University of Stavanger

GIA HOANG
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Pre-study on Multi-access Edge Computing at Communication Technology lab: simulator/emulator

Bachelor's Thesis - Computer Science - May 2022

Contents

Abstract

Multi-access edge computing has been on the rise since the evolution of 5G. There are challenges that 5G has been fighting with, such as latency and data being transferred. In this paper, it will talk about the background of Multi-access edge computing, the state of the art of Multi-access edge computing research and then implementing a simulator to demonstrate Multi-access edge computing functionalities.

Chapter 1

Introduction

Wireless communication networks has become integral to our lives. The few decades of the evolution of wireless communication networks have been through the form of generations of cellular networks called generations, and currently the Fifth Generation (5G) is being deployed. The wireless communication networks have made communication with other people and transferring data very efficient. Instead of waiting for mail to arrive to your mailbox, you can get emails sent to your email box almost instantly. There are always improvements to be made for wireless communication networks, looking to satisfy requirements to improve the *quality of user experience (QoE)* [4]. One of these improvements to the wireless communication networks is a new technology that has been proposed.

Multi-access Edge Computing (MEC) as the term suggests, consists of multiple devices that can compute and are at the edge of the network, which means it is closer to the end users at the edge of the network [18]. There were other technologies such as fog computing that also proposed to improve wireless communication networks [1].

A simulation is generating numbers, it is a mathematical abstract. It can be a model or a collection of models to not make it too complex. A simulation only represent a few elements.

Emulation is used to abstractly send packets out like a computer, however it is in a virtualized environment so it is not experimentation. It can be software such as Packet Tracer.

Experimentation uses actual hardware in real life, such as in a lab. Actual hardware are items such as a physical computer.

There are challenges that surrounds wireless communication networks for 5G. Challenges such as latency, bandwidth, data, storage, computation and performance that need to be solved. High latency that makes it slow to get responses. Low bandwidth that makes data transmission low. Computation of the data and how to store the data. These issues that wireless communication networks have can be solved with solutions like MEC. MEC can lower the latency, use bandwidth to its maximum potential, compute and store the data.

There has been surveys on different MEC simulators and emulators, with most of them only talking about one simulator or emulator at a time. This paper have the following contributions:

- State of the art and MEC simulators and emulators.
  - Related works of the MEC simulators and emulators.
  - Comparisons between MEC simulators.
  - Comparisons between MEC emulators.
- Implementing simulator/emulator and then do tests on the simulator/emulator to demonstrate MEC.

The next chapter will talk about background information and definitions of MEC that is necessary to understand the MEC concept. Chapter 3 will talk about the state of the art and about MEC simulators/emulators. Chapter 4 will talk about implementing and testing the simulator/emulator. Chapter 5 will be the conclusion.

Chapter 2

Background

This chapter will talk about MEC history that describes how it came to be, Fifth Generation that is an evolution of telecommunications, MEC architecture, MEC service composition and MEC vs fog computing.

2.1 MEC background

MEC is a term first coined by ETSI and used to be Mobile Edge Computing, but it was later changed to Multi-access Edge Computing in 2017 [5]. MEC is a new technology that was being standardized in the ETSI *Industry Specification Group (ISG)*. The aim is to reduce latency, which will make the network operation and service delivery more efficient, and result in a better user experience. In order to do this, MEC will be deployed at the edge of the mobile network, but within the *Radio Access Network (RAN)* and close to the mobile subscribers. RAN is what connects end-user devices such as smartphones to the cloud through the use of radio waves. MEC is recognized by the research body European 5G PPP (5G Infrastructure Public Private Partnership) as an emerging technology for 5G networks. MEC is one of the technologies that will enable 5G for a better user experience [3].

2.1.1 The Fifth Generation

The Fifth Generation (5G) is an evolution to the Fourth Generation (4G). 5G will have faster speed than 4G. While 4G's Data Bandwidth can go from 2 Mbps to

1Gbps, 5G can go 1Gbps and higher. The difference in the Frequency Band is also very high, with 4G having 2 to 8 GHz and 5G having 3 to 300 GHz [2]. The upgrade to 5G has provided many services such as *enhanced Mobile Broadband (eMBB)*, which is used for services that has a high data rate requirement that goes up to 200Gb/s. *massive Machine-Type Communication (mMTC)*, which was developed for connecting a lot of Internet-of-Things (IoT), up to 1 million devices/km. *Ultra-Reliable Low-Latency Communication (URLLC)*, for services that need low latency up to 1ms and being very reliable. With the development of 5G, URLLC can aqcuire MEC, which will enable delivering services at low latency and offload tasks. MEC will help evolve the 5G through transforming the mobile broadband network into a programmable world and as such will contribute to the requirements of 5G [3].

2.2 MEC architecture

MEC uses a virtualized platform, and can be combined with *Network Functions Virtualization (NFV)*. NFV is the principle of separating network functions from the hardware they are on by using virtual hardware abstraction [13]. NFV is focused on network functions, but the MEC framework will enable applications at the edge of the network. Infrastructures that hosts MEC and NFV will allow the operators benefit from their investement in the infrastructure by allowing to host both *Virtual Network Functions (VNF)* and MEC applications on the same platform [3]. A VNF is the implenetation of *Network Function (NF)* that can be deployed on a *Network Function Virtualization Infrastructure (NFVI)* [13]. A NFVI is all of the hardware and software components that make the environment where VNFs are deployed [13].

MEC has three different groups of reference points between the system entities:

- MEC platform functionality reference points (Mp)
- Management reference points (Mn)
- External entities reference points (Mx)

There are different MEC related terminology that is used to describe MEC.

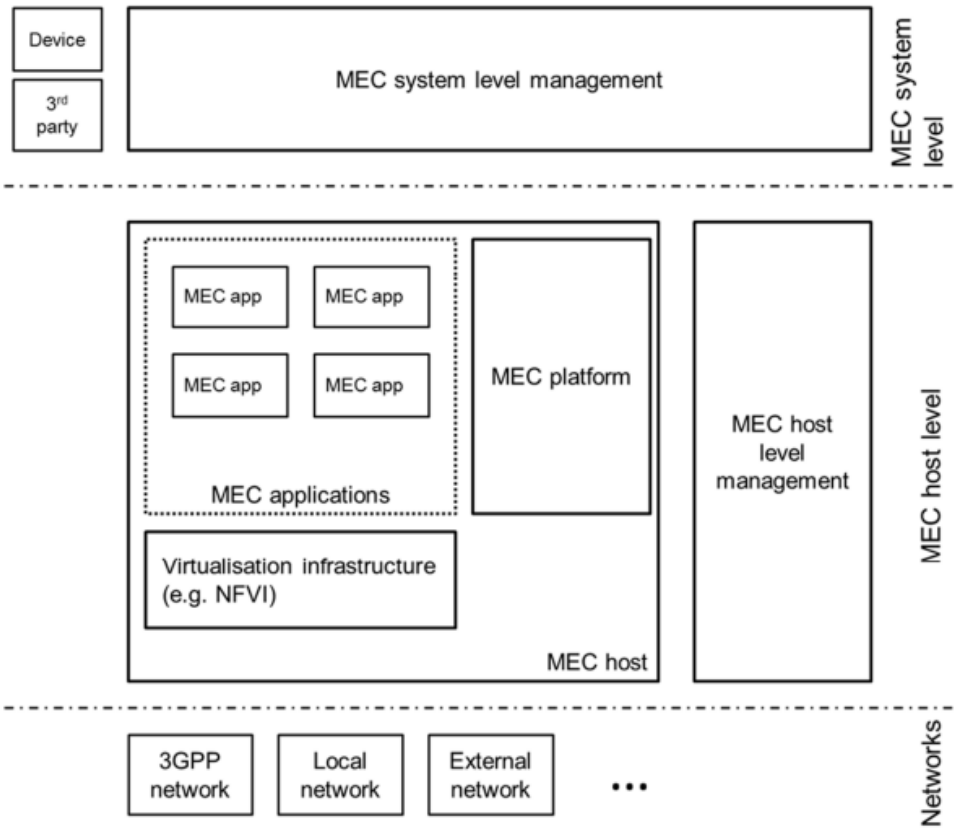
- The MEC system has MEC hosts and MEC management to be able to run MEC applications [9].
- A MEC application is an application that be instantiated on a MEC host be used to make or consume MEC services [9].
- A MEC host is an entity that has a MEC platform and a virtualization storage which gives the MEC applications the ability to compute, storage and network resources [9].
- A MEC platform is a collection of functionality that is required to run the MEC applications on a specific MEC host virtualization infrastructure. This will give the MEC applications on the MEC host to be able to make and use MEC services while also give itself MEC services [9].
- A MEC management is MEC system level management and MEC host level management. A MEC host level management consists of components that handle the management of MEC specific functionality of a MEC platform, MEC host and the MEC applications on it [9].
- A MEC system level management is the management components that has the overview of the complete MEC system [9].
- A MEC service is a service that is provided by the MEC platform, either from the MEC platform itself or from a MEC application [9].

The MEC system level management uses the *Multi-access Edge Orchestrator (MEO)* as the core component and has a complete overview of MEC system as stated above [12]. The MEC host level management consists of *MEC Platform Manager (MEPM)* and *Virtualization Infrastructure Manager (VIM)* and it handles the management of MEC specific functionality of a MEC host and the applications that are on it. A VIM is a functional block that is responsible for controlling and managing the NFVI compute, storage and network resources [13].

A MEC architecture variant is MEC in NFV. The MEC in NFV has a few assumptions for the architecture.

- The MEC platform is deployed as a VNF [12].

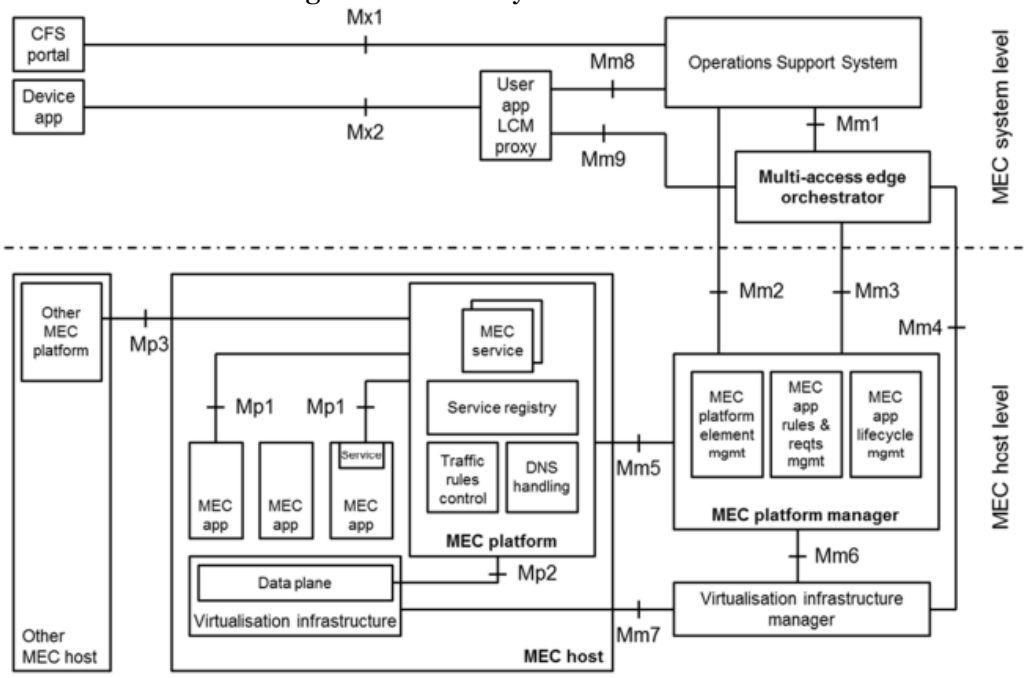
Figure 2.1: MEC framework



- The MEC applications will appear as VNFs for the ETSI *Network Functions Virtualization Management and Orchestration (NFV MANO)* components [12]. NFV MANO is functions that is collectively given by *Network Functions Virtualization Orchestrator (NFVO)*, *Virtualized Network Function Manager (VNFM)* and VIM [13]. A NFVO is a functional block that manages the *Network Service (NS)* lifecycle and coordinates the management of NS lifecycle, the VNF lifecycle, which is supported by VNFM, and the NFVI resources, which is supported by the VIM. This is so that the NFVO can ensure that the necessary resources and connectivity will be allocated optimally [13]. A NS is a composition of NF and NS, which is defined by its functional and behavioral specification [13]. A VNFM is a functional block that is responsible for the lifecycle management of VNF [13].

- The Virtualization infrastructure is being deployed as an NFVI and managed by a VIM [12].
- The MEPM is replaced by *MEC Platform Manager Network Function Virtualization (MEPM-V)* that will use the VNFM to delegate the VNF lifecycle management [12].
- The MEO will be replaced by *MEC Application Orchestrator (MEAO)* that relies on NFVO for resource orchestration and also for orchestration of the set of MEC application VNFs as NFV NS [12].

Figure 2.2: MEC system architecture



In the MEC architecture variant, there are different functional elements that compose the MEC system and the functions will be described.

- The MEC host contains the MEC platform and Virtualization infrastructure that gives the MEC applications the ability for compute, storage and network resources. The Virtualization infrastructure has a data plane that executes the traffic rules from the MEC platform. The data plane will route the

traffic between applications, services, DNS server/proxy, 3GPP network, other access networks, local networks or external networks [12].

- The MEC platform will be responsible for different functions. The MEC platform will offer an environment where the MEC applications can discover, advertise, consume and offer MEC services. This will also apply for MEC services being available via other platforms if it is supported [12]. The MEC platform will instruct the data plane accordingly when it receives traffic rules from the MEC platform manager, applications or services. It can also translate tags that represents *User Equipment (UE)* in the traffic rules into specific *Internet Protocol (IP)* addresses when supported [12]. UE is mobile equipment that is used to access the operator's mobile network and supporting applications that transmit IP packets over the mobile network [9]. The MEC platform will configure a DNS proxy/server from the received DNS records from the MEPM [12]. The MEC platform can host MEC services and provide access to persistent storage and time of day information [12]. Through the use of API gateway functionality for the MEC applications, the MEC platform can access the MEC service APIs [12].

- The MEC application will run as a Virtualized application, such as a *Virtual Machine (VM) [12].* A VM is a virtualized computation environment that emulates a physical computer or server [13]. The MEC application can interact with the MEC platform to consume and provide MEC services through the Virtualization infrastructure that is provided by the MEC host. The MEC application can also work with the MEC platform to do other procedures that support the lifecycle of the application, for example indicating availability and preparing relocation of user state. The MEC application can have requirements validated by the MEC system level management, so it can have requirements and rules associated with them. These can be requirements such as required resources, maximum latency, required or useful services, and be assigned to default values by the MEC system level management [12].

- The MEC system level management has a MEO that is the core functionality. The MEO is responsible for different functions. The MEO maintains an overall view of the MEC system that is based on deployed MEC hosts, avail-

able resources, available MEC services and topology [12]. The MEO will keep a record of on-boarded packages and prepare the VIM to handle the applications. The MEO will also check the integrity and authenticity of the on-boarding application packages, validating the application rules and requirements, and if necessary also change them to make the packages comply with operator policies [12]. The MEO will select the appropriate MEC host for application instantiation based on constraints, such as latency, available resources, and available services [12]. The MEO can trigger application instantiation and termination, and application relocation as needed when supported [12]. The *Operations Support System (OSS)* is the OSS of an operator. The OSS receives requests through the *Customer Facing Service (CFS)* portal and from the device applications for instantiation or termination of applications.
The OSS will decide on granting the requests, the granted requests be forwarded to the MEO for further processing. The OSS can also relocate applications between external clouds and the MEC system when the OSS receives requests from device applications if the OSS supports it [12].
The user application is a MEC application. The MEC system will instantiate the user application in the MEC system when it responds to a request of a user through the application in the device [12]. The user application lifecycle management proxy will allow informing the device applications about the state of the user applications. The device applications can request on-boarding, instantiation, termination of user application and relocate user applications in and out of the MEC system if supported through the user application lifecycle management proxy [12]. The user application lifecycle management proxy interacts with the OSS and MEO to process requests from device applications in the device [12]. The user application lifecycle management proxy is only available if it is supported by the MEC system [12].

- The MEC host level management has the MEPM and VIM. The MEPM has the responsibility for the functions of manageing the lifecycle of applications such as informing MEO of relevant application related events [12]. The MEPM provides element management functions to the MEC platform [12]. The MEPM manages the application rules and requirements such as

service authorizations, traffic rules, DNS configuration and resolving conflicts [12]. The MEPM also receives Virtualized fault reports and performance measurements from the VIM for more processing [12].
The VIM takes care of allocating, managing and releasing virtualized, compute, storage and networking, resources of the Virtualization infrastructure [12]. The VIM prepares the virtualization infrastructure to run a software image. This is done by configuring the infrastructure and can include receiving and storing the software image [12]. The VIM can provide rapid provisioning of applications when supported [12]. The VIM collects and reports performance and fault information about the Virtualized resources [12]. The VIM can perform application relocation if supported. The VIM can perform application from/to external environments by interacting with the external cloud manager to do the application relocation [12].

- The device application are the applications in the device, for example UE or a laptop with internet connectivity. The device applications have the capability to interact with the MEC system through the use of a user application lifecycle management proxy [12].
- The customer facing service portal allows the operators' third party customers, such as commercial enterprises, to select and order a set of MEC applications that solve their problems, and receive back service level information from the arranged applications.

2.3 MEC service composition

The MEC environment is characterized by having low latency, low proximity, high bandwidth and real-time insight into radio network information and location awareness. This will create value and opportunities for business models and improve the mobile broadband experience.[3]

Due to the growth of mobile traffic and pressure on costs, there is a need for implementing changes that will maintain the quality of experience. By optimizing network operations and resource utilization, this will generate revenue and keep the high quality user experience. *Internet of Things (IoT)* will keep increasing and congest the network. This will make the network operators to do local analyses to fix security and backhaul impacts. The enterprises want to engage

their customers more efficiently, with secure and low latency connections. Due to the issue of distance when connecting to the cloud, applications and content providers are challenged with high latency of the network.

There is a need to shorten the time to launch new revenue generating applications for current customers, but also other industries and areas such as automotive, industry automation and welfare industries. Technologies that will provide low latency, more flexibility, virtualization and network will make network operation more cost effective and competitive.

Applications on the smartphone and content are being moved to the cloud. The cloud will need an optimized route to give a high quality experience for consumers. Between the network operators, application and content providers, it is essential that there is a tight collaboration between the aforementioned. The collaboration will lead to deploying applications at the edge of the mobile operator's network to provide awareness of the network and context information.

MEC will offer an IT service environment for RAN edge in the mobile network. MEC will offer localized cloud computing capabilities and also expose real-time radio network and context information. This will allow applications and services from mobile operators to efficiently be integrated across multi-vendor platforms. This will create wider innovation into value and revenue generation. Access to content and applications will be faster by making the flow of bandwidth to the core and cloud quicker. Mobile operators can open up their network to third-parties and make *Over the Top (OTT)* players and application developers innovate applications and services for mobile subscribers, enterprises and vertical segments.

2.4 MEC vs Fog computing

Fog computing was first proposed by Cisco, it extends the cloud computing paradigm to the edge of the network, specifically wireless networks for IoT. Fog computing help cloud computing by being closer to the edge and located away from the cloud datacenters. Fog computing are more spread out and distributed in large amounts, with many nodes in the wide geo-distribution. Due to being closer to the IoT, fog computing will allow lower latency and real-time interactions. Fog computing is also flexible in redirecting the application on the mobile device to a new application instance on a fog node if the device moves to far from the initial

fog node.[1]

Chapter 3

State of the art

This chapter will introduce the state of the art of MEC. The chapter will include related works and own comparisons of the MEC simulators and emulators.

3.1 Related works

3.1.1 MEC platforms with simulator and emulators

MEC is a framework that is designed for supporting cloud resources (such as computing, storage and networking) at the edge of the network, which is closer to the data sources. Due to the reference architecture that is given by ETSI, there are different views on how to implement MEC systems. This causes different developments for platforms that has varying features or cases, such as VMs and container migration or IoT devices and moving devices. Simulation environments plays an important role in creating complex systems to model a MEC platform because it will give control over experimental variables that is used to measure a MEC platform [20].

3.1.1.1 Key characteristics

This survey lists key characteristics that the MEC simulator has to implement.

Virtualization support: Virtualization is essential for MEC because it creates an isolated environment for applications to the host machines. This will allow MEC to be flexible in terms of doing installations without affecting other running programs. Hypervisor and container are techniques that is commonly used for

this. Hypervisor provides the guest OS with a virtual operating platform but it will still share the physical hardware. Container will only have basic services for all the applications in the container using virtual-memory support for isolation. Container is closer to MEC purpose due to having limited resources. Virtualization is essential for a MEC simulator because it will be used to see how multiple services is ran to evaluate the node performance and the flexibility of the system [20].

Network support: Network generation can affect the MEC simulation. In cases of 1G and 2G, there is not a need for a MEC platform because 1G and 2G are radio based and do not support high data rate links. 4G and later generations will need a MEC node that is adequate enough for controlling high flows of big data. 5G networks should support low latency edge servers close to the towers and lower the latency to 10-20 ms. A network controller will also be useful to enable route tracking so the requests can be observed on how they are being allocated in the network. Roaming happens when requests do not go to MEC and they re-route to their home network, showing a demand of network simulator. Bandwidth consumption is also a factor, by aggregating the traffic with MEC and the main server, the bandwidth will be reduced through channels being created for similar routes instead of sending requests one by one to the cloud [20].

Orchestrator support: Orchestrator is a controller for MEC. The orchestrator has the responsibility to work with local controller based on its metric status such as CPU and RAM, and decides if MEC needs to offload or join its computation with other MECs [20].

Local controller: Due to MEC being closer to the user, it will have local rights and responsibility. MEC has to measure or aggregate data before sending it to the main cloud. Examples of this is data from IoT devices or wearable devices that has to be analyzed and cleaned beforehand and only send reports to the cloud or make local decisions such as gas consumption of the device to tell the clients. This is a local analysis [20]. Edge content delivery consists of shifting the computation resources, including service contents, from the main cloud to the edge to reduce a big amount of requests to the remote servers. This could make MEC respond to requested content to user applications with low latency. Content moving to MEC is possible through content caching technique. There are four different strategies for caching [20].

- Content Popularity: The content will be considered to download in advance because users could have a higher chance to cache it from the edge depending on the content's popularity.
- Caching policies and Algorithms: Caching has to follow policies and algorithms. For example, by using a machine learning algorithm to cache a video before MEC receives user requests at a dedicated time.
- Caching File Types: Multimedia files are the most common caching files because a lot of users have the same interests for media such was watching a movie or listening to music.
- Mobility Awareness: Caching could be not good enough if the users are moving around and end up far away, which causes them to disconnect from the current connected MEC. Having mobility awareness caching will optimize the amount of needed data to prepare and transfer to users.
- Caching places: Specific areas such as the core network, radio access network or user devices can have cache units in the mobile networks.

MEC has to control their resource by itself to report power consumption, CPU and RAM utilization. These metrics should be under control through both MEC and the orchestrator so the deployment of VM or containers do not lead to a overload condition of MEC and keep the machine state intact [20].

Offloading: There are two purposes for offloading in MEC.

- MEC could reduce power intensive tasks from UE applications by moving it to MEC instead. An example of offloading UE is information encryption/decryption.
- Services deployed in MEC have response times for said services. By offloading the services to MEC instead of the remote server, the requests from UE are not routed to the main server. Offloading also help MEC support the main cloud when it does not process any tasks, so MEC can merge its computation with the main cloud and improve the overall system performance.

Mobility support: There are two aspects for mobility support. The first is the mobility simulation inside simulator tools, the second is mobility management module in MEC. There are moving clients so it is important to test if a

MEC platform has a solution that is sufficient enough to contribute to the moving clients. Mobility management consists of MEC having only access to a coverage range of its deployed *Base Transceiver Station (BTS)*. When a user requests to MEC, the request can be affected by the traffic when the MEC tries to follow and respond in time due to the distance to the UE or when the UE is outside of the coverage range. Setting up mobility management to estimate how the UE movement will be and make decision to support other tasks for the MEC [20].

Migration Support: Service migration is a process that happens during a handover event, and is one of the main supports for mobility management. Services are deployed in a virtualized and encapsulated environment with virtual machines or containers and the migration process will transfer the states of the services such as file system, RAM and CPU to the next MEC from the current MEC and redirect the network traffic to reconnect with the UE. The service migration requires a mobility module to cause the migration process to happen. There are several types of service migration.

- Stateless migration is used for stateless services only to redeploy a similar instance and redirect requests, but are not necessary to move service states.
- Cold migration is a migrated service that is being paused or terminated while the state of the service is transferring. Cold migration gives less complexity for re-connection configuration but also could lead to downtime for the user application.
- Live migration is transferring the service status while the service is still running. There are two sub strategies of live migration, which are *Pre-copy migration* and *Post-copy migration*. The pre-copy method will use several iterations to move the service states before it restarts. The Post-copy will only send states that are at the initial stage of the service after migration.

Security support: There could be security issues during moving a part of resources and computation from cloud to edge. While a cloud server could be protected by a firewall or antivirus, the same can not be said for a MEC due to it being more lightweight and not meeting the hardware requirements for security solutions. This will create a vulnerable endpoint for each MEC device. An authentication module installed to MEC could protect the cloud network and stop unauthorized attacks to the network [20].

3.1.1.2 MEC platform simulators

There are several MEC platform simulators that this survey reviews.

3.1.1.2.1 IoTSim-Edge The IoTSim-Edge is an IoT and edge computing testbed that is based on CloudSim. IoTSim-Edge allows users to test IoT infrastructure and framework. IoTSim-Edge architecture inherits from CloudSim, but has additional edge controller modules such as edge data center, edge broker and edge device. The edge data center manages the core edge infrastructure and it will handle all incoming events to submit to *Mobile Edge let (MEL)* requests. The edge data center module also model the location-awareness mechanism for IoT and edge devices and supports power-awareness technique to track the battery lifetime of MEC. The edge broker module generates user requests and sends them to MEL and MEC. The edge device module presents a real edge device, hosts MEL and facilitates state control policies. IoTSim-Edge can also simulate IoT devices to generate, send data an perform specific sensors such as light, voice etc. [20].

3.1.1.2.2 EdgeCloudSim The EdgeCloudSim is also based on CloudSim. EdgeCloudSim gives a modular architecture for modules that solves demands of edge computing and supports the functionality for computation and networking abilities. The core simulator module can load from a configuration file and logs the results into a CSV file format. There is a transmission delay of WLAN and WAN that the network module solves for both up and download data. The edge orchestrator module makes decisions for the edge cloud system by deciding when and how a request should be handled. The mobility module gives coordinates with x and y for client and edge [20].

3.1.1.2.3 ECSim++ ECSim++ is a fork from OMNetpp simulator and is designed to simulate edge services and edge devices. ECSim++ framework is also based on INET and has extension to build *Edge Cloud Computing (ECC)* environment. The main module for ECC is EdgeNode that extends from the module Router of INET to present a MEC node. All of the current cloud services that are run on EdgeNode is stored in the service table and each cloud service is presented with a IPv4 address. ServiceAware manages incoming requests and decides which services from the service table can handle or route the request desti-

nation as normal if there is not a service for the request in the edge. EdgeService simulates for a running service where the author focuses on *User Datagram Protocol (UDP)* application. *Edge Node Management (Enoma)* controls services on all edge nodes and is located in the cloud core. The power module is a power control for the edge and calculates the energy consumption on each network node to manage data caching in the edge [20].

3.1.1.2.4 iFogSim iFogSim is also inherited from CloudSim and is a toolkit that models IoT and Fog environments and measures cloud and network metrics. The architecture for iFogSim starts with IoT sensors and actuators that receives response data to the real world. The data will be analyzed in the upper layers. The fog devices in the hierarchy topology act as a gateway for a parent-child pair in communication. Resource management manages the resources in the fog device layer and the policies allow component migration and dynamic changes in device resources. The application model simulates data processing components that is used for transferring data among dependent modules. iFogSim also supports cost, energy moden and VM resource migration from CloudSim [20].

3.1.1.2.5 CloudSimSDN CloudSimSDN is also based on CloudSim, but this simulation framework is focused on *Software Defined Network (SDN)* and has additional components to control SDN behavior. The framework design for CloudSimSDN is closer to telecommunication infrastructure that contains cloud resources such as host, switch and link, and workload control, such as request, processing and transmission, and finally a virtual channel. The virtual channel is created to separate dataflows. The idea behind a virtual channel is to allow priority traffic to consume more bandwidth than normal traffic while also enabling common routes to use the same channel [20].

3.1.1.2.6 YAFS *Yet Another Fog Simulator (YAFS)* has the purpose to analyze the design and deployment of applications through customized and dynamical strategies. The metrics that YAFS reports are network utilization, network delay, response time and waiting time where all of them are stored in a CSV format. The architecture for YAFS consists of 6 main classes. The classes are Topology, Core, Application, Selection, Placement and Population. The Core integrates other components while the topology with the network links prepares

a note graph that is interconnected. The other classes provide allocation of resources and orchestration in entities of the structure. In the YAFS evaluation there is a scenario for dynamic movement, but it is mainly focused on resource allocation and service migration is not mentioned [20].

3.1.1.2.7 EmuFog EmuFog is an emulation framework for fog computing. EmuFog allows designing network topology and use Docker-based applications on nodes that is connected to the simulated network. EmuFog is built on top of MaxiNet which is also an extension of Mininet. EmuFog has a four step workflow, topology generation, topology transformation, topology enhancement, and deployment and execution [20].

3.1.1.2.8 FogTorch FogTorch uses Monte Carlo simulations to deploy applications and components for fog infrastructure. FogTorch uses an agriculture scenario to investigate remote monitoring and irrigation of crops. FogTorch consists of 3 components. First, a ThingsController that interacts with IoT sensors and actuators as a fire sensor, an electronic water valve or a video camera. Second, DataStorage that stores all collected information from IoT devices. Lastly, a Dashboard that takes responsibility for virtualizing, monitoring and controlling data from the sensors, while also doing that for historical data and machine learning engine rules. Monte Carlo simulators are used for accounting for probabilities when assigning *Quality of Service (QoS)* profiles to communication hyper links. FogTorch generates and runs with input for the fog infrastructure with each communication link. The end result will show two aggregated metrics of QoS-assurance, which is percentage of runs for generated deployment and Fog resource consumption, which is aggregated average percentage of consumed RAM and storage in fog nodes [20].

3.1.1.2.9 BMEC BMEC is a blockchain based MEC management system, it is built on top of OMNetpp simulator, INET framework and mobility simulator from SimuLTE, Veins and Sumo. The system architecture has 3 main modules that are the mobility module, orchestrator module and authentication module. The mobility module runs a simulated *Transmission Control Protocol (TCP)* application to forward requests. The orchestrator receives requests and make decisions to deploy a Docker based application into the closest node and the nodes are set up

in independent physical machines so the testing environment can be flexible and scalable. The authentication module controls request and decision flows of all the modules, the service information is stored in a blockchain network and will require access rights to receive. The blockchain also builds a secure channel for infrastructure vendor communication or service [20].

Simulator	Virtualization	Network	Orchestrator	Local controller	Offload	Mobility	Security
IoTSim-Edge	Yes, simulated VM	Up to 4G	No	Basic	Basic	Yes	No
EdgeCloudSim	Yes, simulated VM	Up to 4G	Yes	Basic	Basic	Yes	No
ECSim++	No	No	Yes	Basic	Basic	No	No
iFogSim	Yes, simulated VM	Up to 4G	No	Basic	Basic	No	No
CloudSimSDN	Yes, simulated VM	Up to 4G, bandwidth control	No	Basic	Basic	No	No
YAFS	No	No	No	Basic	Basic	Yes	No
FogTorch	No	Common	No	Basic	Basic	No	No
BMEC	Yes, Docker	Up to 3G	Yes	Basic, no machine state control	Basic	Yes	With blockchain

Table 3.1: Comparison of simulators

3.2 MEC simulators and emulators

3.2.1 MEC simulators

3.2.1.1 IoTSim-Edge

IoTSim-Edge has features to create an environment for IoT and edge computing. IoTSim-Edge uses existing classes of CloudSim and extends them. There are also new classes that were designed for IoTSim-Edge to model an edge infrastructure. These classes are EdgeDataCenter, EdgeBroker, EdgeDatacenterCharacteristics, EdgeDevice, MicroELement and EdgeLet [16].

Figure 3.1: IoTSim-Edge architecture

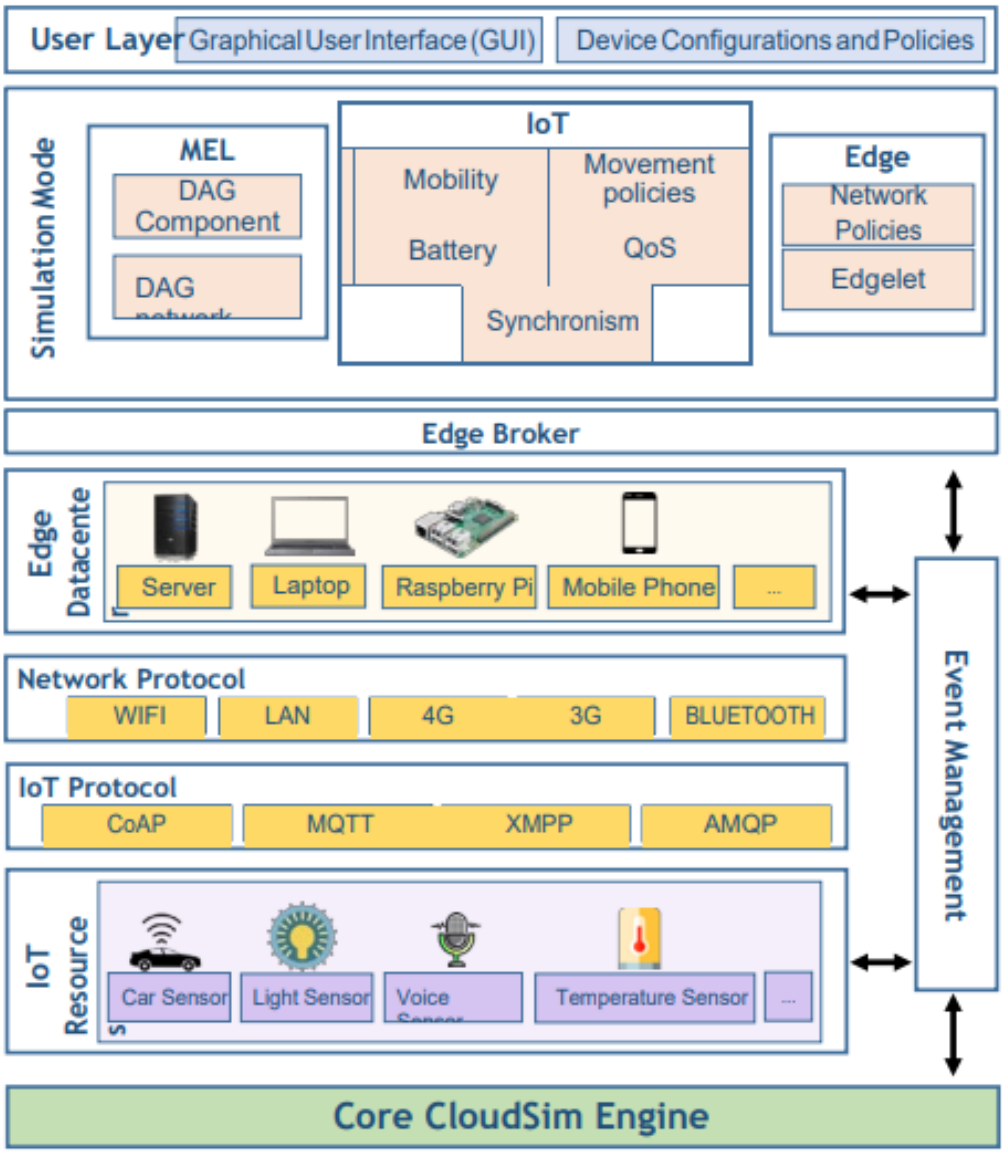
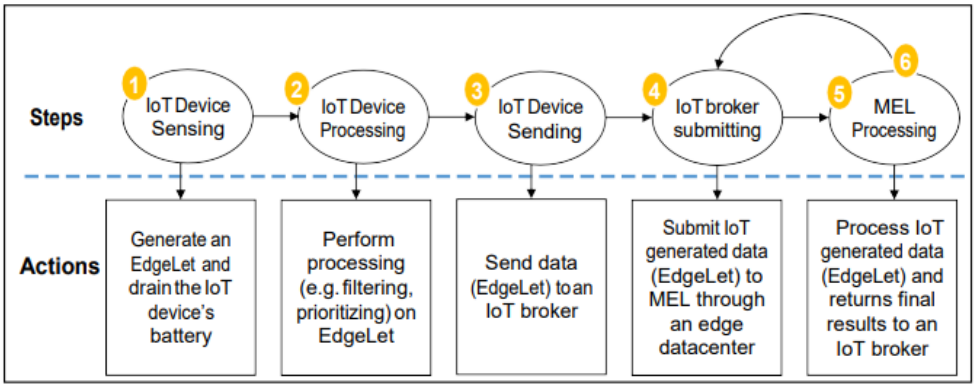


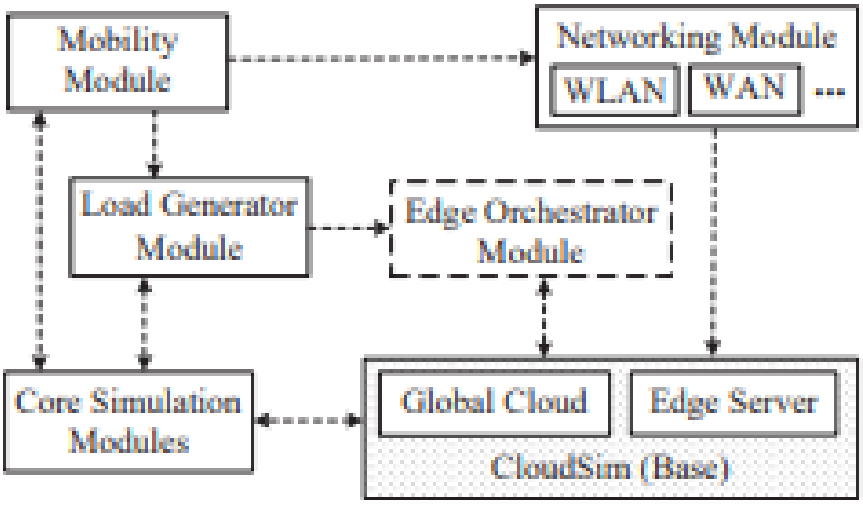
Figure 3.2: IoTSim-Edge steps



3.2.1.2 EdgeCloudSim

EdgeCloudSim is a simulator that provides a modular architecture to support functionalities such as modeling for WLAN and WAN, device mobility model, realistic and tunable load generator. EdgeCloudSim can support simulating multi-tier scenarios where multiple edge servers are running with upper layer cloud solutions. EdgeCloudSim relies on Cloudsim's capabilities and is open source. EdgeCloudSim has different modules to address specific aspects of edge computing. These modules are: Core Simulation Module, Networking Module, Edge Orchestrator Module, Mobility Module and Load Generator Module [8].

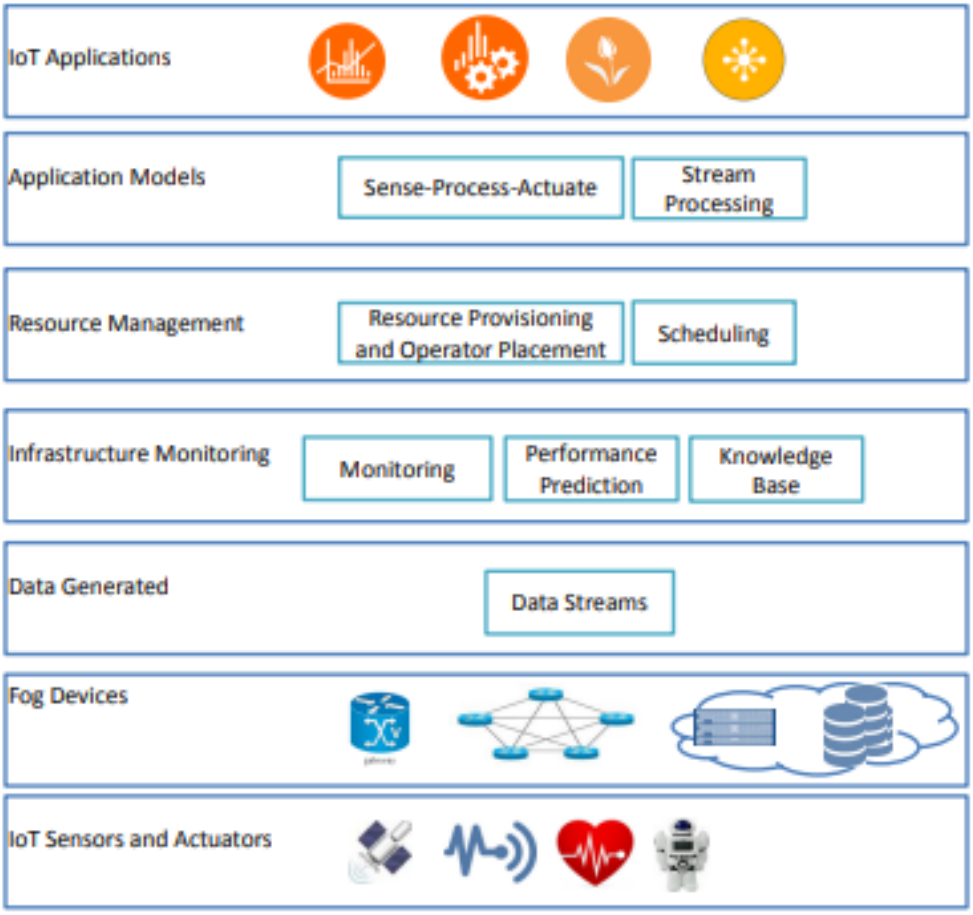
Figure 3.3: EdgeCloudSim modules



3.2.1.3 iFogSim

iFogSim is a simulator that will enable simulation of resource management and create different scenarios of applications scheduling policies across edge and cloud resources. The architecture of consists of IoT sensors and actuators, and the actuators are responsible for controlling a mechanism or system. The Fog Device that hosts the application modules. The IoT Data Streams from the IoT sensors. The Infrastructure Monitoring that collect the data from the IoT sensors, actuators and fog devices, and keeps tracks of performance and status. The Resource Management that manages the resources so that the QoS requirements are met. The Application Model is based on the **Distributed Data Flow (DDF)** model and the applications are developed for deployment for the DDF. An application is a collection of modules for data processing [6].

Figure 3.4: iFogSim Architecture



3.2.1.4 Eclipse fog05

Eclipse fog05 is used for End-to-End management of compute, storage, networking and I/O fabric in the Edge environment. Eclipse fog05 is based on a decentralized architecture [14]. Eclipse fog05 allows virtualization by virtualizing hardware infrastructure such as computational, communication, storage and I/O resources through the use of *Fog Infrastructure Manager (FIM)*. Eclipse fog05 has a network through the component Network Manager Plugin to create network, routers and ports. Eclipse fog05 has an orchestrator called *Fog Orchestration Engine (FOrcE)*. FOrcE uses Zenoh for state distribution and storage [14].

3.2.1.5 Italtel i-MEC

Italtel i-MEC enables services that leverage End-to-End latency such as uRLLC and cV2X, pre-processing at the edge such as mMTC, and broadband services like eMBB. Italtel i-MEC help reduce the traffic load on the backhauling transport network. Italtel i-MEC is virtualized through NFV with VMs. Italtel i-MEC has network capabilities for 5G. Italtel i-MEC also has the MEAO for orchestration [15].

3.2.1.6 Simu5G

Simu5G is based on the OMNeT++ framework. Simu5G MEC model has the goal of offering a tool for rapid prototyping of MEC applications. Simu5G can run as a real-time network emulator and also interface real MEC apps with it. This will make testing MEC applications in computation/communication framework have 5G transport, MEC infrastructure and MEC services. The system level modules has the modules *User Application LifeCycle Management Proxy (UALCMP)* and MEC orchestrator. Both UALCMP and MEC orchestrator are connected to each other. The MEC orchestrator manages the MEC hosts. The host level module MEC host have the MEC applications that can configure the amount of resources such as RAM, storage and CPU. Simu5G Can run both simulations and emulations [17].

3.2.1.7 Comparison between simulators

Here is the comparison between the simulators, taking in consideration of virtualization, network, orchestrator, RNI, location, traffic and security.

Simulator	Virtualization	Network	Orchestrator	RNI	Location	Traffic	Security
IoTSim-Edge	Yes	Common	No	Yes	Yes	No	No
EdgeCloudSim	Yes	Common	Yes	Yes	Yes	Yes	No
iFogSim	Yes	Up to 4G	No	Yes	Yes	No	No
Eclipse fog05	Yes	Common	No	Yes	Yes	No	No
Italtel i-Mec	Yes	Up to 5G	Yes	Yes	Yes	Yes	No
Simu5G	Yes	Up to 5G	Yes	Yes	Yes	Yes	No

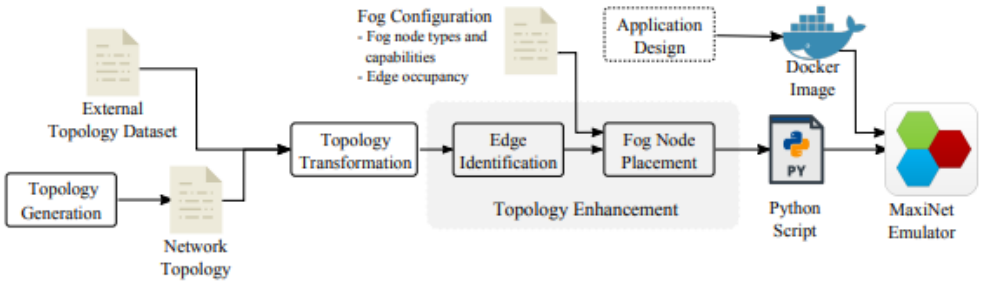
Table 3.2: Comparison of simulators

3.2.2 MEC emulators

3.2.2.1 EmuFog

EmuFog is an emulation framework for fog computing and is built on top of MaxiNet, while MaxiNet is also an extension of Mininet. EmuFog enables designs for network topology and run Docker-based applications on nodes connected by the simulated network. The EmuFog architecture has 4 steps in the workflow. First, the topology generation that generates a network topology or loads a configuration from files. Second, topology transformation that translates an undirected graph of network to a network topology model by EmuFog. Third, topology enhancement where the identified edge will combine with fog configuration policies to create a fog node placement where it will show fog nodes and the computational for the nodes. Lastly, deployment and execution of the fog nodes to be emulated from step 3 and services that are running in Docker containers will get deployed in the nodes to test the performance of the node [7].

Figure 3.5: EmuFog workflow

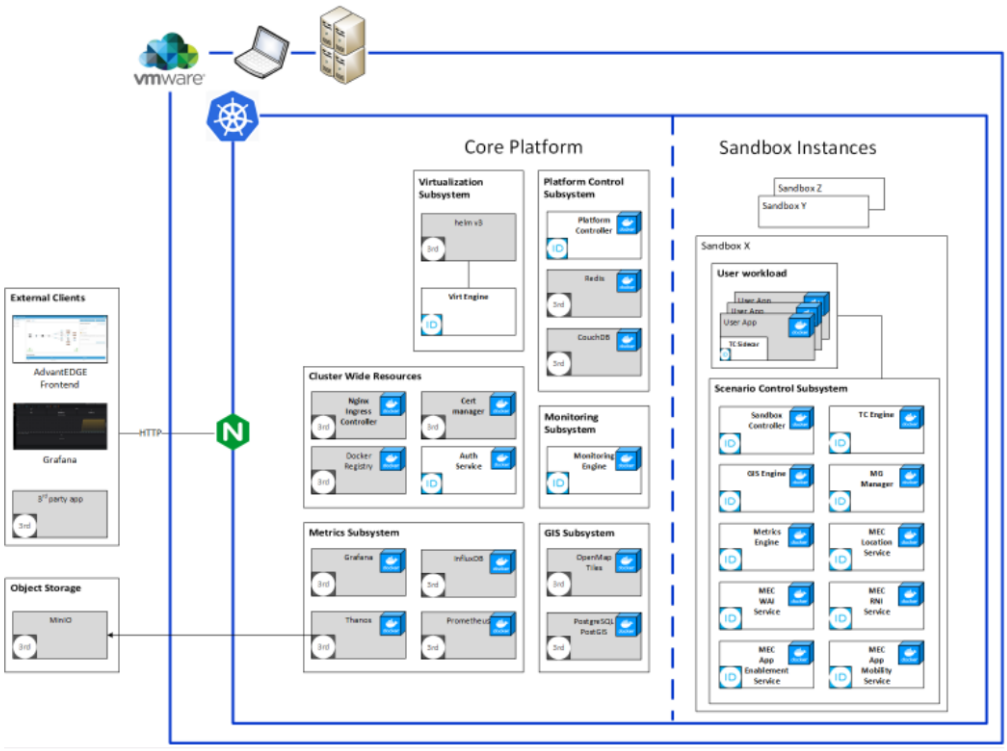


3.2.2.2 AdvantEDGE

AdvantEDGE runs on Docker and Kubernetes, it is a *Mobile Edge Emulation Platform (MEEP)*. AdvantEDGE gives an environment for emulation which will enable experimenting with Edge Computing Technologies, Applications and Services. The platform facilitates edge deployment models [10]. AdvantEDGE has virtualization through Docker containers that are designed to operate in Kubernetes environment. AdvantEDGE supports 4G, 5G and WLAN wireless connectivity. AdvantEDGE can offload to different nodes in the network model it uses. The network model have layers where the data flow through a path. AdvantEDGE

has mobility support through the Mobility Group Manager component in the AdvantEDGE Sandbox. AdvantEDGE supports local controller where AdvantEDGE can limit the CPU and RA4M. AdvantEDGE was last updated in December 2021 as of this writing [11].

Figure 3.6: AdvantEDGE architecture



3.2.2.3 Comparison between emulators

Here is the comparison between the simulators, taking in consideration of virtualization, network, orchestrator, RNI, location, traffic and security.

Emulator	Virtualization	Network	Orchestrator	RNI	Location	Traffic	Security
EmuFog	Yes, Docker	Common	No	Basic	Basic	Yes	No
AdvantEDGE	Yes, Docker	Up to 5G	No	Basic	Basic	Yes	No

Table 3.3: Comparison of emulators

Chapter 4

Design

This chapter is about using a simulator/emulator to demonstrate the usage of ETSI MEC. The simulator being used is Simu5G. Simu5G has a MEC system and MEC host models of the MEC architecture.

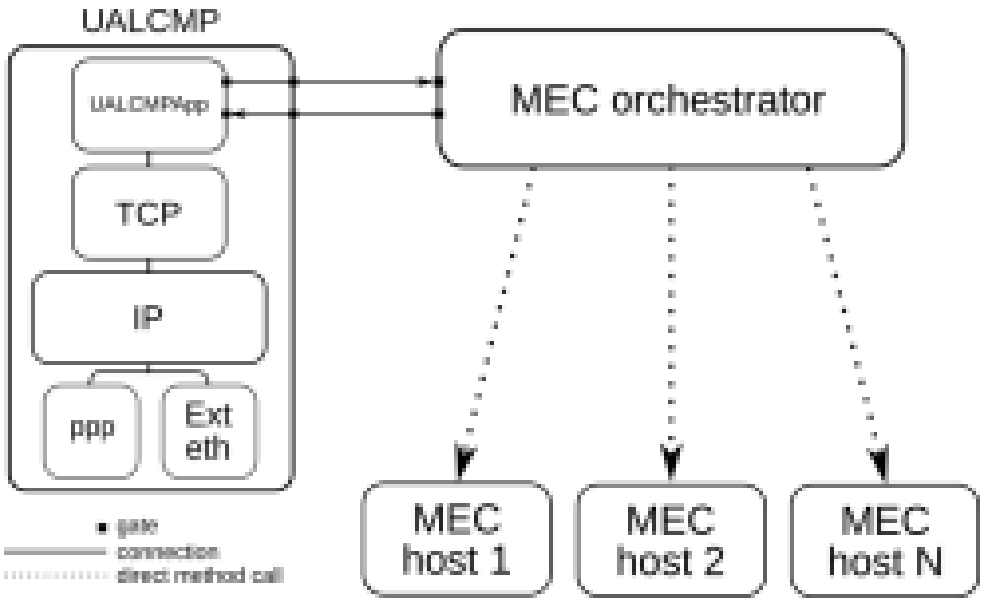
4.1 Model of ETSI MEC

4.1.1 MEC system model

The system level has UALCMP and the MEC orchestrator. The UALCMP module is made out of several parts such as the TCP/IP stack and an application module that provides the *Representational state transfer (RESTful)* API that is being used by the Device app to trigger instantiation and termination of MEC apps through the Mx2 reference point. The MEC orchestrator is a module that is connected to the UALCMP through gates. When there are multiple MEC systems, the MEC orchestrator will manage subset of the MEC hosts in the simulation through the use of mecHostsList parameter. When the UALCMP gives a request to the MEC orchestrator after getting a request from the Device app, the MEC orchestrator will select the most suitable MEC host that are within its MEC system. In the AppDescriptor JSON files are where the list of required MEC services and computing resources are specified. The AppDescriptor files are then checked against the available resources of the MEC hosts. Once the MEC host is identified, the MEC orchestrator will trigger the MEC app instantiation by creating the OMNeT++ module. This will run the application logic and mark the required

MEC host resources as allocated. The Device app is embedded in the UE client, and its functions are standard. For example managing instantiation/termination of the MEC apps by interfacing with the LCM proxy. The Device app will query the LCM proxy through the RESTful API and communicate with the UE app over UDP socket with the use of a simple interface regarding a MEC app. For example messages for creation, termination and acknowledgments for the MEC app [19].

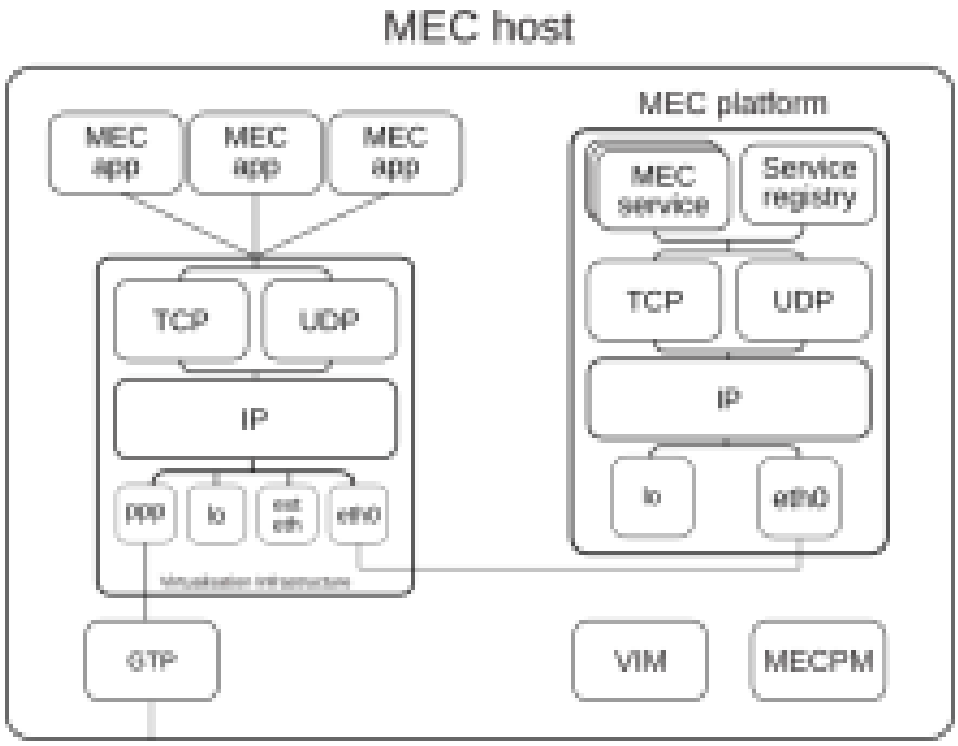
Figure 4.1: MEC system level model



4.1.2 MEC host model

The host module has the MEC host level of the MEC architecture. The host module is configurable through the NED and INI files to change the amount of resources such as RAM, storage and CPU that is being allocated to the MEC apps. The main components of the MEC host module is the VIM and MEC platform modules. The MEC platform entities implement a RESTful HTTP server as a TCP application that runs on top of the TCP/IP protocol stack of the MEC platform when emulated. The module MecServiceBase implements all the non-functional requirements for running an HTTP server [19].

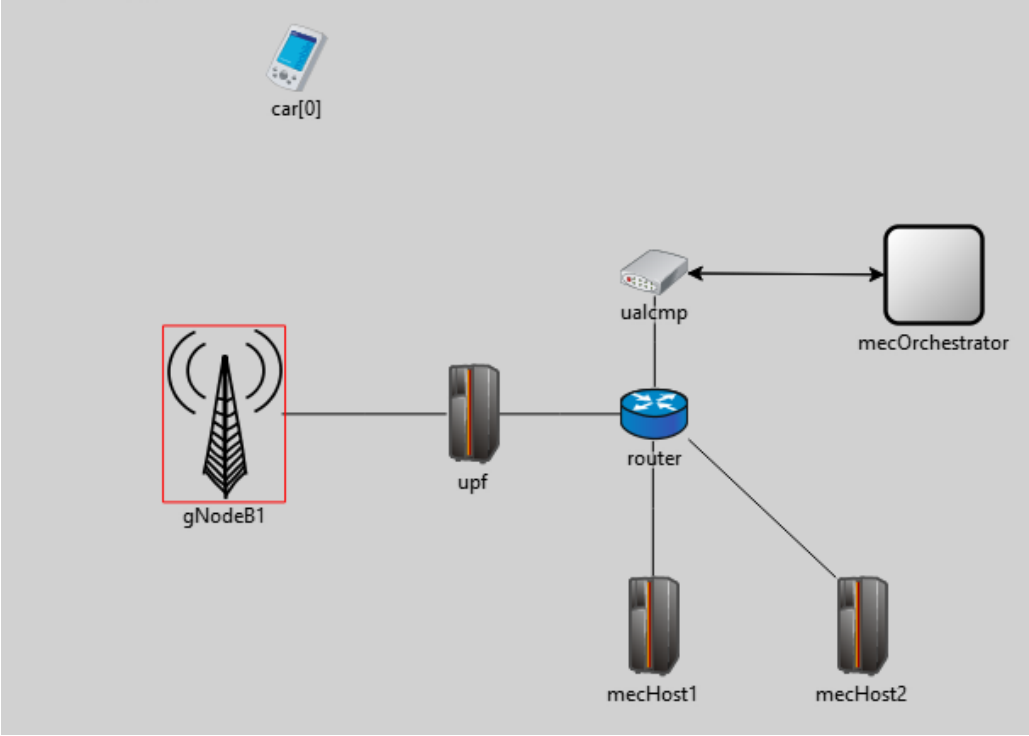
Figure 4.2: MEC host level model



There are two ETSI MEC services that is implemented: *Radio Network Information service (RNIS)*, which allows users to gather up-to-date information about the radio network condition and the UEs that are connected to the base station associated with the MEC host. The other service is the Location service, which provides accurate information about UEs and base station locations, enabling active device location tracking and location-based service recommendations. The ServiceRegistry module allow MEC apps to discover MEC services by implementing REST resources and HTTP methods through the Mp1 reference point. MEC apps are deployed as applications with the use of TCP and UDP transport-layer protocols of the virtualization infrastructure module. The VIM uses allocated resources to compute the delay that is used to model the processing time. CPU is calculated through instructions per second so the MEC app processing time can be modeled based on the MEC host processing speed and load [19].

4.1.3 Simulation of MEC system

Figure 4.3: MEC simulator system



The simulated MEC system has a UALCMP, a MEC orchestrator and two MEC hosts. The two MEC hosts are mecHost1 and mecHost2 and they are connected with a *gNodeB (gNB)*. The car is a vector of UEs and each of the car run an application called UEWarningAlertApp. The UEWarningAlertApp instantiates a MEC app called MECWarningAlertApp. The MECWarningAlertApp is onboarded on the MEC system and during the network initialization will send alert messages to the cars when they enter a danger zone. The cars also run the Device app which is responsible for requesting MEC app instantiation to the UALCMP on behalf of the UEs. The *User Plane Functions (UPF)* module is also in the figure. The AppDescriptor is also required for MECWarningAlertApp [19].

Figure 4.4: MEC AppDescriptor

```
{
"appDid" : "WAMECAPP",
"appName" : "MECWarningAlertApp",
"appProvider" : "simu5g.apps.mec.WarningAlert.MECWarningAlertApp",
"appInfoName" : "appInfoName_",
"appDescription" : "appDescription_",
"virtualComputeDescriptor" :{
    "virtualDisk": 10,
    "virtualCpu" : 1500,
    "virtualMemory":10
    },
"appServiceRequired": [
    {
        "ServiceDependency" :{
            "serName" : "LocationService",
            "version" : "v2",
            "serCategory": "Location"
        }
    }
],

"omnetppServiceRequired": "MEWarningAlertService"
}
```

This shows that the MEC app requires 10 MB RAM, 1500 instructions per second of CPU and 10 MB of storage. The MEC app also uses location service. The next step is to configure the car and MEC entities.

Figure 4.5: MEC car configuration

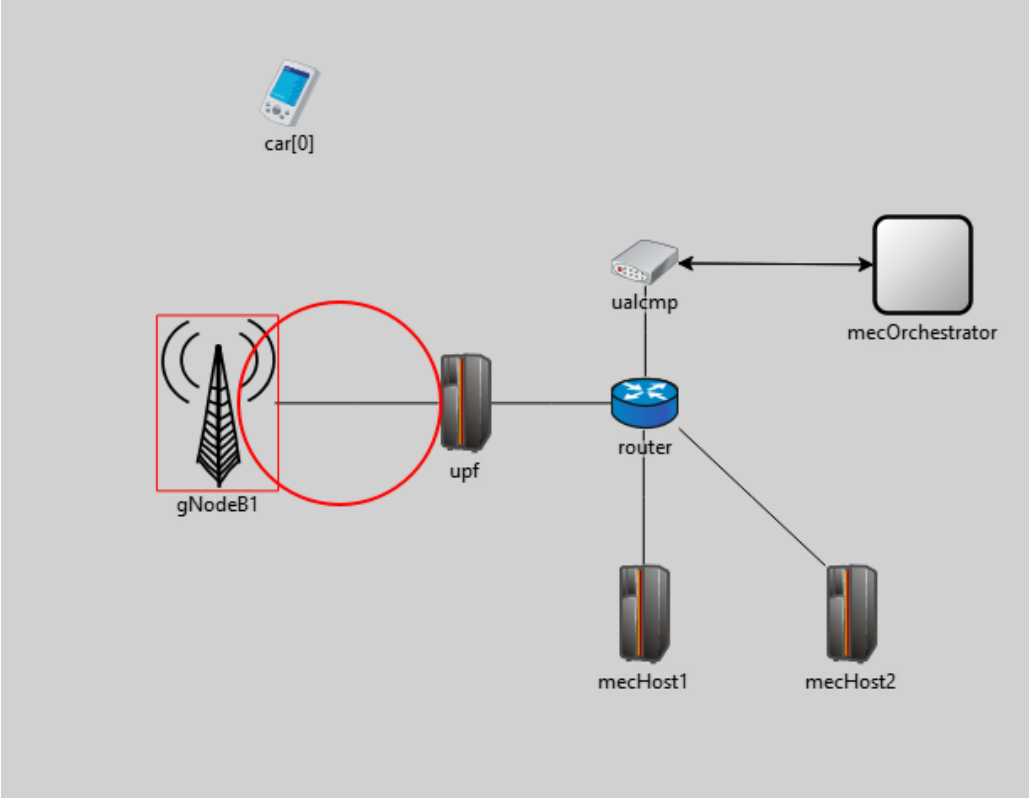
```
123 ##########################################################
124 #                     App Layer                          #
125 ##########################################################
126
127 ########################_Car Side_#######################
128
129 #------------UEWarningAlertApp---------------
130 *.car[*].numApps = 2
131 *.car[*].app[0].typename = "DeviceApp"
132 *.car[*].app[0].localPort = 4500
133 *.car[*].app[0].UALCMPAddress = "ualcmp"
134 *.car[*].app[0].UALCMPPort = 1000
135 *.car[*].app[0].appPackageSource = "ApplicationDescriptors/WarningAlertApp.json"
136
137 *.car[*].app[1].typename = "UEWarningAlertApp"
138 *.car[*].app[1].deviceAppAddress = "car["+string(ancestorIndex(1))+"]"
139 *.car[*].app[1].deviceAppPort = 4500
140 *.car[*].app[1].startTime = 1s                                    #when sending start warning alert app
141 *.car[*].app[1].stopTime = 30s                                    #when sending stop MEC warning alert app
142
143
144
145 #*.car[*].app[0].requiredRam = 10MB
146 #*.car[*].app[0].requiredDisk = 10MB
147 #*.car[*].app[0].requiredCpu = 0.01
148 #----------------------------------------
149
150 ######################_ME Hosts Side_####################
151 # available resources
152 *.mecHost1.maxMECApps = 100                # max ME Apps to instantiate
153 *.mecHost1.maxRam = 32GB                      # max KBytes of Ram
154 *.mecHost1.maxDisk = 100TB                 # max KBytes of Disk Space
155 *.mecHost1.maxCpuSpeed = 400000            # max CPU
156
157 *.mecHost2.maxMECApps = 100                # max ME Apps to instantiate
158 *.mecHost2.maxRam = 32GB                      # max KBytes of Ram
159 *.mecHost2.maxDisk = 100TB                 # max KBytes of Disk Space
160 *.mecHost2.maxCpuSpeed = 500                  # max CPU
161
```

Figure 4.6: MEC system configuration

```
150 ######################_ME Hosts Side_#####################
151 # available resources
152 *.mecHost1.maxMECApps = 100                # max ME Apps to instantiate
153 *.mecHost1.maxRam = 32GB                      # max KBytes of Ram
154 *.mecHost1.maxDisk = 100TB                 # max KBytes of Disk Space
155 *.mecHost1.maxCpuSpeed = 400000            # max CPU
156
157 *.mecHost2.maxMECApps = 100                # max ME Apps to instantiate
158 *.mecHost2.maxRam = 32GB                      # max KBytes of Ram
159 *.mecHost2.maxDisk = 100TB                 # max KBytes of Disk Space
160 *.mecHost2.maxCpuSpeed = 500                   # max CPU
161
162 #----------------------------------------
163
164 *.mecHost*.eNBList = "gNodeB1"             # gNBs associated to the MEC Hosts
165
166 #-------REST Services:---------------
167
168 # MEC host 1 services configurations
169 *.mecHost1.mecPlatform.numMecServices = 1
170 *.mecHost1.mecPlatform.mecService[0].typename = "RNIService"
171 *.mecHost1.mecPlatform.mecService[0].localAddress = "mecHost1.mecPlatform" #da cambiare!!
172 *.mecHost1.mecPlatform.mecService[0].localPort = 10020
173
174 *.mecHost1.mecPlatform.serviceRegistry.localAddress = "mecHost1.mecPlatform" #da cambiare!!
175 *.mecHost1.mecPlatform.serviceRegistry.localPort = 10021
176
177 # MEC host 2 services configurations
178
179 *.mecHost2.mecPlatform.numMecServices = 1
180 *.mecHost2.mecPlatform.mecService[0].typename = "LocationService"
181 *.mecHost2.mecPlatform.mecService[0].localAddress = "mecHost2.mecPlatform" #da cambiare!!
182 *.mecHost2.mecPlatform.mecService[0].localPort = 10020
183
184 *.mecHost2.mecPlatform.serviceRegistry.localAddress = "mecHost2.mecPlatform" #da cambiare!!
185 *.mecHost2.mecPlatform.serviceRegistry.localPort = 10021
186
```

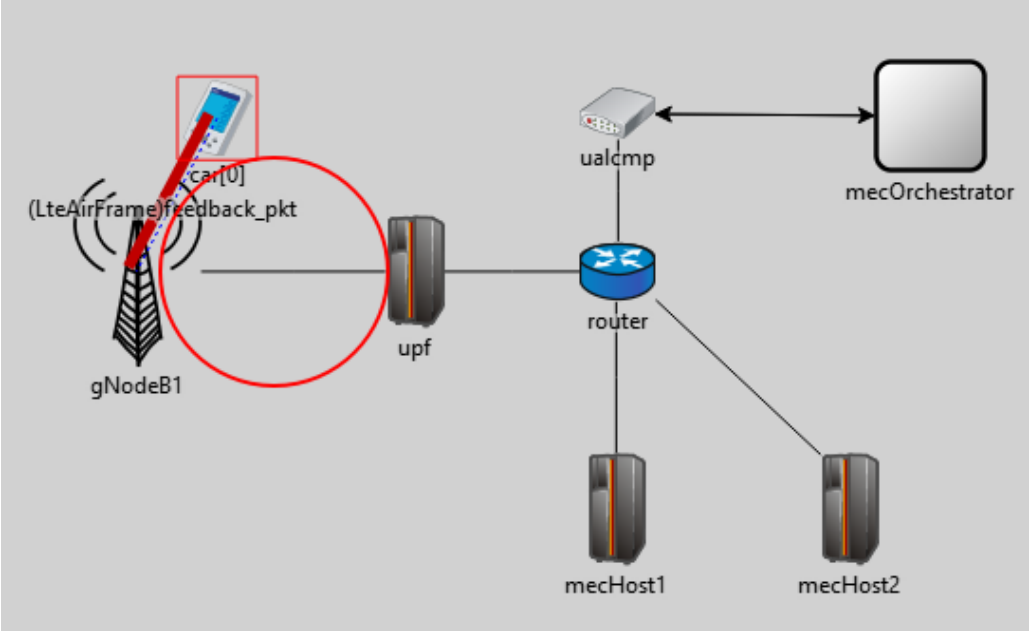
Both of the MEC hosts have the same computational capacity, but only mecHost2 has a Location service on its MEC platform, therefore it will be chosen by the MEC orchestrator to deploy the MEC app.

Figure 4.7: MEC simulator system 2



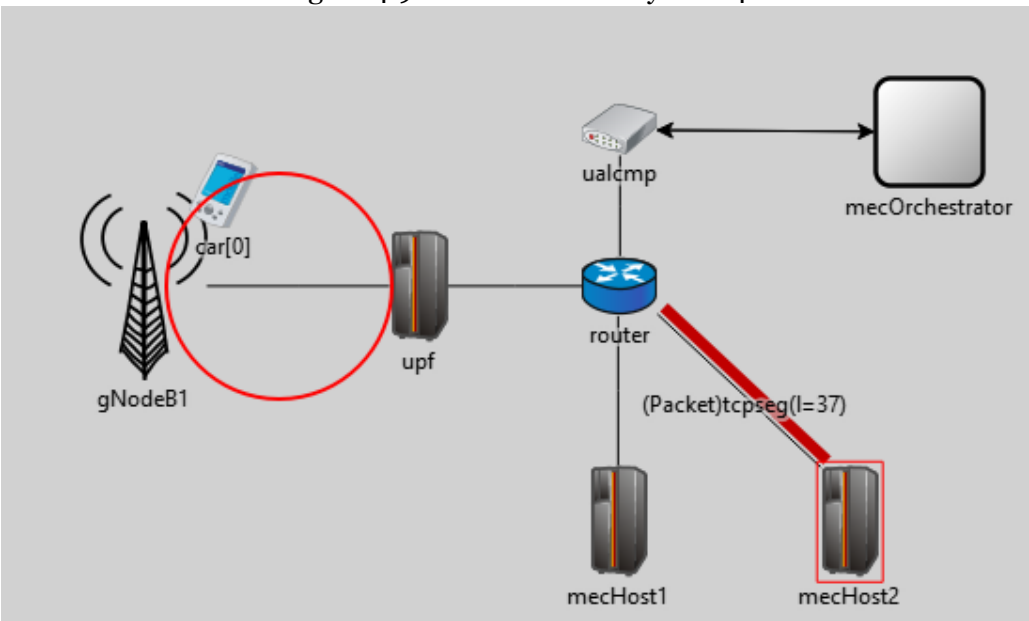
Here the car is approaching gNB1 and the car is moving down towards the red circle as seen in the figure 4.7. The red circle is the danger zone.

Figure 4.8: MEC simulator system 3



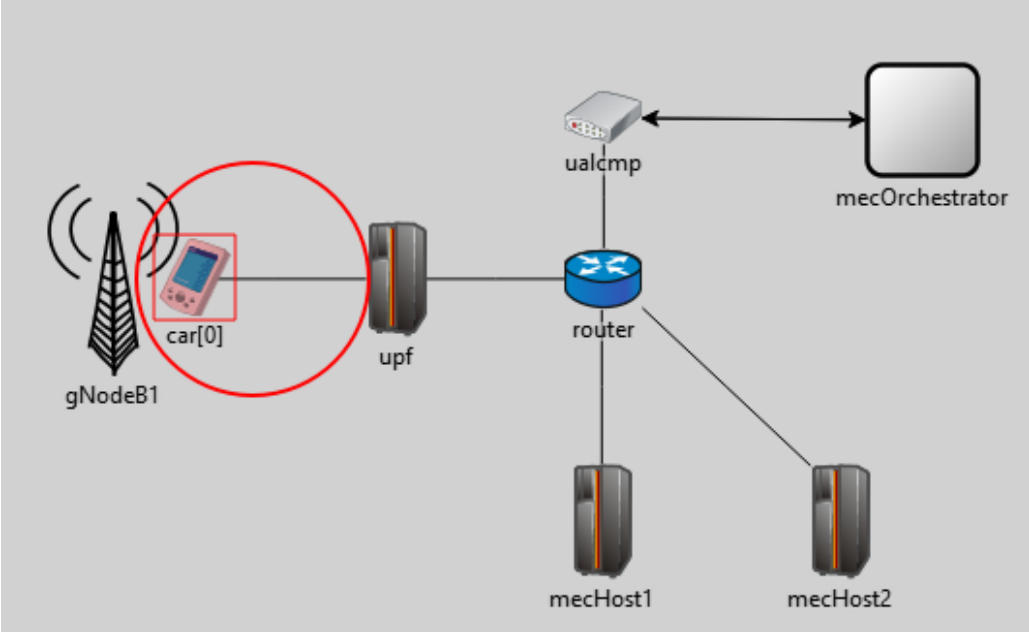
As the car is approaching closer, the car is communicating with gNB1 and receiving information about the danger zone.

Figure 4.9: MEC simulator system 4



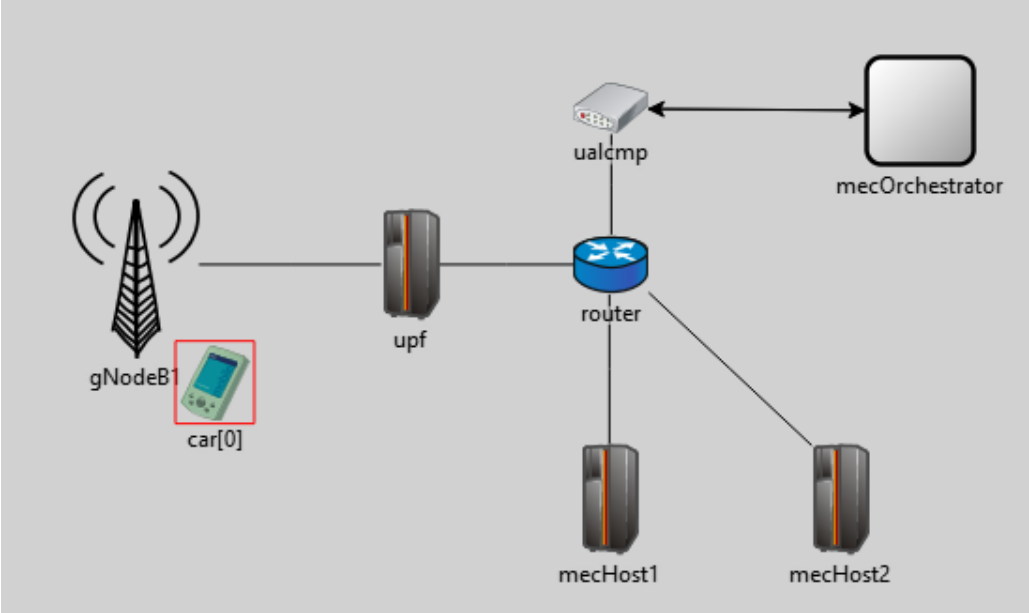
The car has reached the danger zone and now the UALCMP has started instantiation of the MEC apps.

Figure 4.10: MEC simulator system 5



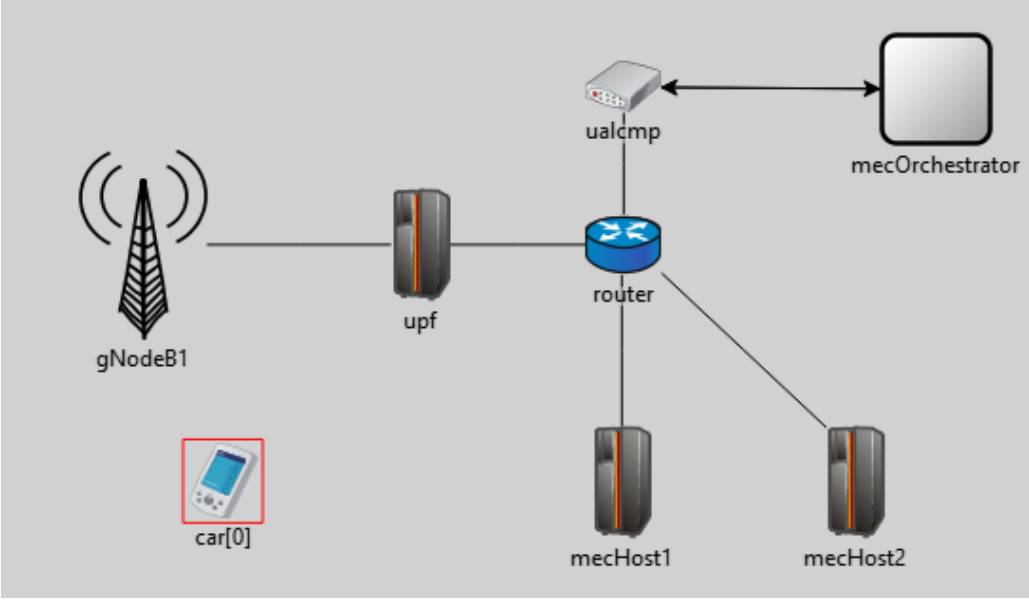
The car is inside the danger zone and is continuously communicating with the MEC system to be alerted that the car is inside the danger zone.

Figure 4.11: MEC simulator system 6



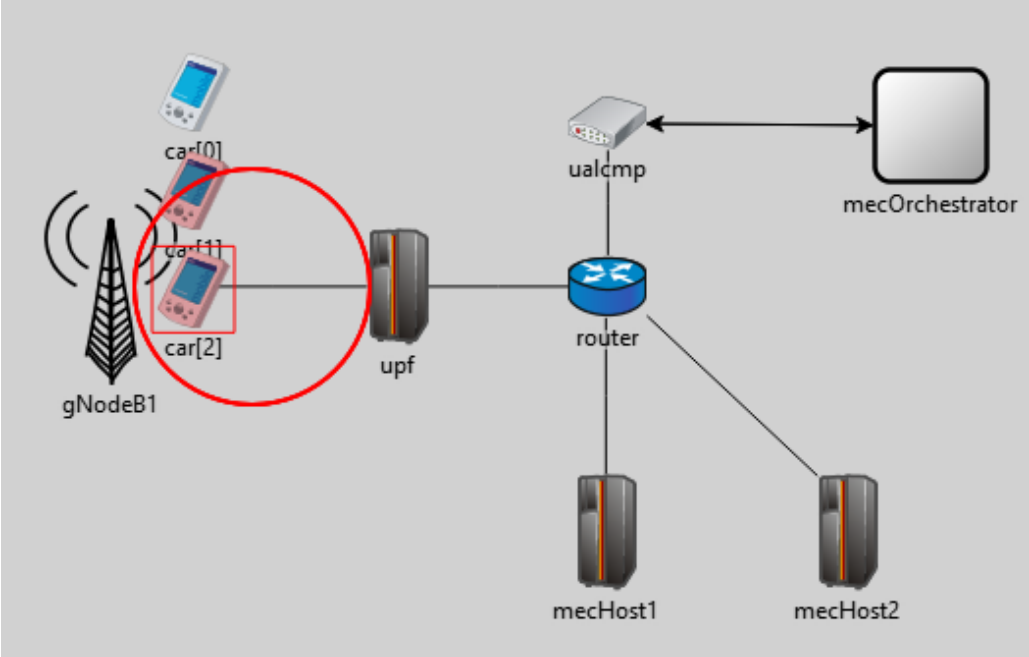
The car has left the danger zone and is communicating with the MEC system about leaving the danger zone.

Figure 4.12: MEC simulator system 7



The car is outside of the range of the gNB and no longer communicating with the MEC system about the danger zone.

Figure 4.13: MEC simulator system 8



It is also possible to have multiple cars and have them communicate with the MEC system about the danger zone, although the performance of the simulator will be slower than only one car that is being used in the simulation.

Chapter 5

Conclusion

In this thesis, there was an explanation of the background of MEC that was used to understand the ETSI MEC standardization which is one of the leading technologies for the realization of 5G technology. Afterwards there was a chapter about state of the art of different simulators/emulators that were compared with each other about their functionalities for MEC architecture. Lastly, one simulator was used and demonstrated to show the functionalities of MEC.

Bibliography

[1] F Bonomi et al. “Fog computing andits role in Internet of Things”. In: *Proceedings of the MCC* (2014).

[2] BG Gopal and PG Kuppusamy. “A comparative study on 4G and 5G technology for wireless applications”. In: *IOSR Journal of Electronics and Communication Engineering* 10.6 (2015), pp. 2278–2834.

[3] Yun Chao Hu et al. “Mobile edge computing—A key technology towards 5G”. In: *ETSI white paper* 11.11 (2015), pp. 1–16.

[4] M Series. “IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond”. In: *Recommendation ITU* 2083 (2015), p. 21.

[5] *ETSI - Welcome to the world of standards!* 2017. URL: `https://www.etsi.org/images/files/ETSInewsletter/etsinewsletter-issue2-2017.pdf`.

[6] Harshit Gupta et al. “iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments”. In: *Software: Practice and Experience* 47.9 (2017), pp. 1275–1296.

[7] Ruben Mayer et al. “Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures”. In: *2017 IEEE Fog World Congress (FWC)*. IEEE. 2017, pp. 1–6.

[8] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. “Edgecloudsim: An environment for performance evaluation of edge computing systems”. In: *Transactions on Emerging Telecommunications Technologies* 29.11 (2018), e3493.

[9] Jan. 2019. URL: https://www.etsi.org/deliver/etsi_gs/mec/001_099/001/02.01.01_60/gs_mec001v020101p.pdf.

[10] Inc. InterDigital Communications. "AdvantEDGE Mobile Edge Emulation Platform". In: (2019). URL: https://interdigitalinc.github.io/AdvantEDGE/.

[11] M. Roy K.Di Lallo S. Pastor. "AdvantEDGE - Github Repository". In: (2019). URL: https://github.com/InterDigitalInc/AdvantEDGE.

[12] *"GS MEC 003 V2.2.1: Multi-access Edge Computing (MEC); Framework and Reference Architecture*. Dec. 2020. URL: https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.02.01_60/gs_MEC003v020201p.pdf.

[13] *ETSI GR NFV 003 v1.5*. Jan. 2020. URL: https://www.etsi.org/deliver/etsi_gr/NFV/001_099/003/01.05.01_60/gr_NFV003v010501p.pdf.

[14] Eclipse Foundation. "Eclipse fog05 The End-to-End Compute, Storage and Networking Virtualisation solution". In: (2020). URL: https://fog05.io/.

[15] Italtel. "i-MEC Italtel Multi-Access Edge Computing". In: (2020). URL: https://www.italtel.com/content/uploads/2020/10/i-MEC-Functional-Description.pdf?x67374.

[16] Devki Nandan Jha et al. "IoTSim-Edge: a simulation framework for modeling the behavior of Internet of Things and edge computing environments". In: *Software: Practice and Experience* 50.6 (2020), pp. 844–867.

[17] G. Nardini A. Virdis. *ETSI MEC model in Simu5G*. 2021. URL: http://simu5g.org/MEC.html.

[18] Gianfranco Nencioni, Rosario G Garroppo, and Ruxandra F Olimid. "5G Multi-access Edge Computing: Security, Dependability, and Performance". In: *arXiv preprint arXiv:2107.13374* (2021).

[19] Alessandro Noferi et al. "Deployment and configuration of MEC apps with Simu5G". In: *arXiv preprint arXiv:2109.12048* (2021).

[20] Thanh Van Le et al. “Mobile edge cloud computing simulation tool survey and literature reviews”. In: (). URL: https://www.researchgate.net/profile/Van-Thanh-Le/publication/344404205_Mobile_edge_cloud_computing_simulation_tool_survey_and_literature_reviews/links/5f71e6f1a6fdcc008643914a/Mobile-edge-cloud-computing-simulation-tool-survey-and-literature-reviews.pdf.