum within MEC open-source community, it is heavily reference in [4] and moreover, several Akraino blueprints reference OpeNESS as a platform they were initially built on [?, ?]. However, the OpeNESS project was shut down and transitioned towards Smart Edge Open. Both platforms will be introduced in the following two subsections.



	Apache* 2.0 License	Intel Proprietary License		
Edge Location	Any Location	On-Premises Edge	Access Edge	Near Edge
Applications	Download & explore the basic foundation of Intel® Smart Edge Open Proof of Concept	Edge deployment at intelligent sensors & gateways at industrial, retail, or enterprise locations. Industrial Healthcare Retail	Last point of infrastructure in a communications service provider network. Tower Street Cabinets	Scalable solutions for network edge aggregation points. Next Generation Central Office
Recommended	Developer Experience Kit	Secure Access Service Edge Experience Kit / Private Wireless Experience Kit	Access Edge Experience Kit	Near Edge Experience Kit

Figure 3.3. Intel edge solutions, [?]

3.2.1 OpenNESS

OpenNESS is a software toolkit that enables building custom platforms for edge computing use cases, including MEC, capable of onboarding and managing applications. Although OpenNESS did not strictly follow the ETSI MEC specifications, it was inspired by them and also references 3GPP and O-RAN specifications that were used to develop APIs for integration with 5G mobile networks [d]. It should be noted that Intel offered its own solution for 5G CN emulation, which was probably compatible with OpenNESS, though that was not part of the open-source package and was offered as a commercial product. [v]

OpenNESS was built on top of Kubernetes and highly optimized for edge computing. A typical deployment consists of an OpenNESS Kubernetes Edge Node and an OpenNESS Kubernetes Control plane, as shown in Figure 13. [v]

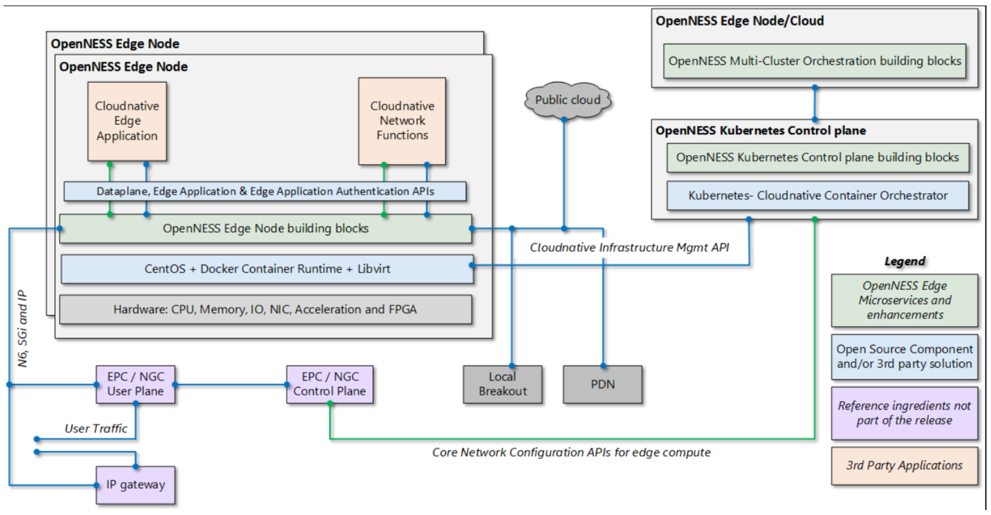

OpenNESS Edge Node
OpenNESS Edge Node
Cloudnative Edge Application
Cloudnative Network Functions
Dataplane, Edge Application & Edge Application Authentication APIs
OpenNESS Edge Node building blocks
CentOS + Docker Container Runtime + Libvirt
Hardware: CPU, Memory, IO, NIC, Acceleration and FPGA
Public cloud
OpenNESS Edge Node/Cloud
OpenNESS Multi-Cluster Orchestration building blocks
OpenNESS Kubernetes Control plane
OpenNESS Kubernetes Control plane building blocks
Kubernetes- Cloudnative Container Orchestrator
Cloudnative Infrastructure Mgmt API
N6, SGi and IP
EPC / NGC User Plane
EPC / NGC Control Plane
Local Breakout
PDN
User Traffic
IP gateway
Core Network Configuration APIs for edge compute
Legend
OpenNESS Edge Microservices and enhancements
Open Source Component and/or 3rd party solution
Reference ingredients not part of the release
3rd Party Applications

Figure 3.4. My PNG image

3.2.2 Smart Edge Open

Smart Edge Open (SEO) is a result of Intel's efforts to unify development of its open-source and commercial products. SEO follows the basics of OpenNESS architecture, as it is also built on top of Kubernetes and the basic deployment model consists of a control node and an edge node. [w] The resulting architecture is shown in Figure 14. Even though SEO development initially started as a rebranding of OpenNESS, the documentation of SEO is not as rich and it further distances itself from the ETSI MEC architecture. Eventually, while the transition from OpenNESS to SEO may have improved the platform's integration into Intel ecosystem, it distances itself from open-source MEC community, where strict following of standardized architecture is necessary to ensure interoperability between individual projects.

Even though it was argued that due to its distancing from strict ETSI MEC reference architecture, SEO is not an optimal solution for testbed development, it was still tested in our lab environment. However, the deployment of the platform was unsuccessful. SEO uses Intel's own provisioning software and while it should simplify the deployment, it makes the process of troubleshooting unfeasible, when an issue is encountered.

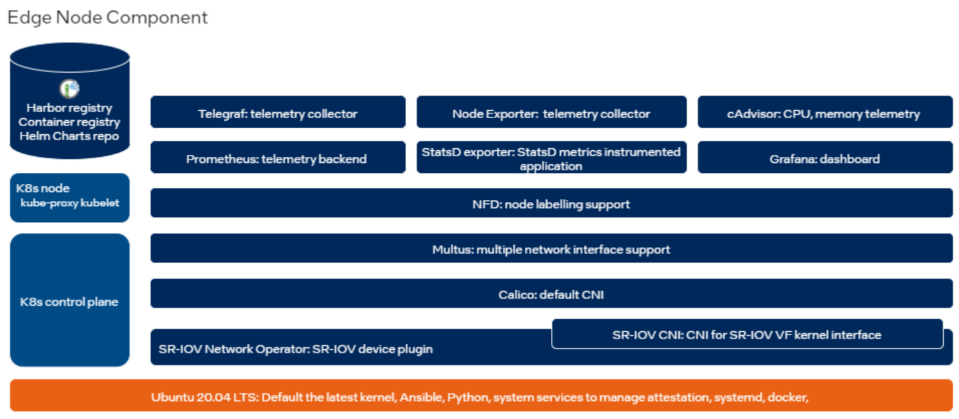

Edge Node Component
Harbor registry
Container registry
Helm Charts repo
K8s node
kube-proxy kubelet
K8s control plane
Telegraf: telemetry collector
Node Exporter: telemetry collector
cAdvisor: CPU, memory telemetry
Prometheus: telemetry backend
StatsD exporter: StatsD metrics instrumented application
Grafana: dashboard
NFD: node labelling support
Multus: multiple network interface support
Calico: default CNI
SR-IOV CNI: CNI for SR-IOV VF kernel interface
SR-IOV Network Operator: SR-IOV device plugin
Ubuntu 20.04 LTS: Default the latest kernel, Ansible, Python, system services to manage attestation, systemd, docker,

Figure 3.5. My PNG image

3.3 EURECOM and OpenAirInterface Projects

The two following projects have been developed by EURECOM and are closely related to OpenAirInterface (OAI), the mobile network emulation platform that is now maintained by OAI Software Alliance (OSA). EURECOM though continues to be involved in the development OAI and plays a key role in expanding the scope of OAI to network slicing or MEC.

3.3.1 LL-MEC

Low latency MEC (LL-MEC) was introduced in 2018 as the first open-source low latency MEC platform, enabling mobile network monitoring, control and programmability [y]. While it was developed in compliance with 3GPP and ETSI specifications, these specifications are not related to 5G MEC integration referenced in chapter 2, since LL-MEC was developed as a 4G MEC solution. One possible deployment of LL-MEC is shown in Figure 15, which also demonstrates that LL-MEC uses Open vSwitch for user plane. Since this thesis puts main focus on integrating MEC in 5G mobile networks, the LL-MEC has not been chosen as the platform for testbed development.

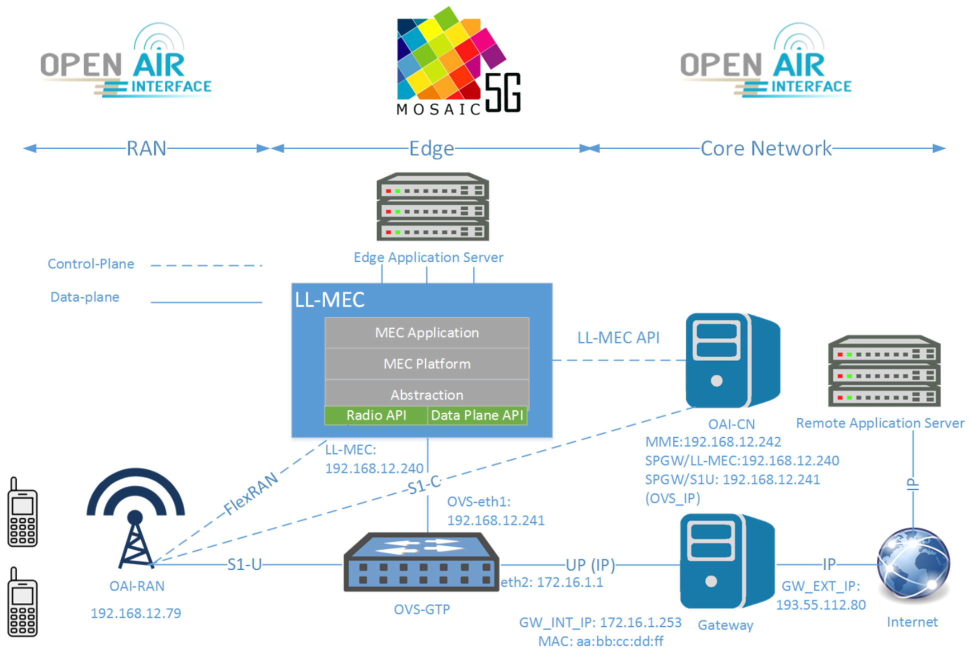


Figure 3.6. My PNG image

3.3.2 MEC platform by EURECOM

This MEC Platform (MEP) has been only recently developed by EURECOM and is closely integrated with OAI emulated 5G network. At the time of writing, the MEP is very much a work in progress as it only implements a MEP environment for registry and discovery of MEC services and offers its own implementation of Radio Network Information Service (RNIS) [x]. However, it has several traits that make it a very viable candidate for VEC testbed development. Its integration with the OAI would significantly reduce the integration implementation overhead. Moreover, it promises to follow ETSI MEC as well as O-RAN specifications. And finally, the developers have already outlined that it is intended to implement traffic rules control via communicating with PCF in 5G CN. This would mean that the project is aiming for a MEC implementation very close to the standardized approach extensively described in chapter 2. The components thus far developed are shown in Figure 16.

3.4 Other MEC projects

3.4.1 AdvantEDGE

AdvantEDGE is a mobile edge emulation platform running on Docker and Kubernetes. Its main ambition is to help MEC app developers decide where in the network edge their application should run and how different network characteristics such as latency, packet

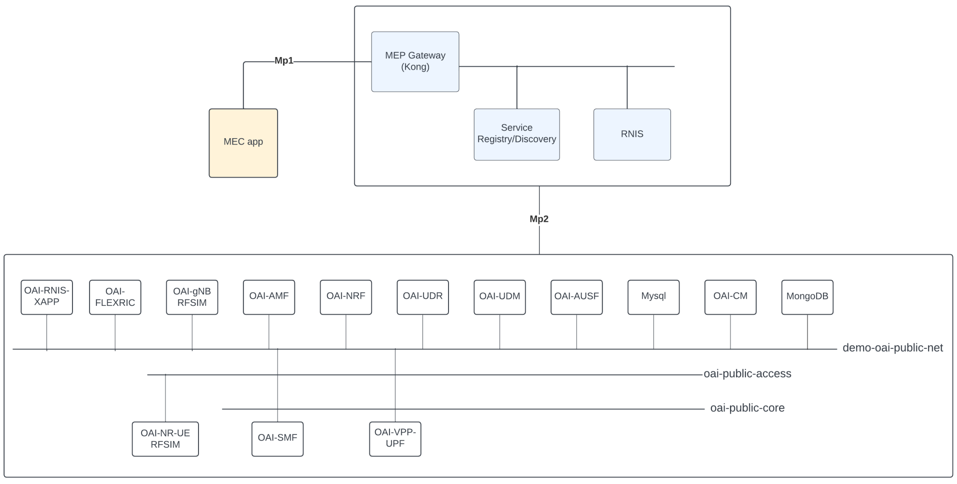


Figure 3.7. My PNG image

loss or jitter impact their application's QoS. This is achieved through emulation of mobile network with the possibility to test the application with different network characteristics settings and different deployment models (how close to the user the app is run). On top of that, the platform enables triggering of mobility events and application relocation even for stateful applications. The platform is not intended to emulate any part of the ETSI MEC reference architecture, but it implements various ETSI MEC APIs, such as RNIS, Location Service or App Enablement Service. The high-level overview of AdvantEDGE micro-service based architecture is shown in Figure 17.

AdvantEDGE was successfully deployed in our lab setup, however, it was deemed not suitable for the purposes of the testbed development, which puts a strong emphasis on the platform architecture and the underlying mobile network.

The following table sumamarizes the overview of examined open-source MEC projects.

Table 3.1. Evaluation of examined open-source projects

	ETSI MEC reference	open-source	successfully deployed
CVB	MEC platfrom	yes	no
EALTEdge	MEC system	yes	no
OpenNESS	MEC plaform + platform management	partially	no
SEO	edge server and management	partially	no
LL-MEC	4G MEC solution	yes	no
OAI MEP	MEC platform	yes	yes
AdvantEDGE	MEC APIs (RNIS)	yes	yes

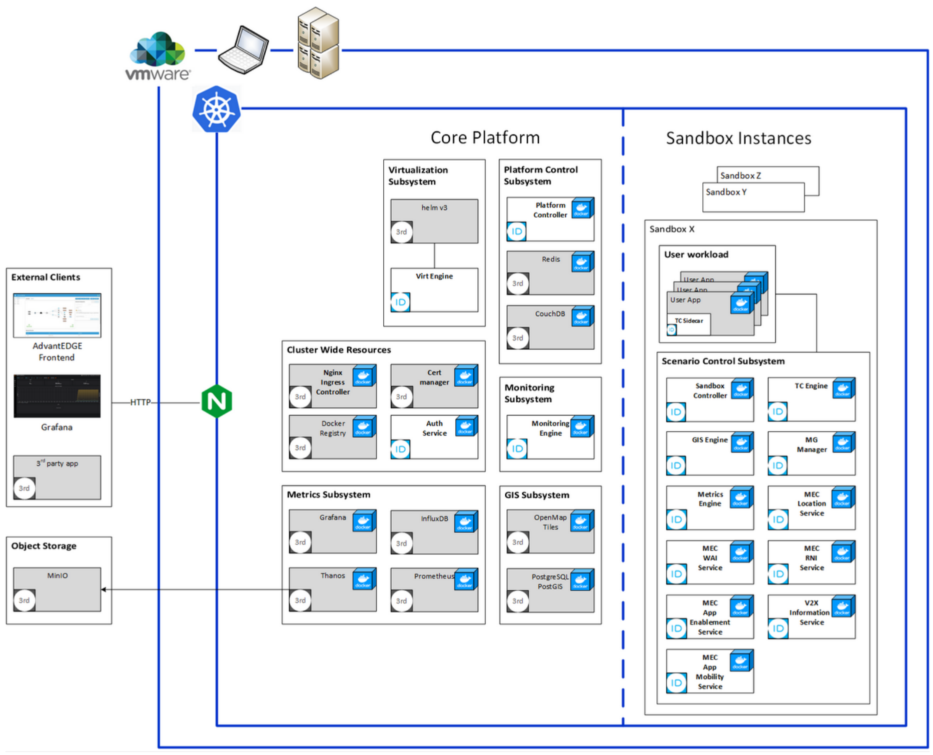


Figure 3.8. My PNG image

4. Mobile network emulation: OpenAirInterface

The goal of this chapter is to provide an overview of an open-source mobile network emulation platform OpenAirInterface (OAI). OAI has the ambition to democratize the research and development of wireless access technologies through delivering a 3GPP compliant software stack for both RAN and CN. The OAI software offers different levels of mobile networks simulation or emulation in flexible deployment scenarios and is intended to run on general purpose hardware [z]. OAI was originally founded by EURECOM, and its development is overseen by OSA, which has the goal of promoting the adoption of OAI across both academia and industry.

OAI development is split into different projects. The two biggest ones are focused on developing the solutions for RAN and CN. While those are definitely the core of OAI, its focus spans further and exploits smart solutions regarding mobile networks' orchestration and programmability under Mosaic5G project. The following sections offer a brief introduction to all of these projects and discuss how they are going to be leveraged in OAI based VEC testbed.

4.1 OAI RAN

OAI RAN offers modems for the UE and base station (eNB in 4G and gNB in 5G). These modems are developed with respect to 3GPP specifications and therefore, OAI RAN implements the entire 5G New Radio (5G-NR) protocol stack [aa]. This thesis is focused primarily on the CN architecture, therefore, the individual protocols of 5G NR will not be exploited in great detail. Below is an example of 5G NR user plane protocol stack with brief introduction of individual layers based on [a].

The three topmost layers are the application layer, the Transmission Control Protocol (TCP) and Internet Protocol (IP). This is common for any IP-based network, including 4G and 5G mobile networks.

The 5G NR user plane protocol stack:

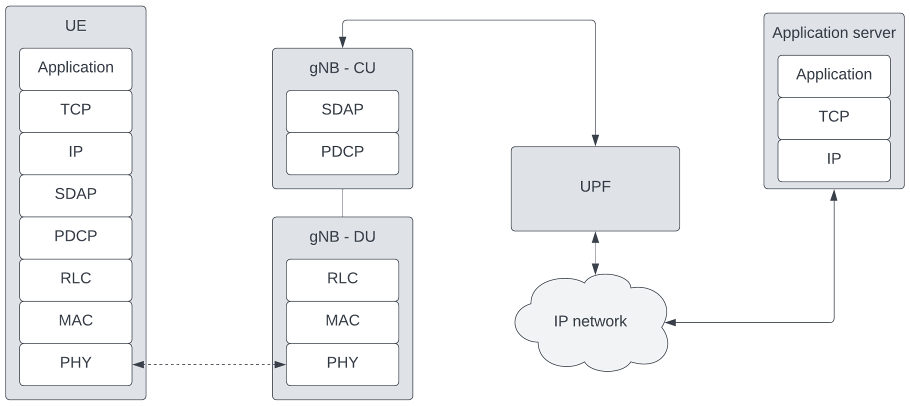


Figure 4.1. My PNG image

- Service Data Application Protocol (SDAP): responsible for mapping of QoS bearers to radio bearers, new layer in NR.
- Packet Data Convergence Protocol (PDCP): performs IP header compression, ciphering and integrity protection.
- Radio Link Control (RLC): segmentation and retransmission handling.
- Medium Access Control (MAC): multiplexing of logical channels, hybrid retransmission and scheduling.
- Physical Layer (PHY): coding and decoding, modulation and demodulation or multi-antenna mapping.

In control plane, the lower-level protocols are the same, but on top of PDCP there is RRC and NAS.

- Radio Resource Control (RRC): setup and release of radio bearers, mobility management, CP signaling.
- Non-Access Stratum (NAS): signaling messages between UE and CN for management of UE mobility and network selection.

For a better understanding of current OAI 5G NR capabilities, some general features based on [aa] are listed below in Table 4.1

OAI is designed to serve as a development tool for accurate mobile networks simulations and emulations. For this purpose, OAI enables different simulation models: RFsimulator, L2 nFAPI simulator and running with true radio heads. RFsimulator replaces the radio heads and allows connecting the UE to eNB or gNB via network interfaces. It is the optimal model to start testing OAI as it significantly decreases the deployment overhead.

Table 4.1. OAI RAN features

Downlink Throughput	1000 Mb/s
Uplink Throughput	100 Mb/s
Round Trip Time (RTT)	4 ms
Number of UEs	16
Duplexing	TDD, FDD
Bandwidths	10, 20, 40, 80, 100 MHz
Frequency Ranges (FRs)	FR1, FR2
Slot format	14 OFDM symbols
Subcarrier modulation	up to 256 QAM
MIMO support	yes, in downlink

L2 nFAPI simulator allows connecting the UEs and eNB or gNB via nFAPI (Network Functional API), thus shortcutting the PHY layer. Finally, OAI is designed to operate with true radio interface, using Customer-Off-The-Shelf (COTS) hardware, such as USRP (Universal Software Radio Peripheral) for the e/gNB and regular COTS UE. [aa]

The development of OAI 5G CN is an ongoing process, while evolution from LTE in 4G to NR in 5G seems to be complete with OAI enabling standalone 5G deployment, OSA defined the short-term goals of OAI as performance improvement or integration with smart RAN solutions developed within the Mosaic5G group. [dd]

4.2 OAI CN

OAI 5G CN project group develops open-source solution for standalone 3GPP compliant 5G CN that is capable of enabling various 5G use cases. The 5G CN components are continuously tested with open-source as well as commercial RAN solutions [bb]. The OAI 5G CN has been gradually developed from 4G EPC to the 5G CN architecture, extensively described in chapter 2. In fact, this work is still ongoing, with the current state-of-the-art shown in Figure 21. According to [bb], OAI 5G CN currently supports many of the key workflows described in chapter 2, such as UE connection, registration and session management. Network slicing is already partially supported, but a key interface for opening the CN to third parties, the AF, is not yet available. This is important from the perspective of integrating an open-source MEC system solution with OAI CN because AF should play a key role in providing the means of accessing the CN's SBA for MEC orchestrator or MEC platform.

The flexibility of OAI 5G CN design allows for different deployment models based on deployed NFs. In the minimalist example supporting basic session management procedures, AMF, SMF, NRF and UPF comprise the whole CN. The basic deployment model consists of the minimalist model and AUSF, UDM and UDR [bb]. UDR (Unified Data Repository) was not described in chapter 2, it is a NF providing storage and retrieval of

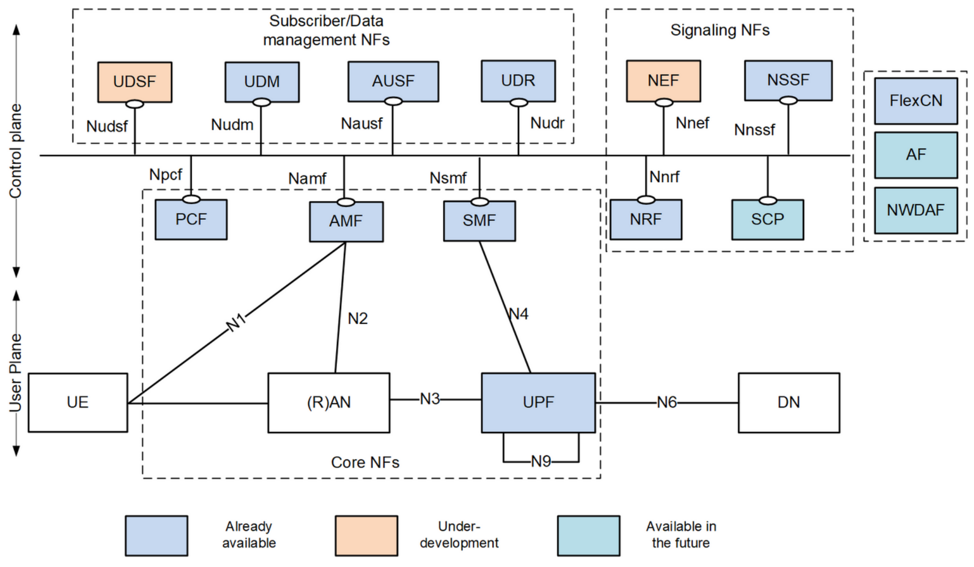


Figure 4.2. My PNG image

structured data for PCF, NEF and UDM [ee]. Finally, slicing deployment model utilizes the basic model with NSSF [bb].

As was the case with OAI 5G RAN project, OAI 5G CN is still being developed to offer new capabilities and enable new use cases. During 2022, OAI released PCF and currently works on integrating PCF induced traffic steering. This is a key part of the MEC integration described in chapter 2.

4.3 Mosaic5G

Mosaic5G is a project group within OAI focused on transforming the underlying OAI RAN and CN into agile platforms providing ecosystem for use cases ranging from centralized network control to edge deployments [dd]. The primary focus of Mosaic5G project group can be dissected into four main projects as defined in [dd]:

- The O-RAN E2 protocol – E2 agent
- A Flexible RAN Intelligent Controller (FlexRIC)
- A Flexible Core Controler (FlexCN)
- Higher layer RAN and CN operator – Tirematics

Currently, most of the Mosaic5G project group's work is centered around FlexRIC and its integration with the rest of the OAI software [dd]. In fact, the OAI MEP discussed in previous chapter also utilizes FlexRIC as a key component [x].

5. VEC testbed

The final chapter describes the implementation of testbed for VEC as the primary goal of the thesis. The structure of the following text is as follows: below, the whole task is dissected into three key stages, then each stage is examined in individual subsequent sections.

Stage 1: VEC testbed design leveraging chosen open-source projects

Stage 2: creating a blueprint for future VEC applications development

Stage 3: evaluation of set up testbed against cloud deployment

5.1 VEC testbed design

The final testbed design leverages the OAI MEP, introduced in chapter 3. While there, the focus was primarily on evaluating it against other MEC projects in terms of network architecture, this section describes the individual components from developer's perspective. OAI MEP does not comprise the whole MEC system, but it provides an ETSI compliant MEP with a service-based environment, fully compatible with the underlying OAI RAN and CN. Such setup is sufficient for experimentation with VEC applications, such as task offloading.

The leveraged platform is a blueprint showcasing OAI MEP with RNIS and it integrates five key projects as shown in Figure 22 [gg]:

- OAI 5G RAN
- OAI 5G CN
- OAI MEP
- OAI Configuration Manager (OAI-CM)
- RNIS

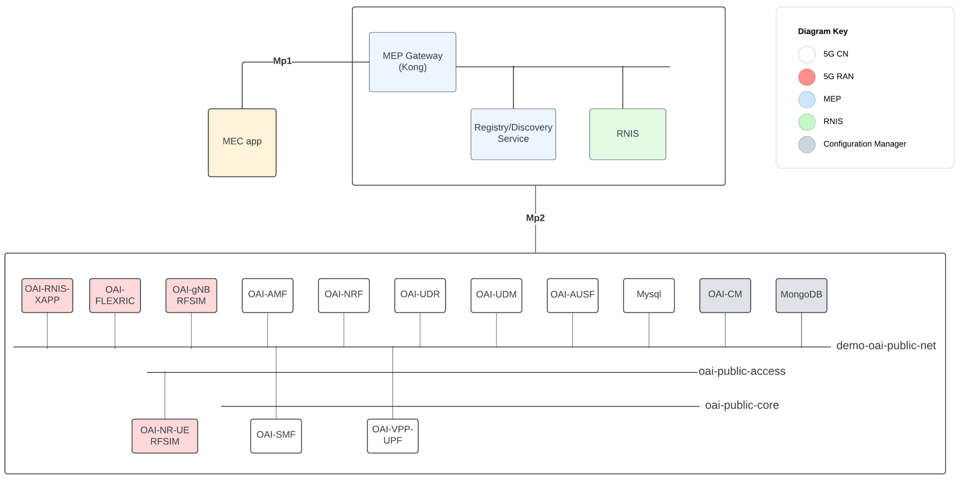


Figure 5.1. My PNG image

This blueprint showcases how MEC apps can consume services, such as RNIS, via the Mp1 interface [gg]. For RNIS to be able to consume mobile network Key Performance Indicators (KPIs) via Mp2 and then expose them via Mp1 to MEC apps, the OAI mobile network required new features to be implemented. The OAI 5G CN is run in a basic deployment, as shown in Figure 22, but the RAN leverages an O-RAN xApp and Mosaic5G FlexRIC on top of the modules for UE and gNB. The xApp provides a northbound interface for the RNIS to subscribe to radio related KPIs [ff]. The OAI configuration manager then provides the functionality to subscribe to AMF and SMF events [hh].

A key component of the OAI MEP is the MEP Gateway, which acts as a reverse proxy, routing the Mp1 traffic to the right service [x]. The reverse proxy is implemented leveraging Kong. It hides the underlying service nodes from the clients and routes client's requests to the right service nodes. This means that all registered services can be accessed via the MEP Gateway.

The Discovery and Registry Service has the responsibility of exposing REST API for the Mp1 interface allowing the MEC apps to do the following, based on [x]:

- Discover existing services (GET)
- Filter hosted MEC services based on service type (GET)
- Register a new service (POST)
- Delete a hosted MEC service (DELETE)

The OAI MEP blueprint delivers docker-compose scripts for the deployment of all the containers. Furthermore, the basic deployment uses rfsim, although [gg] claims that it is possible to run with USRP radio heads and that it was tested in their lab.

After a successful deployment, it should be possible to ping the internet from the UE container via its `oaitun_ue1` interface. In our setup, it looks like this:

```
root@9929963b00c2:/opt/oai-nr-ue# \
ping -c 3 -I oaitun_ue1 comtel.fel.cvut.cz \
PING glumhosting.fel.cvut.cz (147.32.210.146) from 12.1.1.2 oaitun_ue1:
56(84) bytes of data.
64 bytes from glum-146.feld.cvut.cz (147.32.210.146):
icmp_seq=1 ttl=59 time=8.64 ms
64 bytes from glum-146.feld.cvut.cz (147.32.210.146):
icmp_seq=2 ttl=59 time=5.43 ms
64 bytes from glum-146.feld.cvut.cz (147.32.210.146):
icmp_seq=3 ttl=59 time=5.06 ms

--- glumhosting.fel.cvut.cz ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 5.062/6.382/8.649/1.610 ms
```

To verify that the RNIS works properly, it should be possible to access its service with the HTTP GET method from the Command Line Interface (CLI). This is the result achieved in our setup:

```
lab@lab-HP-Z4-G4-7-Workstation:~/blueprints/mep/examples$ curl -X \
'GET' 'http://oai-mep.org/rnis/v2/queries/layer2_meas' -H \
'accept: application/json'
[
{
"KPIs": {
"bler_dl": {
"kpi": "bler_dl",
"labels": {
"amf_ue_ngap_id": 1
},
"source": "RAN",
"timestamp": 1683903896587163,
"unit": null,
"value": 5.605193857299268e-45
...
```

5.2 VEC application blueprint

This stage aims to deliver a blueprint for development of VEC applications running on the OAI MEP and consuming its services. It is split into two subsections, with the first one covering the service registration and the second one demonstrating implementation of a MEC app consuming services offered by the OAI MEP.

5.2.1 Service registration

The registration and discovery service of OAI MEP can be accessed via its Graphical User Interface (GUI) on the host 192.168.70.5 or via the MEP Gateway:
`http://oai-mep.org/service_registry/v1/ui`. It should be noted that for the latter to work, the oai-mep.org domain needs to be added to the local machine DNS, which is documented in [gg]. The GUI offers the services mentioned in the previous section. After a successful deployment, the discovery of available services should yield the following results, as only the RNIS is registered:

```
lab@lab-HP-Z4-G4-7-Workstation:~/blueprints/mep/examples$ curl /
http://oai-mep.org/service_registry/v1/discover
[
{
"description": "The service allows other app or services to consume
network and radio information",
"endpoints": [
{
"method": "GET",
"name": "rab_info",
"path": "/queries/rab_info"
},
...
],
"host": "oai-mep.org",
"name": "rnis",
"path": "/rnis/v2",
"port": "80",
"sid": "6458bfb3aab751de1b2ba9f0",
"type": "RadioNetworkInformation"
  },
```

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def root():
    return "<h1>VEC testbed root page</h1>"

@app.route('/hello')
def hello():
    return '<h1>Hello world page</h1>'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000,
       use_reloader=False)
```

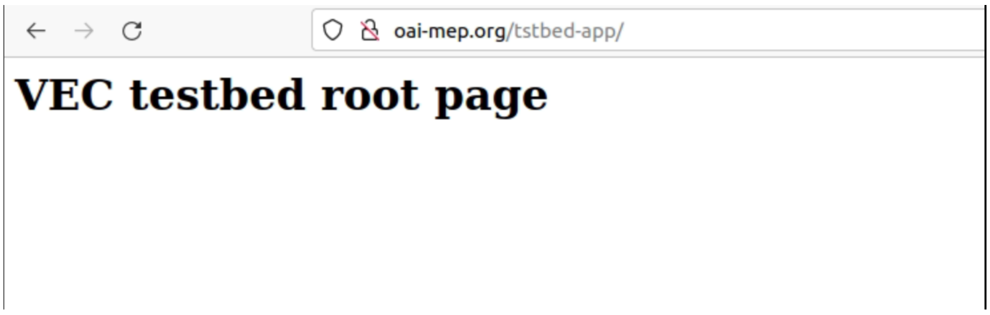
Listing 5.1. Sample flask app in python

To demonstrate the service registry and discovery function of OAI MEP, a sample hello-world flask application has been created. This application runs on the localhost interface of the host machine and displays a simple line of text, the goal is to be able to route this application via the MEP Gateway.

To register the flask application, the GUI of the registry and discovery service has been leveraged. While the GUI provides a sample input for the register service, it does not include any uid, which confuses the underlying database of OAI MEP and results in unsuccessful registration. For a successful registration, the input shown below has been used.

```json
{
"description": "testing registration with sample flask",
"endpoints": [],
"host": "tstbed-app.org",
"name": "tstbed-app",
"path": "/",
"port": 5000,
"sid": "sid",
"uid": "tstbedUID",
"type": "Traffic"
}
```

Successful registration is indicated by the code 201 and a response with json data about the newly registered service, then sample flask application can be routed via the MEP Gateway, as demonstrated below.


oai-mep.org/tstbed-app/

VEC testbed root page

Figure 5.2. My PNG image

5.2.2 MEC app consuming services

This subsection demonstrates how to create an application that would consume the services offered by OAI MEP based on an example app provided in [gg]. The example app is a flask application that subscribes to RNIS and shows the received data. The core of the application with minor changes from the version in [gg] is shown in Listing 5.2.

It is noteworthy that the example app uses the reverse proxy OAI MEP Gateway to subscribe to RNIS as shown on line 3:
`sub_endpoint = http://oai-mep.org/rnis/v2/subscriptions`. Then in `sub_body`, an endpoint for receiving the data from RNIS is provided, and finally, under that endpoint, the `receive_notification()` function handles the received data and puts them into `collected_data` dictionary.

5.3 Evaluation of VEC testbed

To evaluate the VEC testbed, an experiment has been conducted, measuring the RTT between the UE and the OAI MEP host, against the RTT between the UE and a cloud. The diagram illustrating the experiment is shown in Figure 5.3. It does not account for the UE and gNB modules also being hosted on the edge server, although the experiment could be scaled with using radio heads and both UE and gNB modules in standalone machines. The deployment in our lab setup follows the deployment model collocating edge server (MEC host) with UPF [i].

The experiment consisted of running ping from UE's `oaitun_ue1` interface to both the edge and cloud. The measurement was issued every two hours throughout one day and the results are shown in Table 5.1 and visualized in Graph 5.1.

The data show that the average RTT difference between edge and cloud is 7.94 ms. This follows the expectation and demonstrates the feasibility of using OAI MEP setup for VEC testbed.

```python
...
# create a subscription for receiving l2 measurements
sub_endpoint = "http://oai-mep.org/rnis/v2/subscriptions"
sub_body ={
"callbackReference": f"http://{ip_addr}:{port}/subscriptions/l2meas-200",
"filterCriteriaNrMrs": {},
"subscriptionType": "NrMeasRepUeSubscription",
"expiryDeadline": {
    "nanoSeconds": 12133423,
    "seconds": 20
}
}
requests.post(url =sub_endpoint, json=sub_body)
...
# collect received data
@app.route('/subscriptions/l2meas-200', methods=[ 'POST'])
def receive_notification():
    if request.method == 'POST':
        content = request.get_json(force=True)
        print(content)
        kpis = content["Report"]
        for aid in content["AssociateId"]:
            if aid not in collected_data:
                collected_data[aid] = {}
            for kpi_name, kpi_data in kpis.items():
                collected_data[aid][kpi_name] = kpi_data['value']
    return "OK"

# show kpis
@app.route('/dashboard', methods=['GET'])
def dashboard():
    return "<h1> Network Monitoring Dashboard</h1> " + json.dumps(collected_data)
...
```

Listing 5.2. example-app.py

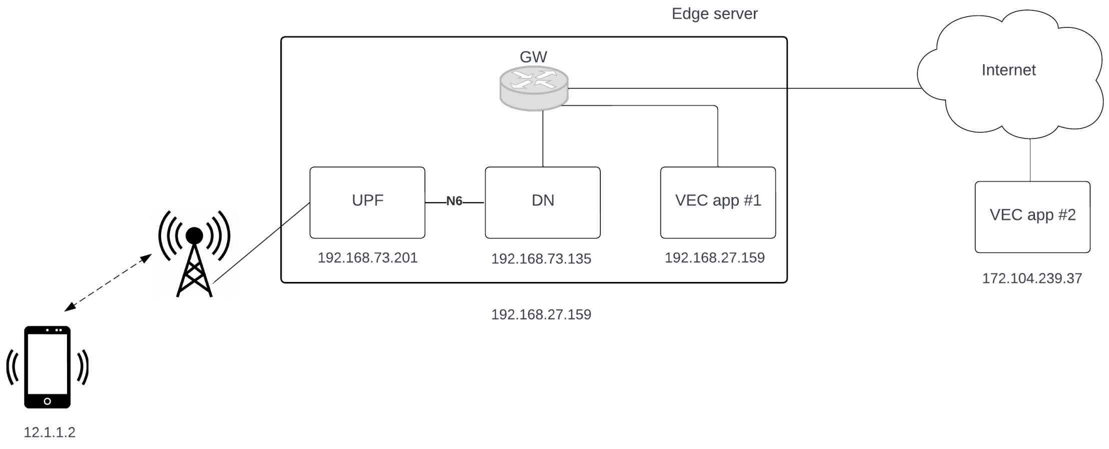


Figure 5.3. My PNG image

Table 5.1. RTT measurement

	Edge [ms]	**Cloud [ms]**
1:00	5.810	13.137
3:00	5.848	13.531
5:00	5.694	13.559
7:00	5.545	14.311
9:00	5.311	13.079
11:00	5.969	13.380
13:00	5.809	13.623
15:00	5.066	13.458
17:00	5.792	13.589
19:00	5.333	13.403
21:00	5.507	13.410
23:00	5.335	13.868

The second experiment evaluated the network performance and available bandwidth with iperf. This experiment deserves special notice. As mentioned in Chapter 4, the total achievable network throughput in RFsimulator mode depends on the underlying hardware, the actual numbers are, therefore irrelevant, as they scale with the hardware. The throughputs achieved for both edge and cloud showed no significant difference.

Table 5.2. Throughput measurement

Edge		Cloud	
Downlink [Mbps]	Uplink [Mbps]	Downlink [Mbps]	Uplink [Mbps]
9,09	7,18	9,01	7,195

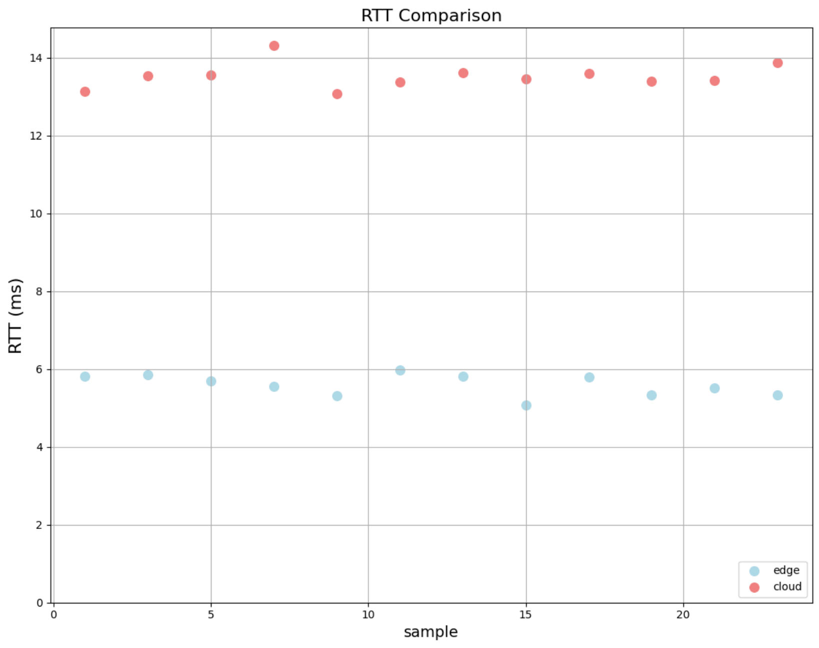


Figure 5.4. My PNG image

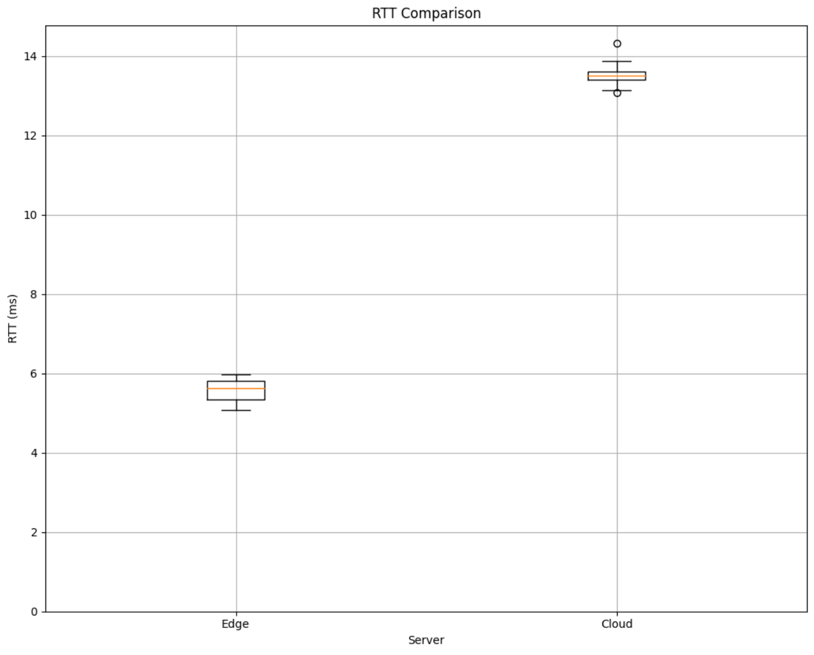


Figure 5.5. My PNG image

6. Conclusion

The thesis's objective was to set up a functional testbed for future research on VEC based on standards developed for MEC. To facilitate these objectives, the standards regarding MEC and 5G have been examined and a framework for their integration has been proposed. Then, existing open-source solutions for MEC or VEC have been evaluated with the purpose of selecting the most suitable platform for leveraging in VEC testbed. The selected platform has been thoroughly examined, tested and evaluated against realistic cloud deployment scenario.

The set up testbed leverages the OAI MEP project. It is built on OAI 5G RAN and CN with additional components that ensure exposing underlying mobile network information to implemented RNIS. The MEP itself acts as a gateway for MEC apps to consume MEP registered services. The 5G NFs and MEC FEs in OAI MEP run as Docker containers in RFsimulator simulation model with the possibility of extending the simulation to use COTS hardware for the radio interface. To enable seamless extension of the testbed and future work, the stable OAI MEP code has been forked to a new VEC testbed repository and a framework for registering new MEC service and consuming registered services has been provided.

The testbed has been evaluated against cloud deployment to assess the achieved reduction in communication latency. The edge deployment model employed in our lab has the edge server collocated with UPF, emulating a common approach. The evaluation results clearly demonstrate a significant decrease in RTT, approximately 59 % when utilizing an edge deployment compared to the cloud alternative. This establishes the testbed's viability as a research tool to evaluate future VEC applications.

In future, the testbed should be updated with future releases of the underlying OAI MEP, which promises to further increase the mobile network programmability through MEP inflicted traffic control via PCF. Furthermore, the testbed could be expanded to work with real radio interface to provide even more accurate evaluations. The platform now enables development of VEC applications leveraging task offloading for various use cases such as remote image processing for AVs. [TOWARDS 6G]

References

[1] M. Sauter, *From GSM to LTE-Advanced Pro and 5G: An Introduction to Mobile Networks and Mobile Broadband.* John Wiley & Sons, Incorporated, 2017.

[2] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for mobile broadband.* Academic press, 2013.

[3] ——, *5G NR: The next generation wireless access technology.* Academic Press, 2020.

[4] D. Sabella, *Multi-access edge computing: Software development at the network edge.* Springer, 2021.

[5] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing.(sept. 2011)," *URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf*, 2011.

[6] S. Dahmen-Lhuissier, "Multi-access edge computing (mec)." [Online]. Available: https://www.etsi.org/technologies/multi-access-edge-computing

[7] H. Zhao, Y. Zhang, P. Meng, H. Shi, L. E. Li, T. Lou, and J. Zhao, "Towards safety-aware computing system design in autonomous vehicles," *arXiv preprint arXiv:1905.08453*, 2019.

[8] F. Giust, G. Verin, K. Antevski, J. Chou, Y. Fang, W. Featherstone, F. Fontes, D. Frydman, A. Li, A. Manzalini *et al.*, "Mec deployments in 4g and evolution towards 5g," *ETSI White paper*, vol. 24, no. 2018, pp. 1–24, 2018.

[9] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "Mec in 5g networks," *ETSI white paper*, vol. 28, no. 2018, pp. 1–28, 2018.

[10] S. Rommer, P. Hedman, M. Olsson, L. Frid, S. Sultana, and C. Mulligan, *5G Core Networks: Powering Digitalization.* Academic Press, 2019.

[11] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "Nfv and sdn—key technology enablers for 5g networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2468–2478, 2017.

[12] "ETSI TS 123 501 V15.2.0, "5G; System Architecture for the 5G System"," ETSI, 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.02.00_60/ts_123501v150200p.pdf

[13] N. Sprecher *et al.*, "Harmonizing standards for edge computing-a synergized architecture leveraging etsi isg mec and 3gpp specifications," *ETSI White paper*, no. 36, 2020.

[14] "ETSI GS MEC 003 V3.1.1, "Mobile Edge Computing (MEC); Framework and Reference Architecture"," ETSI, 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_MEC003v030101p.pdf

[15] "ETSI GS MEC 011 V3.1.1, "5G; System Architecture for the 5G System"," ETSI, 2022. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/011/03.01.01_60/gs_mec011v030101p.pdf

[16] D. Sabella, V. Sukhomlinov, L. Trang, S. Kekki, P. Paglierani, R. Rossbach, X. Li, Y. Fang, D. Druta, F. Giust *et al.*, "Developing software for multi-access edge computing," *ETSI white paper*, vol. 20, pp. 1–38, 2019.