POLITECNICO DI TORINO

MASTER'S THESIS

Multi-Access Edge Computing for Automotive Applications

Author:
Giovanbattista CALIANDRO

Supervisors:
Prof. Claudio Ettore CASETTI
Dr. Fabio TOSETTO

Master's Degree in
Communications and Computer Networks Engineering

March, 2020

Abstract

Multi-Access Edge Computing for Automotive Applications

Multi-Access Edge Computing (MEC) moves the complexity of the network closer to the end user. With the advent of the fifth generation network (5G) this paradigm becomes very important because, together with new technologies, MEC enables a series of services and applications in which a real-time response is strictly required, such as: autonomous driving, robotics, safety etc. Moreover with new applications and Internet of Things the computation demand is destined to grow as well as the core network overload, edge computing can be a solution to face also this problem. This thesis goes into the actual state of the art defined by the European Telecommunication Standard Institute (ETSI) and looks at the several current projects of ETSI compliant MEC platform. They have been evaluated on the basis of cost and reliability to find the most suitable to develop and test automotive applications. The final choice led to OpenNESS, the Intel open source toolkit to create and deploy edge applications. However OpenNESS is a new software in experimental phase and has been tested on few types of hardware, this did not allow to obtain the complete result. To overcome the problem it has been realized a cloud machine with an architecture very similar to the OpenNESS one and through an experiment, the "In-Vehicle Entertainement" (IVE) it has been possible to find, in terms of latency, what is the best location to place this machine to retrieve multimedia contents from a user driving in Turin. The analysis of the collected data confirms that the best solution is the closest one. Finally it has been implemented from scratch a beta version of the See Through, a real automotive Use Case for which the implication of Multi Access Edge Computing is essential. With the knowledges get during the IVE experiment will be possible to run this application on the proper server to obtain the most MEC likely results until the ETSI compliant MEC platform will be available.

Contents

List of Figures

Chapter 1

Introduction

1.1 Context

In the Internet of Things (IoT) era all the physical objects are connected to the network in order to exchange data. This concept bring an exponential growth of the computational demand and, moreover, the several applications can have different requirements in terms of latency in order to be feasible and reliable. The advent of 5G allows to benefit of faster download and upload procedure, low latency and the execution of more applications simultaneously. Anyway this can be not enough, since, as we said before, the several applications require a very big amount of computational resources that, in many cases, lead to an overload of the data centers in the core network. The latency constraints, as well, will not be match if between the source of the data and the cloud there is a relevant physical distance to cover in both directions.
To face this problem we need to move the complexity of the network close to the end user. Adding complexity to the User Equipment (UE) increase the cost of each single device, so the more reasonable solution is to move part of the core network computational resources at the edge of the network, in this way is possible to eliminate the physical latency due to the distance to provide real-time services or, at least, elaborate partially the informations at the edge cloud and send through the core network a lighter amount of data, since it has been already processed, in part, at the network edge.
The challenge in edge computing is to bring Machine Learning (ML) and Artificial Intelligence (AI) from the cloud to the devices at the edge, or to even bring this to the actual sensors at the very edge. The specific challenge is to process data accurately and efficiently in environments with less computing power and storage capacity. Edge computing is not a new technology, but it is now starting to realise its true potential in the real-time data transfer from device to cloud and in real-time data processing at the device. As the number and frequencies of sensor signals increases, smarter algorithms will be required to efficiently process this explosion of sensor data. In Use Cases (UC) where low latency cannot be tolerated or where 100% connectivity should be guaranteed, bringing computing to the edge or the remote sensor is the solution. Between the several applications that requires the the presence of an

edge computing infrastructure there is the automotive one [18].
5G Automotive Association (5GAA) considers Edge Computing as one of the key supporting technologies for many of the foreseen V2X (Vehicle to everything) services for connected vehicles and for Autonomous Driving [2]. Edge Computing in the automotive industry is required to cope with the exponential growth of data in autonomous vehicles. As cars generate significantly more data every day, it is becoming a big challenge to process all that sensor data efficiently in the car and to transfer parts of that data to the cloud. In addition to that, safety related functions need to be available all the time and cannot rely for their functioning on wireless connectivity. For such needs, intelligent efficient edge computing comes to rescue[18].

1.2 The Company

Marelli Europe S.p.A is an international company founded in Italy in 1919, committed for the design and production of hi-tech systems and components for the automotive sector in the following business areas:

- Automotive lighting systems
- Body control system
- Powertrain control systems
- Electronic instrument clusters
- Telematics systems, and computers
- Suspension systems and components
- Exhaust systems
- Motorsport

The plant in Venaria Reale, Italy, also has several groups that work in the innovation field "Technology Innovation", one of them is the "Innovation Connectivity" in which I have worked, this group focuses its attention on all the aspects concerning connectivity for automotive application. In particular the team is focused on the Vehicle to everything (V2X) technology and its implementation with some of the most popular use cases for Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) application like, for example:

- **V2V**
  - Control Loss Warning (CLW)
  - Emergency Electronic Brake Light (EEBL)
  - Forward Collision Avoidance (FCW)
  - Left Turn Assist (LTA)

  - Intersection Movement Assist (IMA)
  - Stationary Vehicle Warning (SVW)

- **V2I**
  - Green Light Optimized Speed Advise (GLOSA)
  - In Vehicle Road Sign (IVRS)

1.3 Goal of the thesis

The aim of this thesis is to implement a Multi-access Edge Computing (MEC) platform for automotive applications and develop some sample applications to be deployed on the edge system. In order to do this we tried to provide all the theoretical concepts required to present in detail Multi-access Edge Computing as it has been standardized by ETSI, showing its architecture and Use Cases. We want also to show the current projects devoted to the implementation of an ETSI compliant MEC platform underlying their pros and cons in this research thesis context.

Chapter 2

Technologies Involved

2.1 5G

5G is the fifth generation wireless network technology. It allows much faster data download and upload speeds, wider coverage and more stable connections than the previous 4G. It is possible by making better use of the radio spectrum and enabling far more devices to access the mobile internet at the same time. Whereas 2G, 3G and 4G were primarily radio focused, 5G will represent an entire system with radio, a telecom core, and OSS all transformed to support new requirements.
The 1G networks transmitted data according to the analogue telecommunication standard, which required very voluminous devices due to the size of the lead-acid battery and of the receiver-transmitter module. This technology had a very low level of security, which therefore made interception and hacking possible. It appeared in 1980s and was then replaced in 1990s by 2G digital telecommunications. The main difference was that 1G networks were analog, while 2G networks were digital: the use of digital signals between telephones and base stations increased the capacity of the system since the voice signal could be compressed more efficiently than the analogue one.
The innovation of 2G was the complete encryption of transmission which prevented illegal interception, better spectral efficiency and the possibility of using services, data such as SMS. The power required for transmissions was less, so the battery life was greater, and the sound quality could be improved.
The 3G technology was introduced in 2001, it allowed the transfer of both “voice” data (digital phone calls), and “non-voice” data, for example downloading from the internet, sending and receiving emails and instant messaging. Among the services of greatest interest to users are, for example, the download of music files, the use of video services with user-generated content.
4G is the fourth generation network technology, introduced in 2012, which improved communication performance and helped to achieve more reliable coverage. 4G and 3G offered diversified levels of performance were reliable for basic web browsing and app use, but 4G outperformed previous technology in more complex functions. The substantial speed boost associated with 4G opened the door to better call quality, faster browsing, clearer video calls and even high definition mobile TV.

The first, widely cited proposal for the use of the millimetre wave spectrum for cellular/mobile communications appeared in the IEEE Communication Magazine of June 2011. The first reports on radio channel measurement that validated the possibility of using millimeter wave frequencies for urban mobile communication were published in April and May 2013 respectively in the IEEE Access Journal and in the IEEE Transactions on Antennas and Propagation. In the 5G technology the service area is covered by providers and divided into geographical areas called cells. All the devices located in a cell can communicate by radio waves with low power automated transceiver and a local antenna array. The local antennas are connected with the Internet and the telephone network by a wireless backhaul connection or a high bandwidth optical fiber. In 2014 the Next Generation Mobile Networks Alliance defined the following requirements for 5G networks: data rate of tens of megabits per second for tens of thousands of users, 1 gigabit per second simultaneously to many workers with offices in the same floor, several hundred thousand simultaneous connections for large and capillary wireless sensor networks, special efficiency significantly enhanced in comparison to 4G improved coverage, enhanced signal efficiency, significantly reduced lately compared to LTE. 2G, 3G and 4G cellular mobile systems occupy frequencies up to 3GHz, while frequencies between 3 and 6 GHz and dedicated to non-licensed N-LoS (Non-Line of Sights) systems, i.e. fixed wireless. Two areas of the spectrum above 6 Ghz stand out: the “microwave” spectrum and the “millimeter wave” spectrum. As known, the lengths of the radio waves are inversely proportional to the spectral frequencies: when the frequencies are very high beyond 60 GHz then the wavelengths are of the order of a millimeter. The frequencies between 6 and 43 GHz are called microwaves, are licensed and are partitioned in FDD (Frequency Division Duplex) with “narrow” channels of 56 MHz, or 112 MHz, the latter available only on 42 GHz. A technique used for increasing the data rate is massive Multiple Input Multiple Output (MIMO): each cell has multiple antennas communicating with the wireless device and received by multiple antennas in the device. In this way bitstreams of data will be transmitted simultaneously and in parallel. The new devices supporting 5G technology also have 4G LTE capability because the new networks use 4G for initially establishing the connection with the cell, as well as in locations where 5G access is not available.

2.2 V2X Communication

V2X stands for Vehicle-to-Everything communication. With this technology vehicles can communicate with the traffic system around them. It uses a short-range wireless signal to communicate with compatible systems, which is also resistant to interference and inclement weather. V2X systems are mainly used to increase safety and preventing collisions. The main types of V2X technology are: V2I (Vechicle-to-Infrastructure), V2N (Vehicle-to-Network), V2P (Vehicle-to-Pedestrian), V2D (Vehicle-to-Device), V2G ()Vehicle-to-Grid) and V2V (Vehicle-to-Vehicle). V2I technology

provides the driver with real-time infrastructure data such as road conditions, traffic congestion, parking availability, accidents. Similarly, vehicle and infrastructure data are used by traffic management supervision systems to set variable speed limits and increase fuel economy and traffic flow. All driverless cars are provided with this kind of technology. V2V communication allows vehicles to send messages to each other with information about what they are doing, through a wireless network. Information include speed data, location, braking, direction of travel and loss of stability. This kind of technology uses dedicated short-range communications (DSRC), Wi-Fi-like. V2V is a mesh network where every node can capture, send and retransmit signals. Some automakers have their own terms of V2V, which encompasses other vehicles and the infrastructure around them. The possible applications of V2X technology are: Road Safety services, aimed at avoiding accidents and improving road safety; Traffic Management & Efficiency services, designed to facilitate traffic flow; Infotainment & Business services, to create value for both the driver and passengers on board. Almost all Road Safety and most of the Traffic Management & Efficiency services require the periodic transmission of V2V and V2I messages with very low latency and high reliability.

2.2.1 802.11p IEEE

IEEE 802.11p is an approved amendment to the IEEE 802.11 standard to add wireless access in vehicular environments (WAVE), a vehicular communication system. It defines enhancements to 802.11 (the basis of products marketed as Wi-Fi) required to support Intelligent Transportation Systems (ITS) applications. This includes data exchange between high-speed vehicles and between the vehicles and the roadside infrastructure, so called V2X communication, in the licensed ITS band of 5.9 GHz (5.85–5.925 GHz). IEEE 1609 is a higher layer standard based on the IEEE 802.11p.[9] IEEE 802.11p is the basis for the dedicated short-range communication (DSRC). It is the standard supporting the GeoNetworking protocol for V2V and V2I communication.
IEEE 802.11p-enabled stations start sending and receiving data frames as soon as they arrive on the communication channel, using the wildcard BSSID in the header of the frames they exchange. The data confidentiality and authentication mechanisms provided by this standard and its amendments cannot be used and must be provided by higher network layers. To work with V2V and V2I communications, it uses WLAN technology which is known in US as Wireless Access in Vehicular Environments (WAVE) and in Europe as ITS-G5. The communication takes place through messages with very low data volumes, i.e. Cooperative Awareness Messages (CAM), Decentralised Environmental Notification Messages (DENM), Service Requested Message (SRM), Basic Service Message (BSM), etc.

2.2.2 C-V2X 3GPP

In the context of the path towards 5G, the 3GPP has defined a solution capable of responding both to the requirements of direct communications between vehicles and other road users, and to the needs of services that require a wider dissemination of messages and hence the use of the network. C-V2X technology stands for Cellular V2X and works through cellular networks, the C-V2X solution of 3GPP intends to bring the benefits of the entire mobile ecosystem and in particular of 5G to the automotive world. In 2015, the study of the adaptation of the LTE network to support V2X applications (Cellular V2X, C-V2X) began in 3GPP, with a view to specifying an adequate transport for both V2V, V2I messages and V2N messages to guarantee the complete coverage and continuity of service. The goal of 3GPP is to support the development of new business opportunities for the telecommunications industry by arriving at a system with performance capable to support all levels of driving autonomy, from assisted driving (SAE Level 1) up to the most challenging and complex one of autonomous driving (SAE Level5), according to an incremental phase approach. LTE V2X radio access was introduced by 3GPP in Release 14. The characteristics of V2X communications required changes to the LTE radio interface, according to the requirements to be met. In particular, a key element that guided the work of the 3GPP is the presence or absence of the network in the areas in which to communicate. This led to the specification of two communication modes: V2V and V2I communications based on the PC5 interface and V2N communications. V2V and V2I communications interface connects two devices directly, without first passing through the network: the signal transmitted by one device is received directly by the other devices.This typology is further divided into two modes: Mode 3 and Mode 4. As shown in Figure 2.1, Mode 3 communications take place in network coverage and the base radio station manages radio resources in order to maximize radio performance. Mode 4 is required in areas without cellular coverage (but can also be used in the presence of the network); in this mode the devices directly manage the use of radio resources. V2N communications based on the Uu interface connects the

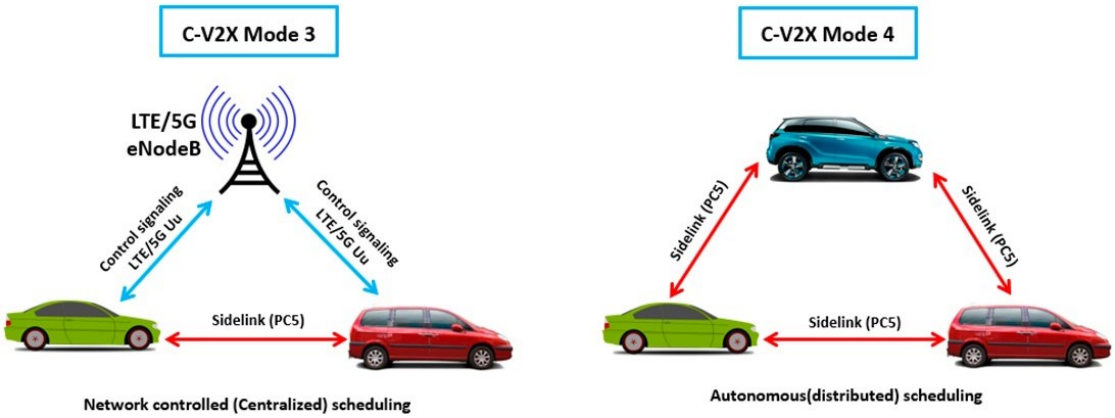

FIGURE 2.1: C-V2X Transmission Mode [1]

devices with the LTE base station (eNB), thus operating as in normal communications and also being able to carry out V2N2V and V2N2I type communications.

2.3 Multi-access Edge Computing

Multi-access Edge Computing (MEC) offers application developers and content providers cloud-computing capabilities and an IT service environment at the edge of the network. This environment is characterized by ultra-low latency and high bandwidth as well as real-time access to radio network information that can be leveraged by applications. MEC provides a new ecosystem and value chain. Operators can open their Radio Access Network (RAN) edge to authorized third-parties, allowing them to flexibly and rapidly deploy innovative applications and services towards mobile subscribers, enterprises and vertical segments. [11] Edge computing will allow multiple mobile networks to more easily manage the transfer of the massive amount of data for Virtual Reality (VR) and Augmented Reality) AR requirements. The low latencies will serve to make technologies react to situations in real time. In some cases, as in self-driving cars, a few milliseconds of delay in data transfer could have fatal consequences. Through edge computing, data in large volumes are efficiently processed near the source: this allows to reduce the use of Internet bandwidth, eliminate costs and ensure effective use of applications in remote locations. In addition, the ability to process data without ever transferring it to the public cloud adds a useful level of security for sensitive data.
The MEC server platform consists of a hosting infrastructure and an application platform. The MEC hosting infrastructure consists of hardware resources and a virtualization layer. The details of the actual implementation of the MEC hosting infrastructure, including the actual hardware components, are abstracted from the applications being hosted on the platform. The MEC application platform provides the capabilities for hosting applications and consists of the application's visualization manager and application platform services. [16]
Communication between applications and services in the MEC server is designed according to the principles of Service-oriented Architecture (SOA). Mobile-edge Computing transforms base situations into intelligent service hubs that are capable of delivering highly personalized services directly from the very edge of the network while providing the best possible performance in mobile networks. Proximity, context, agility and speed can be translated into unique value and revenue generation, and can be exploited by operators and application service providers to create a new value chain.

2.3.1 Functional Splitting

A monolithic functional implementation such as the Base Transceiver Station (BTS) is an example of why the current mobile supply chain is outdated. With this implementation, operators must pick one vendor per market and harmonize the macro

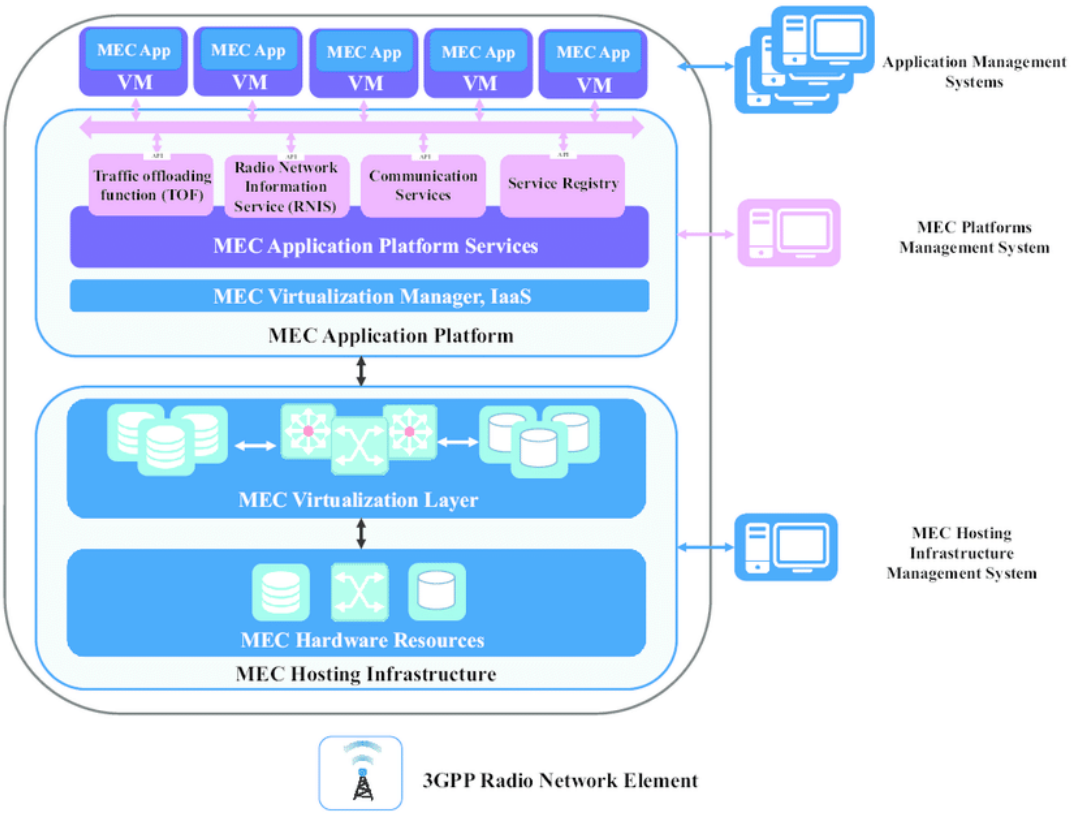

FIGURE 2.2: MEC Overview [19]

vendor markets to a "lowest common denominator" set of features. The result is a limited set of applications that operators can provide to their customers.The new cloud Radio Access Network (RAN) architecture addresses, among other things, the challenges of building multivendor networks and harmonizing to a common feature set. One fundamental characteristic is the decomposition of the radio signal processing stack using standardized splits [3]. The 5G decomposes and disaggregates its functionalities enabling the passage to an architecture that implies the use of an edge infrastructure that combines the handling of the subscribers with the access functionality. The decomposition of the Radio Access Network from the core lead to a logical intermediate location. This intermediate location separate the CU workloads and the User Plane Function (UPF) from the mobile core. Having distributed UPFs helps the offload, enable local virtualized services or more efficient peering at a metro level. [3]. The unified platform approach supports infrastructure workloads and a variety of service-oriented workloads. These workloads can support business-to-business (B2B) services such as tenancies offered to other businesses, and business-to-consumer (B2C) services in support of the operator consumer business. RMN's (Rakuten Mobile Network) innovative edge architecture uses vRAN, Control and User Plane Separated (CUPS) packet core, and distributed telco cloud. It enables MEC for both infrastructure functions and a variety of low-latency and content-centric services. Examples of such services include optimized content delivery, live TV, connected car, augmented and virtual reality, on-line gaming, connected

stadiums, and more. While others are looking at similar possibilities afforded by 5G, RMN will tap into these opportunities with both 4G and 5G to deliver the best possible user experience.[3].

Chapter 3

ETSI-MEC

ETSI is the European Standards Institute for Telecommunications. From 2015 until now has published several white papers in which has presented the concept of edge computing considering its integration in an 4G/5G network infrastructure, has standardized the MEC architecture to let the coexistence of many network operators and has provided the guide lines to deploy applications.
The general reference architecture provided by ETSI is the one depicted in Figure 3.1.

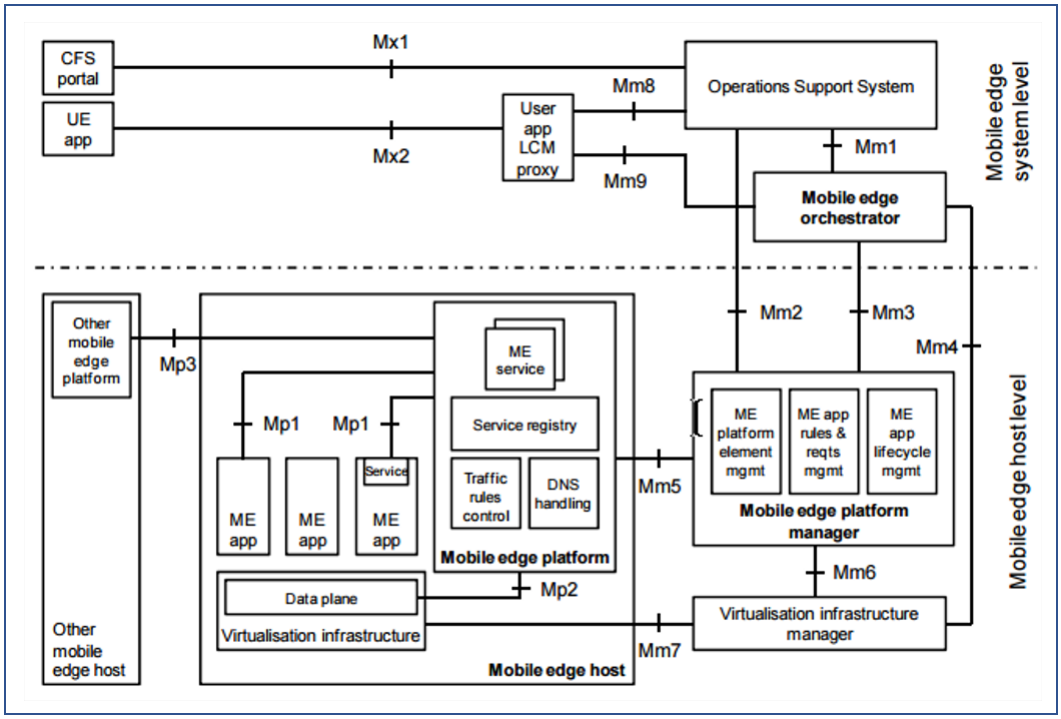

FIGURE 3.1: MEC Architecture [10]

The architecture shown represents all the functional element of the mobile edge system. These functional elements are interconnected through three groups of Reference Points. Each of them has a specific task.

- Reference points regarding the mobile edge platform functionality (Mp)
- Management reference points (Mm)

- Reference points connecting to external entities (Mx)

The system depicted is divided into two parts, the Mobile Edge Host and the Mobile Edge Orchestrator essential to run mobile edge application within an operator network. These two fundamental entities are connected each other through the Mobile Edge Host Manager that act as bridge between the host and the orchestrator. [10]

3.1 Mobile Edge Host

As we can see from the Figure 3.2 the Mobile Edge Host consists of: the Mobile Edge Platform and the Virtualization Infrastructure that provides to the Edge Platform all the network resources together with the computing and storage resources. The Virtualization infrastructure, moreover, contains the data plane that is in charged of maintains all the traffic policies received from the mobile edge platform and to route the traffic toward services, applications, local and external networks. On top of the Virtualization Infrastructure, instead, we have the several Mobile Edge Applications. [10]

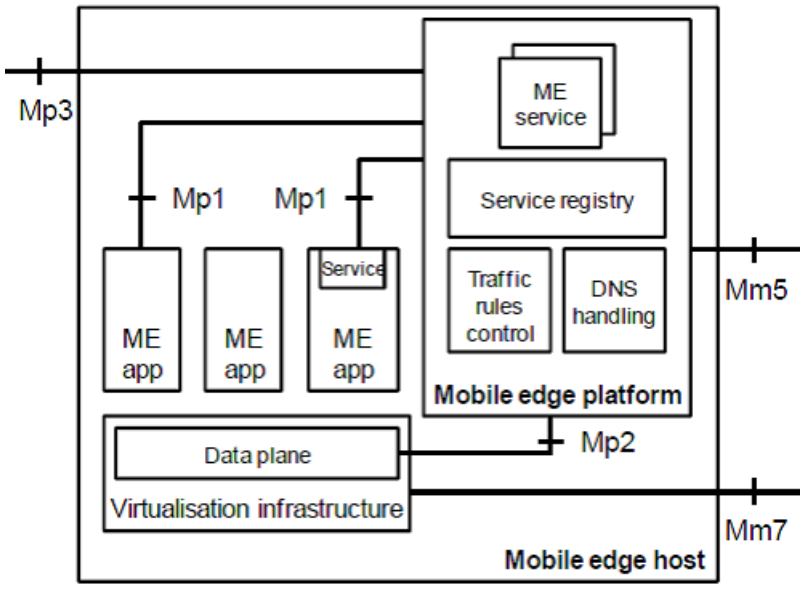

FIGURE 3.2: Mobile Edge Host [10]

- **Mobile Edge Platform**
  The Mobile Edge Platform is responsible for the following functions:
  - It offers to the mobile edge applications an ecosystem to discover, advertise consume and offer mobile edge services, furthermore, if supported, it allows to use mobile edge applications and mobile edge services of other mobile edge platform.

  - It receives the traffic rules from the entities connected, such as the Mobile Edge Platform Manager, Mobile edge application and services the traffic policies and instruct the data plane accordingly. Moreover, when supported, it allows to translate the tokens representing the UEs received with the traffic policies into coherent IP addresses.
  - It handles the configuration of the DNS server/proxy using the DNS records received from the mobile edge platform manager.
  - It hosts the mobile edge services.
  - It gives the possibility to access to a persistent storage and to get the information about date to allow synchronization.

- **Mobile Edge Application**
  As anticipated before the Mobile Edge Applications runs typically as Virtual Machines (VM) or, as we will see in the next chapters, as Dockers on top of the Virtualization Infrastructure provided by the Mobile Edge Host. Its main purpose is to interact with the Mobile Edge Platform to produce and consume Mobile Edge Services. Mobile Edge Applications need a certain number of requirements to be useful and reliable. These requirements can be either devoted to latency necessity or to the quantity of resources. The Mobile Edge System Level Management is in charged of validate these requirements or reject. In the case which the requirements are not asked the mobile edge system level management entity can assign a default value. Sometimes Mobile Edge Applications interacts with the Mobile Edge Platform to integrate support procedures of the application lifecycle management including availability and relocation if needed. [10]

3.2 Mobile Edge system level management

This foundamental entity consists of the Mobile Edge Orchestartor and the Operations Support System (OSS). They have a crucial role in the interaction between the Mobile Edge System and the end users.[10]

- **Mobile Edge Orchestrator**

  - It has a complete view of the Mobile Edge System that includes all the deployed Mobile Edge Hosts with their services and applications, the available resources and the selected network topology.
  - It has a direct connection with the Virtualization Infrastructure Manager to allow the handling of the applications. It is also responsible for the security operations, including the checking for the integrity and authenticity of the application on-boarded packages and the validation of the Mobile Edge Application rules that the Mobile Edge Orchestrator can also modify to let them be compliant with the operator policies.

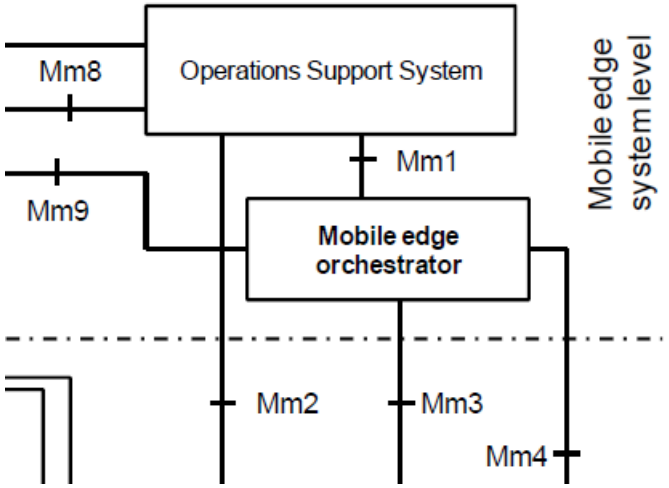

FIGURE 3.3: Mobile Edge System Level Management [10]

  - Having the complete view of the Mobile Edge System it can evaluate on the basis of the constraints required by the application, like maximum latency or required resources, on which Mobile Edge Host is worth to deploy the application.
  - It triggers the instantiation and the termination af applications
  - When supported, it manage also the application relocation

- **Operations Support System (OSS)**
  The Operations Support System (OSS) refers to the OSS of a network operator. It is connected with with the User Equipments (UE), and so with the onboarded User Application, and receives through the CFS portal requests for the instantiation, termination and, when supported, relocation of the applications. It decides whether to grant or not these requests. The granted requestes are forwarded to the Mobile Edge Orchestrator for a further processing. [10]

3.3 Mobile Edge host level management

This is mostly an interconnection infrastructure between the Mobile Edge Host and the Mobile Edge Orchestrator rather than a Mobile Edge System entity. It includes the Mobile Edge Platform Manager and the Virtualization Infrastructure Manager that act as pass-through between the elements of the two Mobile Edge System entities. [10]

- **Mobile Edge Platform Manager**
  The mobile edge platform manager is responsible for the following functions:
  - It manages the Mobile Edge Applications lifecycle and tells to the Mobile Edge Orchestrator if there are relevant events regarding the application status.

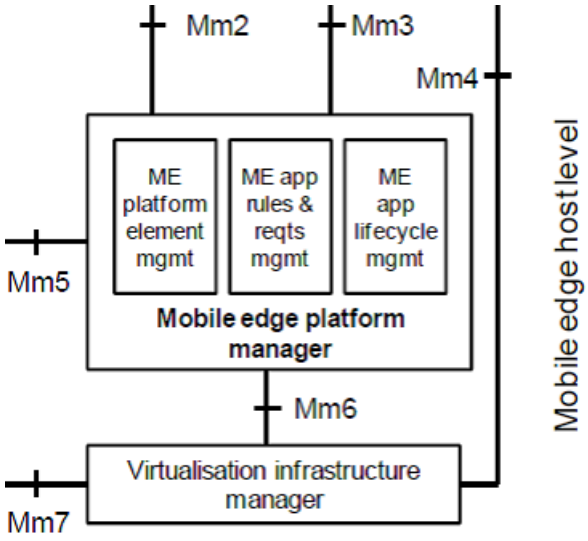

FIGURE 3.4: Mobile Edge Host Level Management [10]

- It gives to the Mobile Edge Platform the resources and functions to manage the applications.
- It manages the rules regarding traffic, authorization of services and DNS configuration of the deployed applications.
- It is also responsible to get all the informations regarding errors faults and status from the Virtualization Infrastructure. These informations are processed later if needed.

- **Virtualization Infrastructure Manager**
  - It allocates and manage the virtual resources like storage and computing requested by the Virtualization Infrastructure.
  - It is in charged of having a Virtualization Infrastructure ready to execute a software image. The Virtualization Infrastructure is considered ready if it is configured and able to store a software image.
  - It provides fast applications if they are supported.
  - It looks at the Virtualization Infrastructure to get all the informations about occurred faults and performances evaluation.
  - If application relocation is supported, it is responsible to communicate with external cloud manager to trigger the relocation of the application from one system to another.

3.4 Reference Points

We have three groups of reference points, they are discriminated on the basis of the Mobile Edge System entities that they connect. Below there is a complete list of reference point that can be found on Figure 3.1

- **Reference points related to the mobile edge platform [10]**
  - **Mp1**
    The Mp1 reference point is located between the Mobile Edge Platform and the Mobile Edge Applications. It aims at the provision and the consume of specific service functionality. In particular it gives the communication support for services enabling services discovery and registration. Moreover it support the application relocation, when included, manages the access to the storage resources and time information. Takes also care of the DNS configuration/activation and the application of traffic rules.
  - **Mp2**
    The Mp2 reference point is located between the Mobile Edge Platform and the Data Plane of the Virtualization Infrastructure. The only thing that ETSI told us about this reference point is that is used to steer traffic between network, applications and services.
  - **Mp3**
    The Mp3 reference points is located between the Mobile Edge Platforms of different Mobile Edge Systems and is used to manege the communication between them
- **Reference points related to the mobile edge management [10]**
  - **Mm1**
    The Mm1 reference point is located between the Mobile Edge Orchestrator and the Operations Support System. It is used to trigger the instantiation and termination of the applications in the Mobile Edge System.
  - **Mm2**
    The Mm2 reference point is located between the Operation Support System and the Mobile Edge Platform Manager. It is used to configure the Mobile Edge Platform and manage its performance and error recovery.
  - **Mm3**
    The Mm3 reference point is located between the Mobile Edge Orchestrator and the Mobile Edge Platform Manager. It is used to handle the applications in their lifecycle and rules. Moreover it keeps track of the current available mobile edge services.
  - **Mm4**
    The Mm4 reference point is located between the Mobile Edge Orchestrator and the Virtualization Infrastructure Manager. It is used to store and manage the informations about the available virtual resources for the relative Mobile Edge Host and to handle the application images.
  - **Mm5**
    The Mm5 reference point is located between the Mobile Edge Platform

Manager and the Mobile Edge Platform. It is used to manage and configure the Mobile Edge Platform and its applications from the Mobile Edge Platform Manager. In particular it is also in charge to support the application lifecycle management and relocation.

- **Mm6**
  The Mm6 reference point is located between the Mobile Edge Platform Manager and the Virtualization Infrastructure Manager. It is used to handle the virtual resources in order, for instance, to implement the application lifecycle management.
- **Mm7**
  The Mm7 reference point is located between the Virtualization Infrastructure Manager and the Virtualization Infrastructure. This reference point has not been futher specified by ETSI, the only thing that we know in that it is used to handle the Virtualization Infrastructure.
- **Mm8**
  The Mm8 reference point is located between the Operations Support System and the User application. It is used to interact with the User Equipment listening they request for running and stopping applications in the Mobile Edge System. Also this reference point is not further specified.
- **Mm9**
  The Mm9 reference point is located between the Mobile Edge Orchestrator and the User Equipments. It is used to handle the Mobile Edge Applications when they are requested from a User Equipment application.

- **Reference points related to the external entities [10]**
  - **Mx1**
    The Mx1 reference point is located between the Operations Support System and the Customer Facing Service (CFS) portal. It is used by third-parties to trigger the instantiation of Mobile Edge Applications in the Mobile Edge System.
  - **Mx2**
    The Mx2 reference point is located between the User Application Lifecycle Manager Proxy and the User Equipments application. This reference point to be available must be supported by the Mobile Edge System and the network infrastructure. It is used to let the User Equipments ask for running or moving an application in or out the Mobile Edge System.

3.5 Deployment

As its definition suggests the MEC host should be located at the edge of the network but it can be collocated also in the core network if we are sure of the designed packets route. The MEC host is always coupled with the User Plane Function (UPF) that

is in charge of steering traffic toward the selected MEC application. The choice of the place to deploy the UPF and the MEC host is done by the network operators that evaluates where is the best location on the basis of applications and users requirements like maximum delay or the estimated/evaluated load etc.
The MEC orchestrator helps the management of the network resources and dynamically choose on which MEC host is better to deploy the Mobile Edge Applications. In the Figure 3.5 are shown the possible physical deployment of the MEC host and the relative UPF, the decision of what is the best solution depends on the target results in terms of performances and security.

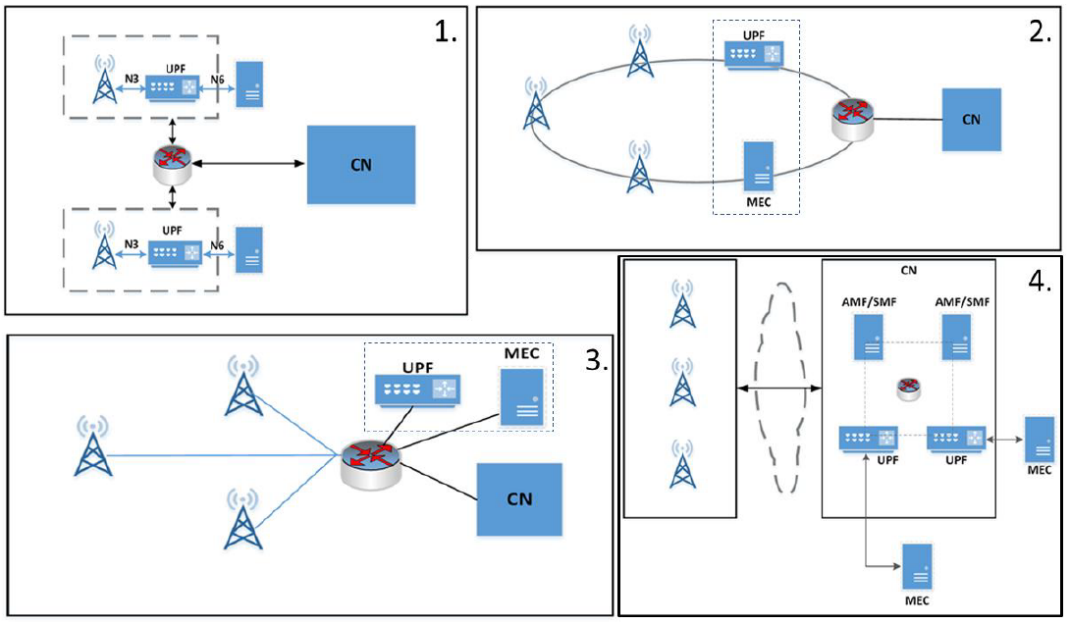

FIGURE 3.5: MEC possible deployment [15]

1. MEC and the local UPF are located together with the Base Station (BS)
2. MEC is located with a transmission node and, if is possible, with a local UPF
3. MEC and the local UPF are located in a network aggregation point
4. MEC is located within the Core Network functions (i.e. in the same data centre)

As we already anticipated the Figure 3.5 shows how the MEC can be deployed in several network location, from the edge to the core. Anyway a MEC host located in the core network lose its properties of reduce the physical latency due to the distance. Common in all the deployment option is the presence of a UPF that is used to route the traffic toward the network or toward the target MEC host application.[15]

3.6 Example of MEC ETSI Compliant

Amazon Web Service, Vodafone and Saguna have realized and presented together this PoC (Proof of Concept) at the MWC18 (Mobile Word Congress). They have combined the Vodafone's LTE nework infrastructure, the Saguna's ETSI compliant MEC platform and the Amazon Web Service Greengrass's machine learning application to implement the possibility of driver monitoring using the images retrieved from a frontal smart camera in the vehicle.
The PoC idea is that a driver can install a frontal camera on his car that takes video frames and send them to the nearest processing point. In the LTE Vodafone network we have the Saguna MEC platform that hosts Amazon Web Service Greengrass for a fast collection and a smart processing of the frames. In this way is possible to send, almost in real time, an alert to the driver if the processing result detects that he is distracted. This kind of implementation helps the car manufacturers and suppliers to increase the security of cars and, primarily, of their drivers. The real time feature can be reached thanks to the presence of the MEC host in the edge network that allows to obtain low latency in the exchanging of information between the vehicle and the application. Moreover this solution avoids to have complex and costly devices and application on-boarded in the car to serve this purpose moving the complexity on the edge host.[17]

3.6.1 Solution Architecture

In the Figure 3.6 is shown the high layer architecture used to implement this demo. Below we will detail all the involved participants.

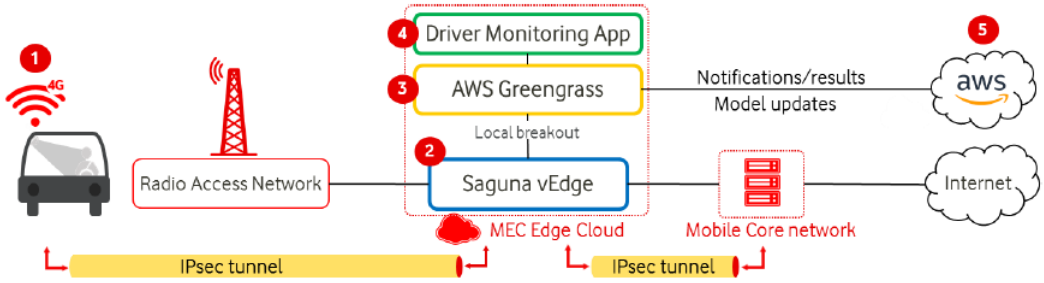

FIGURE 3.6: PoC Architecture [17]

1. **The in-car device**
   The User Equipment used is a Raspberry Pi equipped with Pi Camera. It is connected with the mobile network and continuously streams the video frames captured of the driver.

2. **Saguna vEdge**
   It is the implementation of the ETSI MEC platform. It takes into account the traffic policies to filter the received stream (e.g. matching the typical 5-tuple flow identifier, matching IMSI/APN). The Saguna vEdge redirects the

received video frames to the coupled instance of Amazon Web Service Greengrass instance implementing the "local breakout" that is a mechanism to stop the streaming session at the MEC edge cloud avoiding to go through the mobile core network.

3. **AWS Greengrass Core**
   It is deployed in the MEC edge cloud and work since the frontal camera is registered in the same Amazon Web Service Greengrass Group. It is an example of Mobile Edge Application that thanks to the traffic rules implemented in the Mobile Edge Platform receives the traffic from the radio access network and using the Amazon Web Service Lambda functions is able to monitor the driver and send back the alert (if necessary) to him almost in real time.

4. **The driver monitoring application**
   This application uses a neural network to process the received frames and detect if the driver is assuming a dangerous behaviour for him or for the other road users (talking or texting on the phone in this proof of concept).

5. **AWS cloud**
   The Amazon Web Service cloud is used to train the machine learning model used in this proof of concept. In particular it sends updated model toward the Amazon Web Service Greengrass instance. moreover it receives from the latter notifications and results that can be useful to train the other connected applications.

The communication between the driver monitoring application and the in-vehicle application does not rely on the traditional client-server paradigm, but, instead, follows a three-tier model since it has the MEC Edge Cloud in the middle that acts as an intermediate. The process of machine learning is then splitted in three location as depicted in Figure 3.7:

- The source of data in the in-vehicle device (Raspberry Pi)
- The processing of information and the quick response at the edge cloud network using the Saguna MEC host and the Amazon Web Service Lambda functions
- The training in the cloud where the computational capabilities are more powerful

This type of model is very useful to obtain the target results in terms of latency and reliability. Moreover it helps very much the application developers since they have to approach with an ecosystem that offers:

- The same type of environment both at the edge and the public cloud

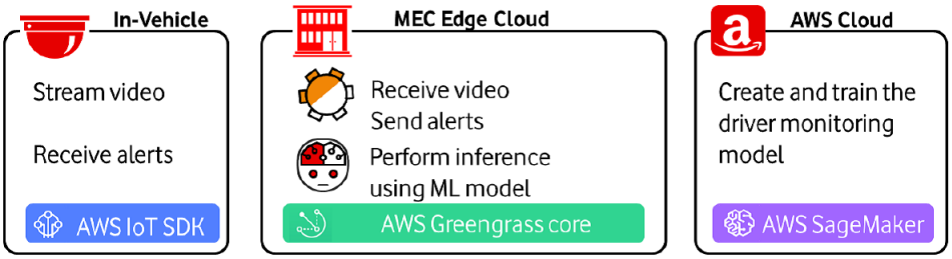

FIGURE 3.7: Three-Tier Module [17]

- The MEC platform that exploiting that using its traffic rules in able to steer the traffic from the user application to the cloud and vice-versa through the LTE/5G network
- An easy way to deploy useful application on the MEC platform

Chapter 4

Use Cases

In this section will be explained the several Use Cases enabled by the presence of an edge infrastructure. In the first part we will see the Use Cases that cover the daily challenges of real-time applications. In the second part, instead, we will move the focus on the automotive applications in which is essential a MEC server to have a reliable service.

4.1 General Use Cases

4.1.1 Active Device Location Tracking

This Use Case provides the user coordinates with an accuracy level that depends on the network real-time measurements like: Global Navigation Satellite System (GNSS) that measures the transmitting time of GNSS signals from four or more satellites to give the position, Received Signal Strength Indicator (RSSI) that, instead measures the received signal power to provide the location and many other advanced measurements. By looking at the MEC architecture defined previously the location service is registered and discovered over the Mp1 reference point. In this scenario, a MEC application can ask only for the raw data since the coordinates and all the other requested informations can be computed locally using locationing algorithms. The MEC can also provides all the informations regarding the UEs (User Equipments) in a specific area and their in or out movement [7].

4.1.2 Augmented Reality Content Delivery

This Use Case aims to provide on the basis of the user geo-location and interest Augmented Reality (AR) contents. Today, the AR contents are too heavy to be processed by the end user and at the same time is not possible to do it in the cloud as the latency will be too high. The MEC applications acts as pass-through node to provide the geo-location information of the user with a specific accuracy and to cache the AR contents for a much faster delivery and improve the Quality of Experience (QoE). Using the AR argument is also possible to negotiate the Guaranteed Bit Rate (GBR), the Maximum Bit Rate (MBR), packet loss and priority policy [7].

4.1.3 Video Analytics

This Use Case provides Video Analysis at the edge to avoid the overload of the core network and to have better results in terms of latency. Video Analytics is an end to end solution that is worth to provide video surveillance to cities, municipalities, and enterprises over an LTE/5G network. MEC server is in charge of analyzing raw video streams from surveillance cameras connected over the selected network infrastructure, and for forwarding all the relevant and target events to the control MEC applications [7].

4.1.4 RAN-aware Content Optimization

This Use Case aims at the provisioning of RAN-aware contents, to do this the MEC application requests first of all the cell and the end user radio interface status. The advantage is an improvement of the QoE and the network efficiency in terms of throughput and video quality. Using the congestion window and the load status provided by the MEC host mobile operators can adapt their transmission policy to these informations in order to reduce the latency and the overhead. To do this MEC application needs to provide the interest topics (identity and RAN) informations to MEC services [7].

4.1.5 Distributed Content and DNS Caching

This Use Case relies on the conventional web browser caching system to speed up the distribution of frequently requested contents on demand, live streaming and services, to avoid the blackhaul congestion and improve the QoE. The contents are cached on the MEC server for a specified period of time. This paradigm according with a BT TSO Research & Technology is able to reduce of 35% the core network overload. Moreover the DNS (Domain Name System) caching at the same time, can reduce the web page download time by more or less 20% of the needed time [7].

4.1.6 Application-aware Performance Optimization

This Use Case takes care of improving the network efficiency and the QoE in terms, for instance, of buffering time, and audio video quality individually for each high layer application (Skype, YouTube, etc...). Operators, in order to improve the QoE can apply Deep Packet Inspection (DPI) to discriminate each traffic flow and provide the switching table and the routing policies. Thus to implement this Use Case we need to identify first of all the application type and the application control protocol, and then we need also to know the arguments regarding the requested RAN (Radio Access Network) and the application quality argument in order to allocate the new QoS (Quality of Service) policy on the application [7].

4.2 Automotive Use Cases

The Use Cases for the automotive sector have been divided into four categories that represents the challenges they deal with. The four categories, that will be detailed in the following paragraphs are : Safety, Convenience, Advanced Driving Assistance and Vulnerable Road Users.

4.2.1 Safety

This is the first (Relevant to MEC) group of automotive Use Cases, they combines the Vehicle to Vehicle (V2V) and the Vehicle to Infrastructure (V2I) communication to provide road safety to all the users.This type of use cases was specifically listed in the US DOT NHTSA publication 2016-0126 and ETSI TR 102 638 [8].

- **Intersection Movement Assist (IMA)**
  The purpose of this Use Case is to warn a user if there are other users incoming from a lateral direction. The application calculates the potential of a collision and advises the driver with progressive urgency [8].

- **Queue Warning**
  A queue of vehicles represents in many situations a potential danger beyond produce a lot of traffic delay for instance when a turning queue extends to other lanes. Using this Use Case is possible to provide to users in advance informations about the queue. At a very basic level, a Queue Warning System works by monitoring the speed of vehicles along a stretch of roadway and then provides warnings to approaching drivers when a slow-down is detected [8].

4.2.2 Convenience

This group of Use Cases includes all the software updates an the telematics services requested for the V2X. They can be implemented using the existing access technology and in part are already supported by the car manufacturers. This group of V2X Use Cases requires an high communication cost with the back end server, so it is ideal to be deployed on a MEC server. This group of Use Cases in also in charge of manage the health of the vehicle [8].

4.2.3 Advanced Driving Assistance

Advanced Driving Assistance represents one of the most challenging requirements for V2X. This group of Use Cases offer the technologies to help the driver avoiding accident. It requires that a large amount of data must be transfered with high reliability and low latency. The users moving along the road must have the possibility to receive reliable prediction of what will happen if they continue their maneuver. The Advanced Driving Assistence group of Use Cases include also the two described below:

- **Real time situational awareness and high definition (local) maps**
  Real time situational awareness is an essential requirements for autonomous driving, in particular to signal changing in the road conditions or sudden traffic queue. Moreover an high definition local map must be downloaded from a backend server, to do this, of course, we need that this map should be cached in an edge deployed server.
  The Use Case for real time situational awareness and high definition (local) map should not be seen as an application to manage slow and isolated changing in the road condition but to aggregate and distribute locally a set of useful informations about the actual conditions using RSU (Road Side Unit), that are all the road infrastructure like semaphores. An alternative to edge computing in order to satisfy these requirements is to let the several users themselves create and manage these informations retrieved by the peer users, but, as we can imagine, they need a powerful and reliable equipment. However edge computing keep be the best solution, also because, it allows offloading computation and is able to broadcast a much larger amount of data [8].

- **See-Through (or High Definition Sensor Sharing)**
  This Use Case will be explained better in the following chapters. The concept is that vehicles such as truck, minivans or cars shares the images of their frontal camera with the vehicles behind them. In particular this Use Case can be fundamental during a passing maneuver to warn the vehicle that is approaching the overtake of possible hazards along the road as well as to show if there is an incoming vehicle in the opposite direction to prevent fatal head-on collisions. The forwarding video together with the analysis of the vehicles data to detect the head-of-the-line and the overtaking vehicle requires to deal with a big amount of data in a small amount of time, thus a MEC server in strictly required [8]

4.2.4 Vulnerable Road User (VRU)

This group of Use Cases cover all the other road users, such as pedestrians and cyclists, exploiting their smartphones and tablets. A crucial point in this situation is to have a real-time accurate information about the position of these users in the urban environment. With this hypothesis the presence of an edge computing infrastructure is essential. The cooperation of VRUs and vehicles in general is crucial for the users safety and to avoid accidents [8].

Chapter 5

Frameworks and projects related to MEC

In this section will be presented some of the most relevant platforms that implement an ETSI compliant MEC platform. We will look in detail to their architecture and we will choose the most suitable for our purpose, taking into account that these are still open project that relies also on the users contribution to be improved in terms of scalability and reliability.

5.1 Mosaic 5G

The first platform we will analyze is the one proposed by Eurecom, that is called Mosaic5G.
Mosaic 5G combines together the concepts of SDN (Software Defined Network), NFV (Network Function Virtualization) and MEC (Multi-access Edge Computing) to provide an open source environment for testing. In particular, as we can see from the Figure 5.1,
Mosaic5G consists of five software components that cover the required network

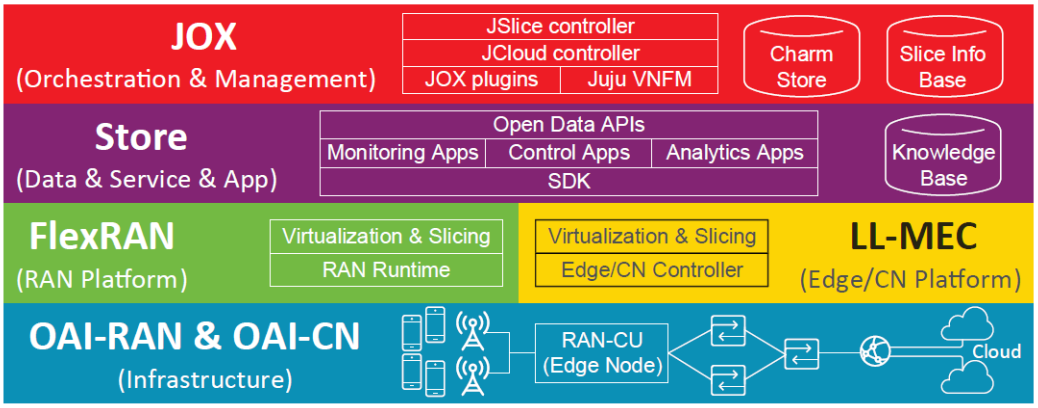

FIGURE 5.1: Mosaic5G ecosystem [12]

layers, from the physical one to the application one and both the MEC entities, Host and Orchestrator:

- **JOX as the service orchestrator**
  JOX implements the concept of Mobile Edge Orchestartor, it is a Juju based software for the virtualized network that integrates network slicing allowing to devote different network resources to each slice depending on its requirements. Using JOX, thus, we can optimize between each network slice the resource configuration, Virtual Network Function and service chains. JOX runs on top of the Juju Virtual Network Function Management (VNFM) and exploit a series of plugins to interface with the FlexRAN, the LL-MEC and Virtual Infrastructure Manager (VIM). JOX can be deployed either in a 4G or 5G network environment since it is compatible with both infrastructure. By looking more in detail JOX as we can see from the Figure 5.2 we can discriminate between two main components:

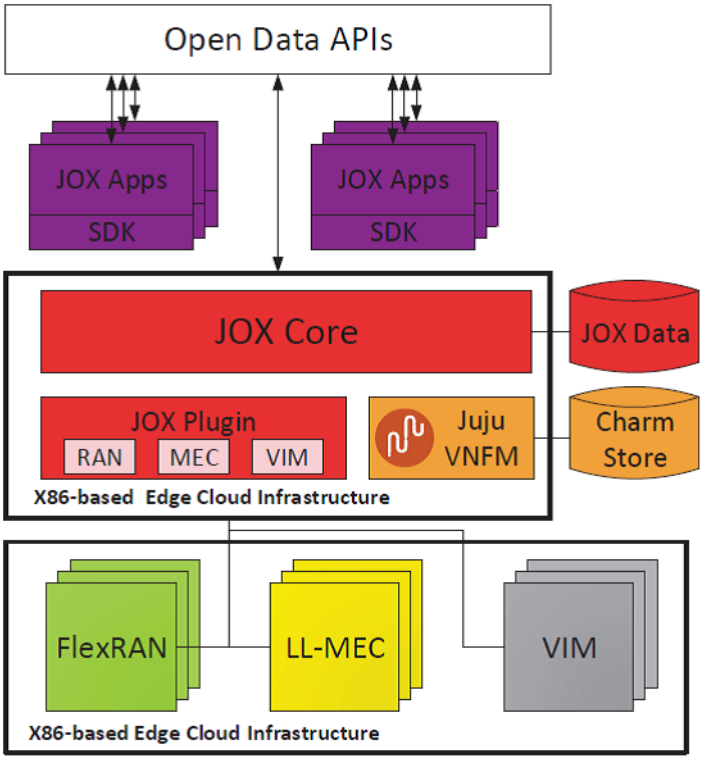

FIGURE 5.2: JOX Components [12]

  - **Jox Core**
    It comprises the JOX Slice functionality that represents each slice as a reference model with already specified traffic policy, and the JOX Cloud feature to control both the slicing and the cloud resources.
  - **JOX plugin framework**
    It consists of a series of different plugins that allows to interact with the other components of the system, such as: FlexRAN, LL-Mec, Core Network and the Virtual Infrastructure Manager. Moreover it exposes the

north-bound REST API to provide to each network slice connected to a JOX Cloud the configuration and utilization.[12]

- **Store as the repository of applications and dataset**
  The Mosaic5G Store is set of repositories that contains applications and control applications, like the ones in Figure 5.3 useful to test the Mobile Edge System. in particular it consists of: datasets, models, control applications, SDK (Software Development Kit) etc. Its main purpose is to develop plug-and-play network applications devoted to specifics Use Cases and, moreover, to adapt the network service delivery platform to reuse the already existing applications. The control applications has their specific field to control and moreover they can provide APIs to control other control applications. Each of them take informations from the SDKs platform with specific level of detail. Datasets are a way to aggregate network informations that are processed later to discover eventual unpredicted behaviors. The control applications relies on this already existent datasets to provide offline services or can produce new records in real-time. Using decision making algorithms on these datasets we are able to validate some hypothesis or to take decisions on the basis of the acquired knowledges, in this sense they are worth for the control applications to have a flexible way to operate. Furthermore using the open data APIs, applications can share their new knowledge with other applications and, at the same time, receive these informations from the others. The Mosaic5G store contains also snaps for the other softwares that compose the system, such as: FlexRAN, JOX, LL-MEC, Open Air Interface Core network. It contains also the Charms, a set of YAML configuration and installation files to implement different Use Cases [12]. These Charms can be found for instance at: JAAS

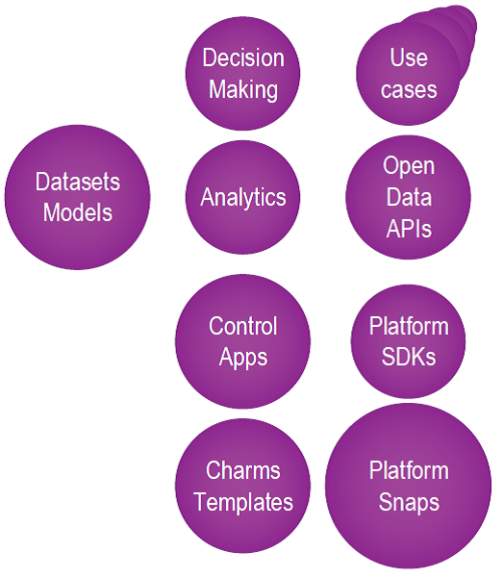

FIGURE 5.3: Mosaic5G Store [12]

- **LL-MEC as Core Network (CN) and edge controller**
  The LL-MEC (Low-Latency), in Figure 5.4 represents the MEC host of the Mobile Edge System. It uses the concept of SDN to divide the user plane from the control plane both at the edge and the core network in order to enable the defined MEC features. The LL-MEC in connected to an Open Virtual Switch (OVS) and through the OpenFlow traffic policies and rules is possible to abstract the user plane in order to be analyzed, monitored and programmed to have a flexible and dedicate control. Moreover, as we said before, in the store there are SDK that allows to have an ecosystem where is possible to deploy edge applications. The LL-MEC is compliant with the standard defined by ETSI and includes also the Mp1 and Mp2 reference points discussed previously. In particular the Mp1 reference point enables the low latency application requirements using REST and Core APIs and message bus. The Mp2 reference point, instead, is in charged to tell to the user plane the best way to steer the traffic between networks, services, applications etc. From the LL-MEC we can benefit principally of two services:

  - **Edge Packet Service (EPS)**
    It allows to control the traffic policies in a statical or dynamical way and, moreover manages the OpenFlow libraries and the Open Virtual Switches.
  - **Radio Network Information Service (RNIS)**
    It takes from the Radio Access Network real time informations on the actual status and send them to the control plane to take a decision.

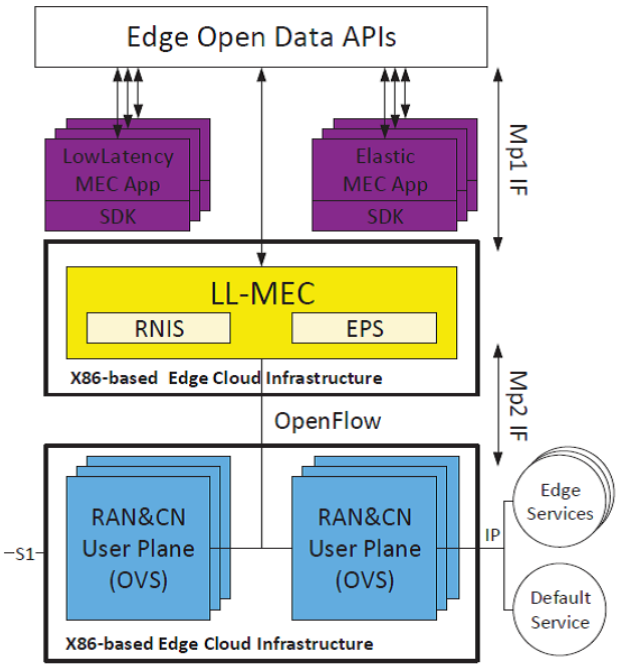

FIGURE 5.4: Mosaic5G LLMEC [12]

- **FlexRAN as the real-time Radio Access Network (RAN) controller**
  The FlexRAN platform is one of the first open source software defined Radio Access Network that that support itself the division between the user plane and control plane functionality. Furthermore, since it can be associated to multiple Base Station, exploiting the SDN principles is possible to centralize the control of the environment or to have a distributed control among different infrastructure. The FlexRAN, thus, can offer a control framework with a set of control functions that allow to monitor and consequently manage the informations in the Radio Access Network domain. As we can see from the Figure 5.5 the FlexRAN consists of two main components:

  - **Real-time controller (RTC)**
    It allows to control several Base Station connected to the FlexRAN, it provides also the primitives and SDKs to manage the applications.
  - **RAN runtime**
    It is controlled by the Real Time Controller and is in charged of the virtualization of the radio access network resources, supply the SDKs to manage the applications and pipelines the RAN service function chain. It is also implemented to support the network slicing by looking at the specifics requirement for each slice. Moreover the communication protocol between the RTC and the RAN runtime in able to evaluate updated statistics, configuration and reconfiguration ad move the control applications [12].

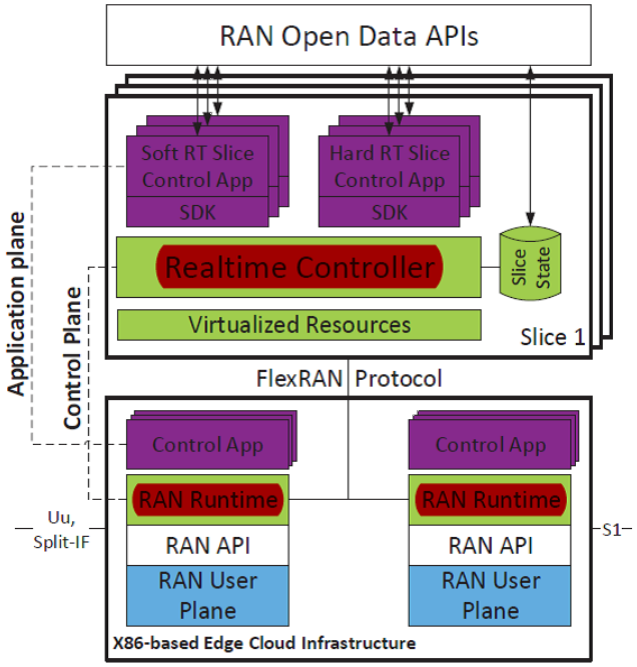

FIGURE 5.5: FlexRAN [12]

- **OpenAirInterface (OAI) RAN and OAI CN as 3GPP compliant implementation of LTE/LTE-A feature**
  Open Air Interface (OAI) Software Alliance (SA) produced by Eurecom is a software that provides an emulation of the core networks features compliant with the 3GPP standard. It comprise the access network (EUTRAN) environment as well as the Evolved Packet Core (EPC) allowing the user to work in both network portions.

5.2 Saguna

Saguna, "the Multi access Edge Cloud Computing pioneer", is a company that helps the network operators to improve, optimize and accelerate their networks. It has solutions that covers both the edge and the cloud requirements and is compatible with any type of access network, from the wired to the cellular one. Their products span all the use case applications, including connected cars, healthy, virtual and augmented reality, Geo location, Internet of Things, autonomous driving for cars and drones, distributed DNS service and many others.
The Saguna MEC solution is compliant with the ETSI standard and 3GPP 5G requirements. It allows to simplifies the implementation and the management of the edge platforms as well as the development and deployment of edge applications. The Saguna MEC supply Ultra Reliable and Low Latency Communication (URLLC) that exploit the existing LTE network also to enable the 5G requirements if chose as designed network infrastructure.

5.2.1 Saguna MEC Starter Kit

The Saguna MEC Starter Kit is the Saguna solution to Multi-access Edge Computing, it provides an environment ready to be used for deploy edge platforms and develop edge applications. It provides also sample application target to be tested in the edge ecosystem to help the developments of demo and Proof of Concepts.
The MEC Starter kit consists of:

- **Saguna vEdge**
  The Saguna vEdge offers all the requested virtualized resources to implement a cloud service environment that allows the applications to operate at the edge of the network, closer to the end User Equipments. In order to obtain these results the Saguna Edge Cloud provides all the traffic services to connect the end users to the edge applications. The Saguna vEdge has also a complete view of the network adapting its policies to the network access conditions. It is possible to add new services that can be integrated with the existing applications to encounter the market needs [14].

- **Saguna OMA**
  Saguna OMA (Open Management and Automation) adds scalability features

automating the 'collective of cloudlets' operations. It is also possible to integrates, since is supported, the possibility to have a centralized orchestration rather than a distributed one and a management system.

- **Sample Edge applications**
  The MEC starter kit has also a series of sample applications like CDN and IoT applications that helps the developer to provide quickly new application integrating the already existing ones.

- **Support Package**
  Support and training package provided by Saguna's team of MEC experts.

5.3 Linux Foundation

Linux Foundation is a no-profit organization that aims at the acceleration of Linux growth providing a complete series of services to compete with the closed source platforms.

5.3.1 EdgeXFoundry

EdgeX Foundry is the Linux Foundation solution for Multi-access Edge Computing. It is composed by a series of microservices and SDK tools that provides the minimal Mobile Edge System features. These SDKs and microservices was originally written in java but actually they have been transposed in C or Go languages.
EdgeX Foundry allows to containerize these microservices using Docker, a technology that will be explained better later, and have them available into the source repository.

5.4 OpenNESS

The Open Network Edge Service Software (OpenNESS) is the Intel toolkit designed to develop and deploy applications at network edge and on-permise exploiting the concept of Docker. The OpenNESS toolkit offers an open source environment that helps the software developer to implement and deploy edge applications and services. It can be integrated both in the LTE and 5G networks infrastructures [4].
It is composed by the two main entities defined by ETSI:

- The **Edge Controller** as Mobile Edge Orchestrator
- The **Edge Node** as Mobile Edge Host

5.4.1 Edge Controller

OpenNESS Controller Community Edition is the Mobile Edge Orchestrator and MEC Platform Manager defined by ETSI MEC Multi-access Edge Computing: Framework

and Reference Architecture. It is composed by a series of microservices that enable the orchestration of several edge nodes and the management of application lifecycle. The Edge Controller microservices are deployed in three containers:

- **Web-UI**
- **Controller API backend**
- **Database**

5.4.2 Edge Node

OpenNESS Edge Node is an ETSI compliant implementation of the Edge Host. As well as the Edge Controller it consists of a series of microservices that allows the edge deployment. These microservices are: ELA, EVA, EAA that can be deployed together or in separate containers, Syslog, DNS server and NTS dataplane that, instead must run in separate containers.
The OpenNESS edge Node microservices allows to execute the applications locally on the edge node or forward the traffic toward other edge nodes connected on a Local Breakout [4]. Below is possible to see in summary the four Edge Node containers:

- **edgenode_appliance_1** −ELA, EVA, EDA and EAA
- **nts** −Dataplane NTS (not present when OVS is used as dataplane)
- **mec−app−edgednssvr** −Edge DNS Server
- **edgenode_syslog−ng_1** −Syslog

5.5 Comparison between frameworks and selection of the most suitable

All the analyzed frameworks are effective ways to implement Mobile Edge Computing environments. Anyway the most suitable for our purpose is **OpenNESS** since it is very easy to be installed and configured and is accessible to all the users, moreover, since it is a new, revolutionary software it has an active community and a support system that helps to fight the issues encountered during its utilization. Mosaic5G is another useful solution but we give up to use it because it needs Open Air Interface to run, that is a software very huge and difficult to be configured, mostly with a limited amount of time available during a working thesis. The other solution that we tried to consider in our experimentation has been Saguna, but it requires a partnership collaboration to be delivered that is, again, a concept non very appropriate for a thesis. The platform implemented by Linux Foundation, instead, during the analysis was not very complete in its explanation, as is possible to see from the dedicated paragraph so we never real take it into account.

Chapter 6

OpenNESS

6.1 Introduction

OpenNESS is the Intel platform to enable orchestration, developing and deploying edge applications. It is compliant with the Multi-access Edge Computing ETSI standard architecture. OpenNESS allow the applications to be published on the platform and for other applications to exploit those services. Moreover it is able to implement a multi-platform, multi-access and multi-cloud architecture by using the major edge orchestration frameworks such as Kubernetes and OpenStack. The services running on the platform can span all types of Use Cases such as: Computer Vision, radio network information, V2X and safety applications. OpenNESS is multi-access in the sense that it is access network agnostic since it is able to interoperate with: wired, WiFi, LTE and 5G networks. OpenNESS provides also APIs to let the orchestrator or the controller to configure routing policies in a uniform manner. It consists of two separate entities, the controller and the edge node. Both softwares need CentOS as operative system to run and they are composed of a series of dockers. In particular as we can see from the figure we have four dockers for the edge node and three dockers for the controller [4].

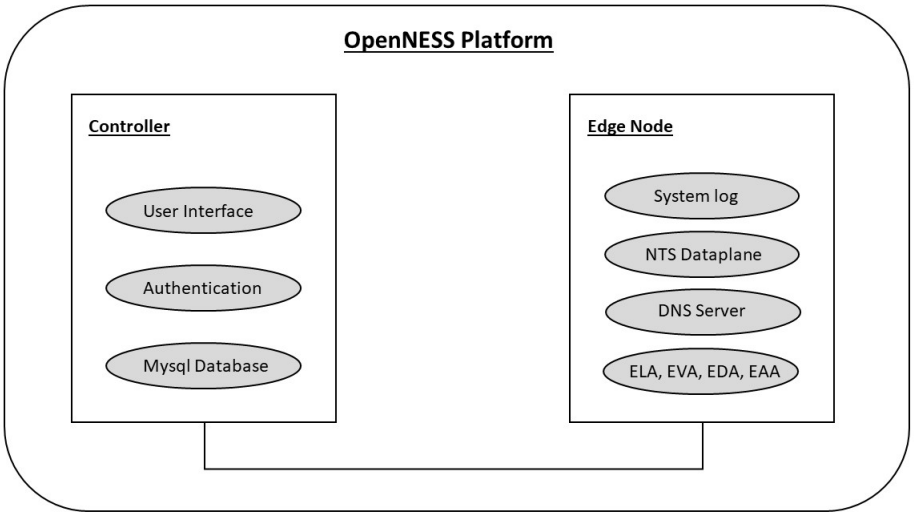

FIGURE 6.1: Controller and Host Dockers

6.1.1 CentOS

CentOS stands for Community Enterprise Operating System is a Linux distribution that provides a free, community-supported computing platform of class enterprise functionally compatible with its upstream source, Red Hat Enterprise Linux (RHEL)

6.1.2 Docker

Docker is a set of PaaS (Platform as a Service) products to develop, ship and deploy applications. The main idea of Docker is to separate the application from the infrastructure and manage them independently. In this way is possible to deliver software quickly reducing the delay between writing software and running it in production. The applications and their dependencies are packaged in a loosely environment called containers. They provide isolation and security allowing to run simultaneously more containers on the same host and even on the same virtual host. Containers, indeed, are more lightweight than Virtual Machines since they don't need any hypervisor but interact directly with the kernel. Docker provides all the necessary tools to manage the container's lifecycle. The software that hosts the containers is called Docker Engine. Applications can be deployed in the production environment as containers or orchestrated service. In this way the production environment behaves as a local data center or a cloud provider [6].

6.1.3 Key Terminologies defining OpenNESS

- **Orchestration**
  Orchestration in the context of OpenNESS refers to exposing northbound APIs for Deploying, Managing, Automating the Edge compute cluster and Applications that run on the cluster. E.g. OpenNESS North bound APIs that can be used by Orchestrators like ONAP for managing the OpenNESS edge solution.

- **Edge Services**
  Edge Services in the context of OpenNESS refers both to those applications that serves end-user and those that serves other Edge applications. For instance, CDN (Content Delivery Network) is an Edge application target to end-user traffic whereas Transcoding is an example of application that provides services to another application (CDN in this case).

- **Network Platform**
  Network Platform in the context of OpenNESS are those nodes deployed in Network or On-Premise edge compute processing. These are typically open-source platform that can host both applications and Virtual Network Functions.

- **Access Technologies**
  Access Technologies in the context of OpenNESS refers to the several types of

network traffic with which it can interface, i.e. LTE (GTP/IP), wired (IP) and WiFi (IP).

- **Multi Cloud**
  Multi Cloud in the context of OpenNESS refers to support in OpenNESS to host multiple Public or Private cloud application on the same node or in the OpenNESS compute cluster. These cloud applications can come from e.g. Amazon aws greengrass, Baidu cloud etc.

6.2 Software Architecture

An OpenNESS subsystem consists of one or more OpenNESS Edge Nodes, and a controller that hosts OpenNESS Controller microservices, these microservices could also be integrated in to pre-existing cloud platform hosting edge services. OpenNESS reference edge architecture combines the cloud-native and Network Function Virtualization Infrastructure optimizations for Virtual Machine and Container cloud on COTS (Commercial Off-The-Shelf) Architecture (CPU,Memory,IO and Acceleration) taking into account the several open-source projects that has an essential amount of edge compute specific APIs to provide a complete platform for edge computing [4].

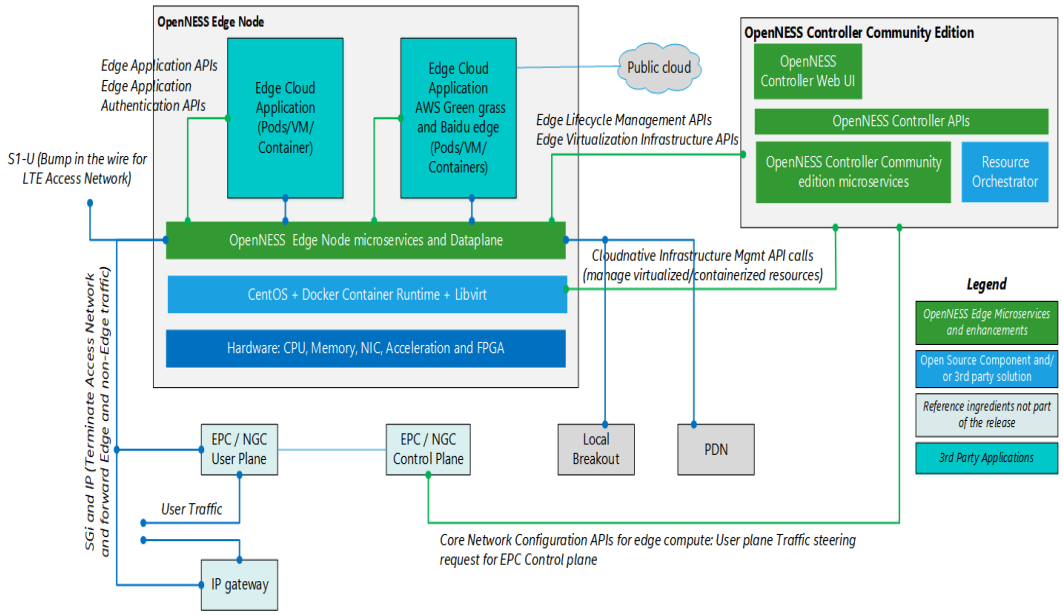

FIGURE 6.2: OpenNESS Reference Architecture [4]

- **Controller**
  As we already saw, the OpenNESS Controller consists of a series of microservices that enable edge orchestration and applications lifecycle management. These microservices in particular are: Web User Interface, Database and Controller API back-end. When the Controller coexists with an orchestrator like

Kubernets it uses the orchestartor APIs without duplicate the functionalities of its dockers for lifecycle management and traffic policies. As consequence we have two different types of deployment for the controller [4]:

  - **OpenNESS Native Deployment** (OpenNESS deployed using controller which interfaces NFV infrastructure directly, libvirt/docker runtime).
    In this deployment scenario all the services implemented in the controller docker must be used, from the management of the edge nodes and its applications to the core network configuration and the user account management.
  - **OpenNESS Infrastructure Deployment** (OpenNESS deployed using Kubernetes as an orchestrator).
    In this case we can exploit the Kubernetes (or other orchestrator) APIs, thus the microservices functionality are basically only devoted to Telemetry and core network configuration.

- **Edge Node**
  The edge node, host a set of microservices to implement edge deployment and computing. Also in this case depending on which deployment scenario we chose we can have all the microservices or only a subset running on the edge node.

  - **OpenNESS Native Deployment**
    In this deployment scenario we have microservices running directly on the edge node platform or the traffic is forwarded to applications running on connected platform on a local breakout. All the requested microservices run on the local edge node including the edge application traffic policy and edge node virtualization infrastructure.
  - **OpenNESS Infrastructure Deployment**
    In this case, as we saw before, we can exploit the Kubernetes components to use only specifics containers running on the edge node. In particular with this deployment scenario, we need only the DNS docker and the enrollment/authentication functionalities.

Both these platforms have been implemented through a wide use of Go and C program languages [4].

6.3 System Architecture

In this section will be shown the several system architectures we used to have a working OpenNESS MEC platform.

As first attempt we tried to install and configure the two OpenNESS entities (controller and edge node) on two Virtual Machines running on the same physical machine as shown in Figure 6.3.

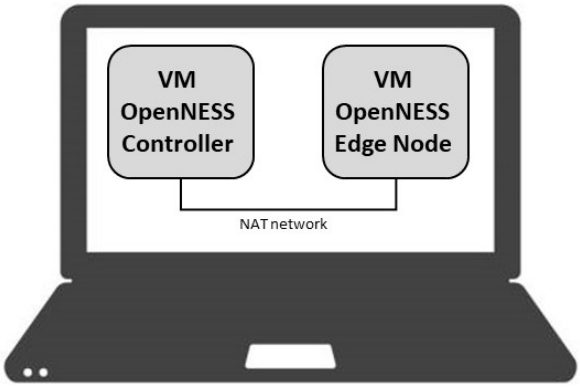

FIGURE 6.3: Virtual Machines Configuration

We knew in advance that this kind of configuration was not supported but it was not specified the reason. Thus, both to acquire familiarity with the software (it was the first approach to OpenNESS) and to give our contribution to the research we decide to try with this configuration. Through the NAT network the two virtual machines was able to communicate to each other and also to communicate with internet. The installation and configuration of the controller was successful as Virtual Machine. Unfortunately we didn't get the same result with the Edge Node. In particular, after the execution of the ansible scripts and the rebooting the virtual machine was no more able to start for a problem with the service manager *systemd* and to be more specific with *journald*, a possible solution was to try another service manager but this could be only a resource wasting since the Virtual Machines configuration was not supported.

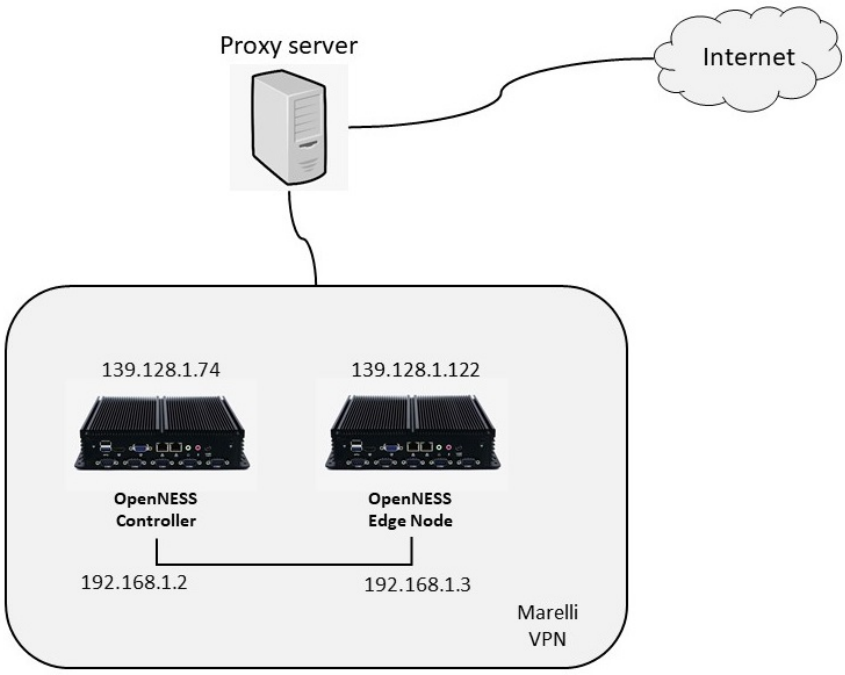

FIGURE 6.4: Native Deployment on Marelli VPN

As logical consequence we moved to a configuration in which the two entities have been installed on two separate physical machines. In the scenario represented in Figure 6.4 we can see that both the controller and the edge node are on the company VPN (Virtual Private Network) and as all the entities on the Marelli VPN need a proxy server to go out on Internet, in particular we used *proxytrn* on the port 8080. Moreover for the inter-communication between the controller and the edge node we have an Ethernet end-to-end link. We followed all the passages for the installation of the system behind a proxy, that implies adding the proxy settings to all the files that manages the connection. The installation in this case terminated successfully for both the entities but due to the restriction of the company network we were not able to map the interfaces of the edge node on the controller. The last solution, rep-

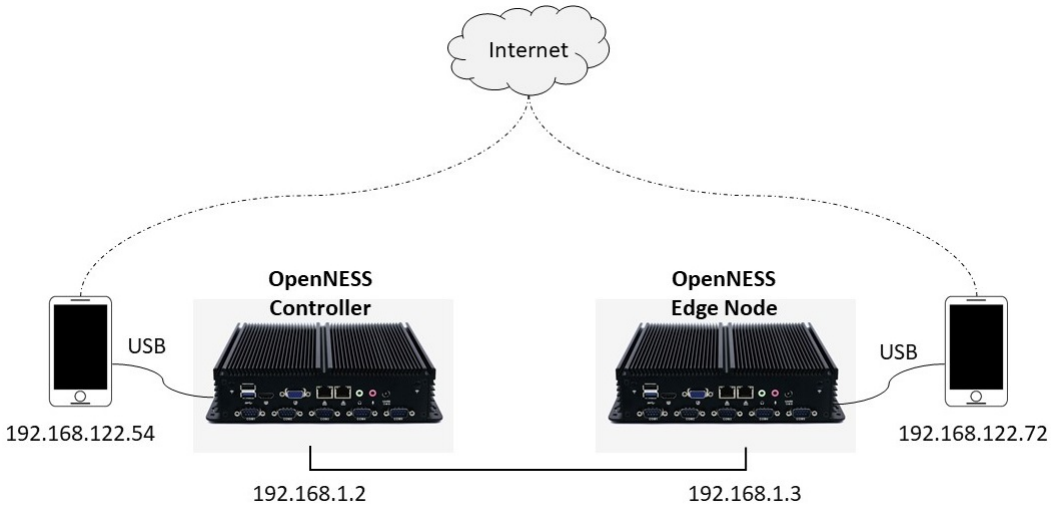

FIGURE 6.5: Final Deployment

resented in Figure 6.5, removes the controller and the edge node from the company VPN and give them the access to the network using the USB tethering. Moreover they are connected each other, again, using a dedicated Ethernet link. The routing table has been configured so that they have to use the USB tethering to reach the network and the Ethernet link to communicate locally. With this configuration everything work as expected and we were able to see the interfaces of the edge node on the controller but unfortunately when we configured these interfaces and we commit the changes the edge node docker dedicated to the connection goes down blocking all the system.

6.4 Installation and Configuration

In this section we refer to the last system architecture, Figure 6.5, that represent the actual state of the system. The controller and the edge node have been installed on two car-pc with the hardware in the Figure 6.6 and both with CentOS as operative system, the controller with the complete version, while the minimal one was used on the edge node, as recommended in the HowTo guide. For the specifications you

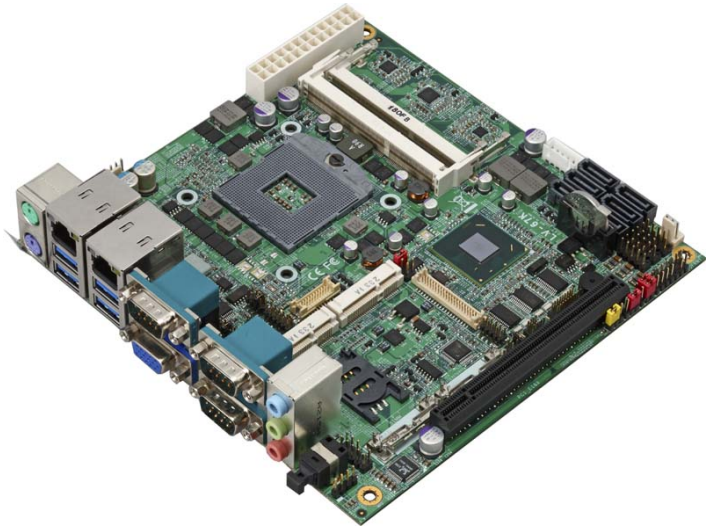

FIGURE 6.6: CAR-PC Motherboard

can refer to the following link: Motherboard LV-67K
The mobile phones through which we enable the USB tethering were instead two Samsung Galaxy S10. The installation of the controller and the edge node is managed by the ansible scripts, we need only to edit the configuration files; in particular for the controller we have to configure the IP address, the access port, username and password to guarantee the login using a browser. For the edge node we have to edit a file with the authentication key of the controller and its IP address, then run the automatic scripts. Before completing the last script the edge node prints a key on the screen that must be added on the controller interface to be recognized and let the installation complete. At this point we have the interfaces of the edge node mapped on the controller that ca be edited adding the traffic policy. When we try to commit the changes, however we obtain an error message that tell us we can not connect to the edge node as shown in Figure 6.7 by investigating on the reason on that failure we found that the edge node docker devoted to the network communication exit as soon as we commit the changes. In Figure 6.8 and in Figure 6.9 is possible to see the screens that shows the state of the two entities dockers. As we can see while the controller dockers work properly the NTS docker of the edge node exit and even if we start it manually we don't obtain the desired result. We asked to the support noting that it was a common problem and it was due to the Ethernet device that must support DPDK that is a software for fast packet processing in data plane applications. Unfortunately this answer comes very late for our purposes, since we were already moved to a backup solution very similar in its architecture with the OpenNESS edge node that allow us to go on in our experimentation.

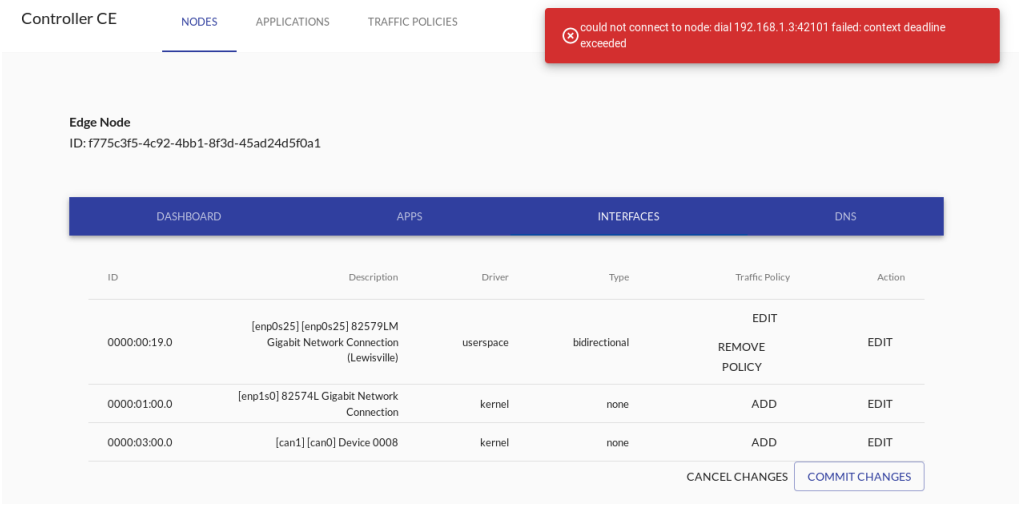

FIGURE 6.7: Controller User Interface

```
[root@localhost edgecontroller]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
                    NAMES
0c3e9d2d83ea        ui:latest           "docker-entrypoint.s…"   6 days ago          Up 14 minutes       0.0.0.0:3000->80/tcp
                    edgecontroller_ui_1
6ba0e9ffcf6b        cce:latest          "/cce -adminPass mar…"   6 days ago          Up 14 minutes       0.0.0.0:6514->6514/tcp, 0.0.0.0:808
0-8081->8080-8081/tcp, 0.0.0.0:8125->8125/tcp   edgecontroller_cce_1
e342cbfc0f2a        mysql:8.0           "docker-entrypoint.s…"   6 days ago          Up 15 minutes       33060/tcp, 0.0.0.0:8083->3306/tcp
                    edgecontroller_mysql_1
```

FIGURE 6.8: Controller Dockers

```
[root@localhost edgenode]# docker ps -a
CONTAINER ID        IMAGE                     COMMAND                  CREATED             STATUS                      PORTS
                    NAMES
a8305764f1f9        appliance:1.0             "./entrypoint.sh"        24 hours ago        Up About a minute           192.168.122.1:80->80
/tcp, 192.168.122.1:443->443/tcp, 0.0.0.0:42101-42102->42101-42102/tcp   edgenode_appliance_1
f73331f2b881        nts:1.0                   "/root/entrypoint.sh"    24 hours ago        Exited (1) 58 seconds ago
                    nts
64e164b56386        edgednssvr:1.0            "/bin/sh -c './edged…"   24 hours ago        Up 35 seconds
                    mec-app-edgednssvr
1876d18c9bc2        balabit/syslog-ng:3.19.1  "/usr/sbin/syslog-ng…"   24 hours ago        Up 6 seconds                601/tcp, 192.168.122
.1:514->514/udp, 6514/tcp   edgenode_syslog-ng_1
```

FIGURE 6.9: Edge Node Dockers

6.5 Backup Solution

To overcome the problems had with OpenNESS we moved on a backup solution that involves the use of a cloud Virtual Machine exposed in the Azure's data center, that is the Microsoft public cloud to provide cloud computing services. The architecture of this Virtual Machine, as we can see from the Figure 6.10, is very similar with the OpenNESS one depicted in Figure 6.1, since it is based on Docker and has a set of containers dedicated to the Edge Node and another dedicated to the controller, the main difference is that the backup solution controller can not manage several Edge Node like the OpenNESS one. In this way the applications that we develop and deploy on this machine can be deployed also on OpenNESS when it will be available.

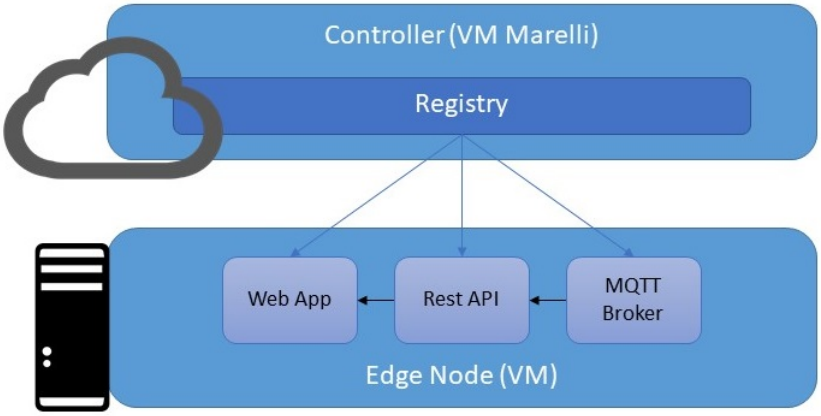

FIGURE 6.10: Backup Solution

Chapter 7

In-Vehicle Entertainment Use Case

As one of the strategic focus areas of 5GAA, Multi-Access Edge Computing (MEC) is a key technology offering cloud computing capabilities and an IT service environment at the edge of 5G mobile networks. Among other use cases, infotainment is a promising market for automotive. Drivers & passengers spend more than 300 hours a year in a vehicle. 5G infrastructure offers entertainment opportunities to optimize that time for riders.

7.1 Use Case Description

This demo, realized by Intel, together with Marelli, Terranet and Equinix, shows how MEC can support immersive high-definition (HD) entertainment for all occupants of a moving vehicle, including video streaming, gaming, virtual reality (VR), office work, online education, advertisement, etc.
The demo is implementing a real-time 5G emulated environment connected with commercial terminal and a MEC server. The availability of information about Predictive Quality of Service (QoS) can be delivered by MEC to better support immersive High-Definition (HD) entertainment for all occupants of a moving vehicle. The benefits for passengers will be sustained high quality entertainment over 5G networks utilizing predictive QoS and MEC.

7.2 Requirements Analysis and Architecture

This Proof-of-Concept demo features an In-Vehicle Entertainment (IVE) MEC App and Edge MEC Service developed by Marelli. As summarized in Figure 7.1, the MEC IVE App runs on a Marelli On-Board Unit (OBU) and serves high-definition video contents to passengers based both on their interests and car context (e.g. position, heading, speed, etc.) Video contents are retrieved from off-board services in the Cloud or at the Edge. As Edge we leverage an Edge Node provisioned by Equinix using OpenNESS – the open-source solution developed by Intel. The MEC App also monitors, shows and act upon key communications KPIs, namely network delay, bandwidth, buffering time. This is needed to dynamically switch among different MEC or cloud content providers and improve the IVE user experience. In general,

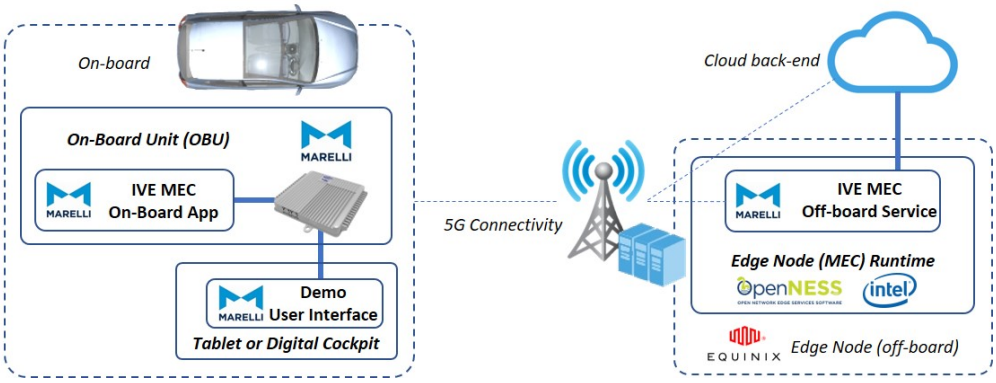

FIGURE 7.1: Companies Contribution

MEC Apps are suitable when the application requires low communication latencies, faster service response times, higher bandwidth and more dynamic and fine-grained geographical provisioning.

7.3 Development and Test

The test has been done in Corso Unità d'Italia, between Corso Maroncelli and Corso Spezia, adjacent to "Museo dell'Automobile", in Turin, as represented in the Figure 7.2.

FIGURE 7.2: Test Route

During the test cyclically we ping four hosts, the Edge one, and the other three located in data centers placed respectively in: Frankfurt, Amsterdam and United States as represented in Figure 7.3. Using these ping we measured five parameters:

- **RTT (Round Trip Time)**
- **Jitter**
- **Upload Bandwidth**
- **Download Bandwidth**
- **Time to First Frame**

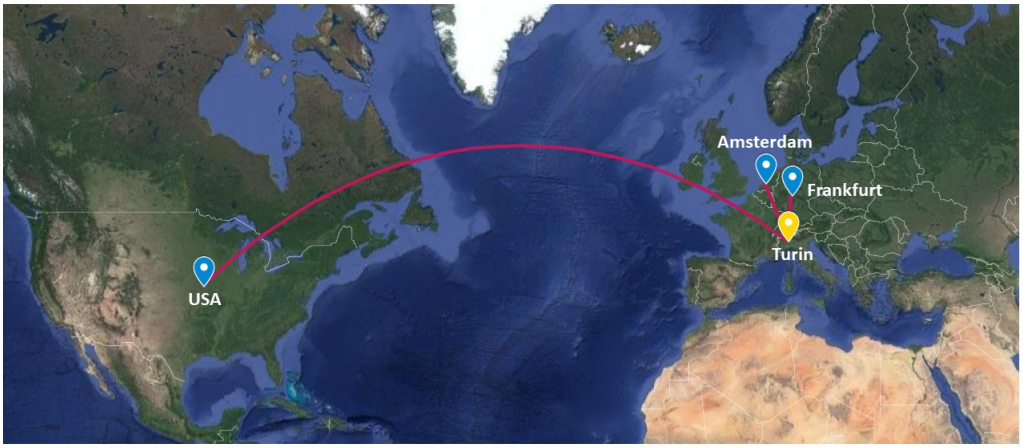

FIGURE 7.3: Virtual Machines Locations

In particular, the RTT has been measured through the tool TCPping of the library android-netdiag (github: https://github.com/qiniu/android-netdiag), this tool measures the duration of the three-way-handshake between client and server for the TCP communication.The jitter gives an idea of the RTT variation found as standard deviation af the last ten samples. The Time to First Frame is measured starting from the downlink bandwidth evaluated with speedtestlib as well as the uplink bandwidth. Since we didn't upload a real video, we basically did some assumptions get the results:

- we need at least 30 seconds of buffer (that's of course is not true, the buffering dimension depends itself from the available bandwidth and its variance)
- 30 seconds of video holds more or less 187.5 Mb
- we need about 30 RTT to take into account the several handshake of the video transfer protocols (TLS negotiation, codecs, etc..)

Of course for a real estimation we need to have more reliable assumptions and take into account the server load. But this was out of the scope of this Proof of Concept. In the Figure 7.4 is possible to see a snapshoot of the Web App GUI where we can check in real time the measured parameters, the vehicle state and the available topics.

FIGURE 7.4: User Interface Snapshot

The desk version of the demo is based on a re-play of actual statistics collected during test-drive experiments performed in November 2019 in Torino and its architecture is shown in Figure 7.5

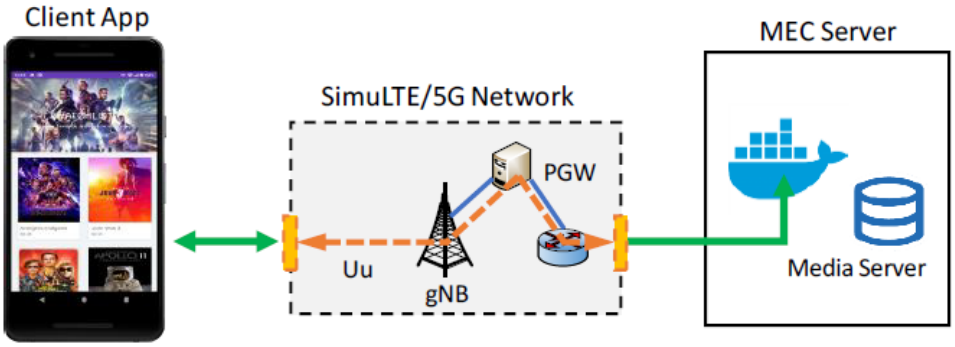

FIGURE 7.5: IVE Desk Version

7.4 Data Analysis

The first box that is possible to see in Figure 7.4, the green one, would have been the machine to use if we had OpenNESS but, since that machine was not available we have to chose the best between the other three by looking at the collected data.
As is possible to see from the Figure 7.6 and Figure 7.7 but, mostly, from the Table 7.1, we have better results in terms of latency (RTT, TTFF) with the server placed in Frankfurt, that is also the closest, even if the available bandwidth is less, as shown in Figure 7.8 and Figure 7.9. We could not tell it in advance since we are not able to predict the route of the packets when they have to cross the core network, but this experiment confirms that the closer is the cloud, the better is the latency. In this way until OpenNESS will be available we know which is the reference data center to obtain the most MEC likely results.

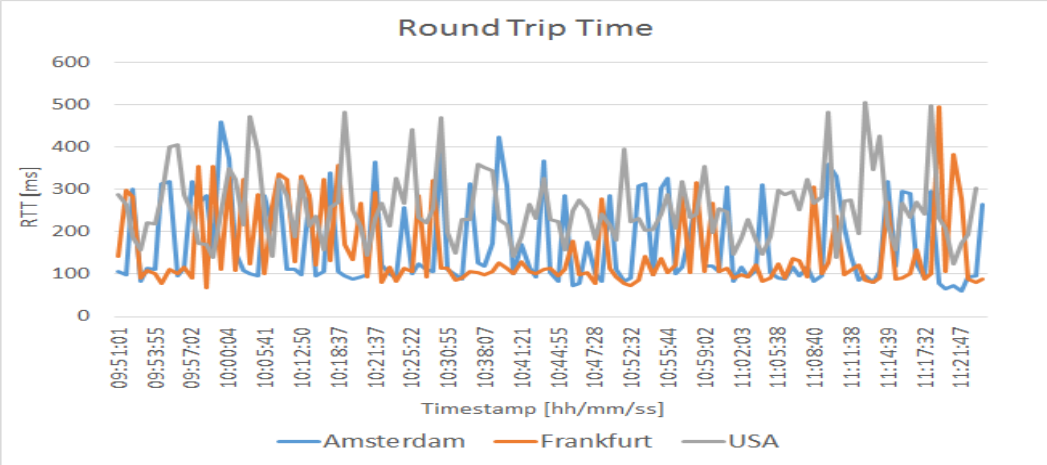

FIGURE 7.6: Round Trip Time

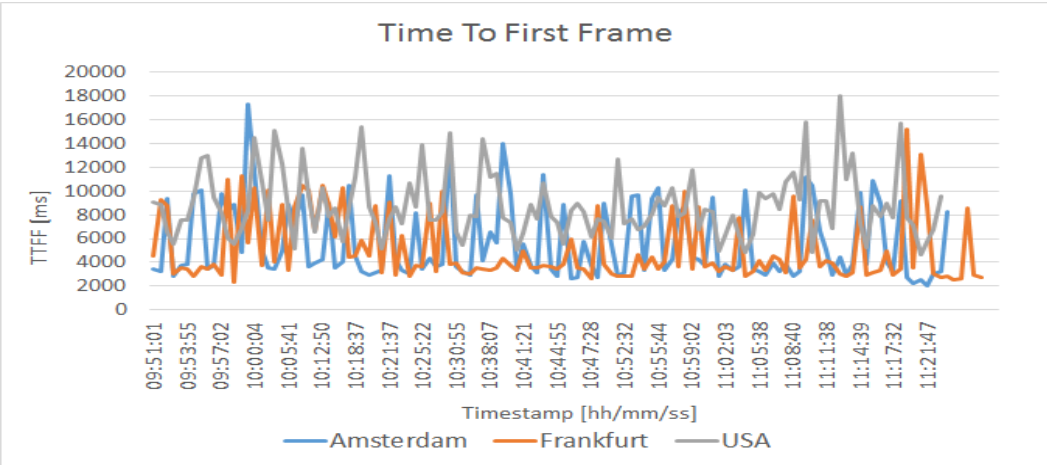

FIGURE 7.7: Time To First Frame

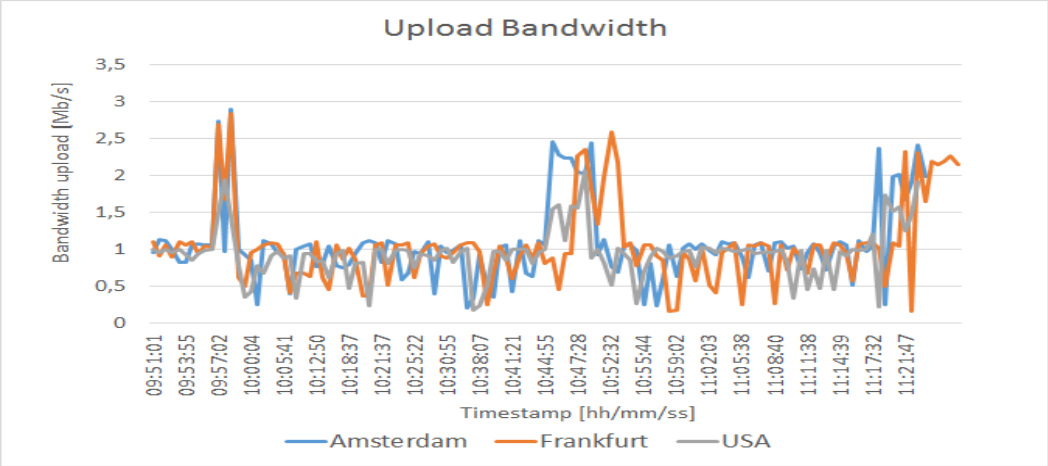

FIGURE 7.8: Upload Bandwidth

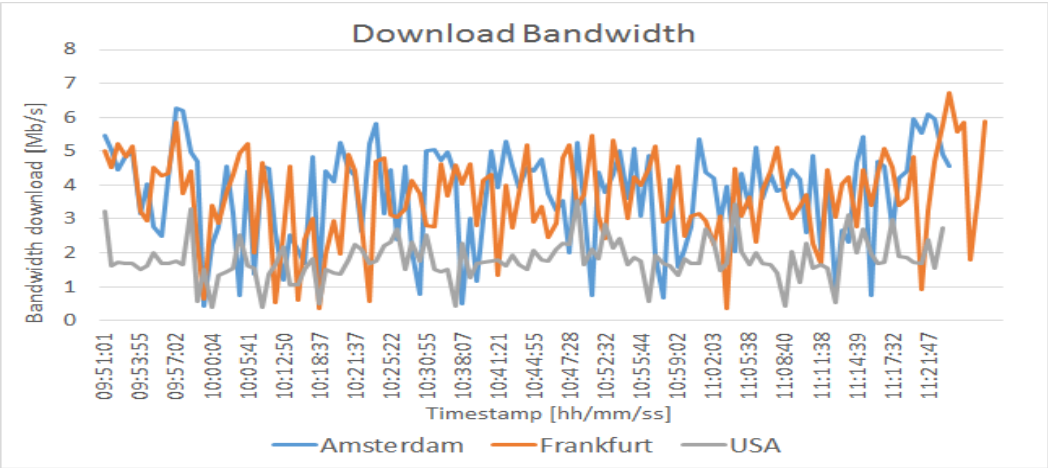

FIGURE 7.9: Download Bandwidth

Host	RTT [ms]	Upload Bw [Mb/s]	Download Bw [Mb/s]	Time to First Frame [ms]
Amsterdam	169,5042	1,0722	3,7262	5650,5391
Frankfurt	154,7339	1,0556	3,6368	5211,6797
USA	261,7899	0,9431	1,8001	8855,7854

TABLE 7.1: Average values

Chapter 8

See-Through Use Case

Edge Computing is an ideal solution for Use Cases like See-Through where very low latency communication and local context are key characteristics. The communication between the participating vehicles and the edge computing application is almost in real-time. Edge application can also benefit from additional informations that are not directly available for the other road users.

8.1 Use Case Description

According to the Use Case when a vehicle has to overtake another vehicle as soon as it signals the overtake the head of the line vehicle sends to him a live stream video taken from its dash cam. In this way the vehicle who wants to start the passing maneuver can see if there is an incoming vehicle in the opposite direction and consequently stop the overtaking.

In Figure 8.1 is represented the scenario along with the main operational states identifying this use case:

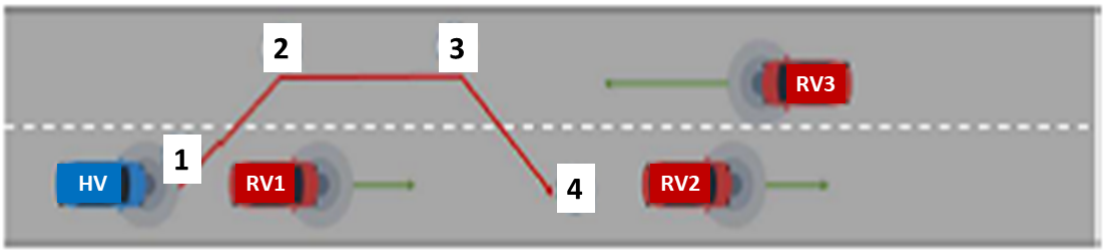

FIGURE 8.1: Overtake Procedure [5]

- State 1 = HV starts receiving streaming video from RV1
- State 2 = HV has fully moved into the passing lane, continues receiving video streaming from RV1
- State 3 = HV has reached the position in the passing lane when it is ready to start the maneuver to return to the starting lane
- State 4 = HV completes the passing maneuver and can stop receiving the streaming video from RV1

In summary, here the aim is to provide HV driver a clear, reliable and real-time view of the road situation in front of the vehicle it is trying to pass and help avoid a possible collision. The HV driver, using the additional video information provided by RV can see whether there are incoming vehicles in the opposite direction to understand if is safe to go on in the passing maneuver. The computational capability of the MEC server must be much higher then any embedded processor in the vehicle, since the application must be able to evaluate several other parameters useful to conclude in a safe way the passing maneuver. Beyond the real time video streaming the application should provide in an analytic way the information regarding the distance path required, the trajectory of the oncoming RV3, the estimated gap between RV1 and RV2 at the end of the maneuver and the risk of a crash between HV and RV3 if their paths overlap. For all these reasons and mostly to provide a reliable video stream a MEC server is strictly required [5].

8.2 Architecture

In the Figure 8.2 is possible to see a realistic scenario in which is shown the high layer architecture of the system.

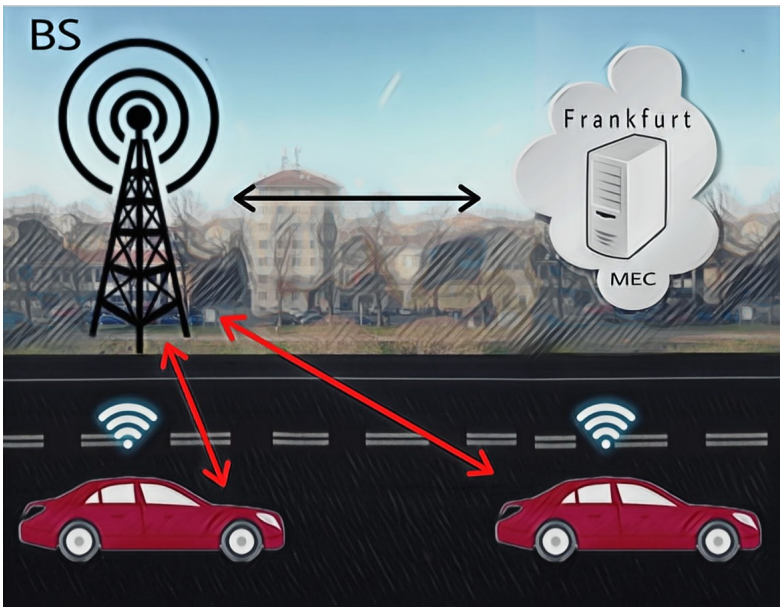

FIGURE 8.2: See-Through high layer Architecture [10]

The vehicles on the road are connected, using the mobile network, to a base station and consequently to a MEC server. In the ideal solution this MEC server should be at the edge of the network, but through the IVE experiment we know that our backup solution placed in Frankfurt gives acceptable and predictable results in terms of latency. In the following paragraphs will be possible to see more in detail each component.

8.3 Components

Since we are referring to the OpenNESS architecture the applications running on the MEC server must be containerized to allow a future deploy on the edge node platform.

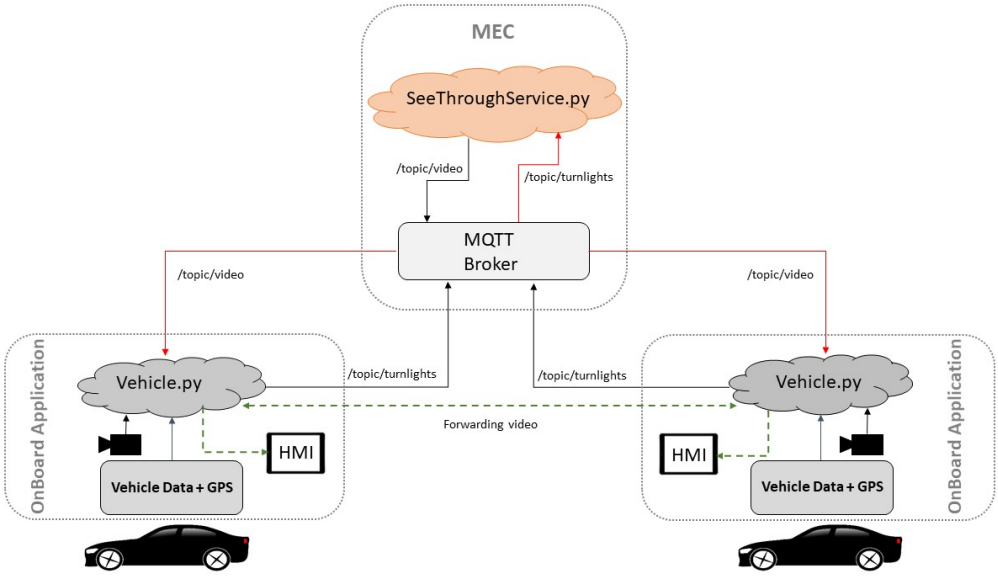

FIGURE 8.3: See-through Components

8.3.1 OnBoard Application

The code has been written in Python and run in the vehicles involved in the simulation. It is composed by two threads, as shown in Figure 8.4, that perform two different tasks:

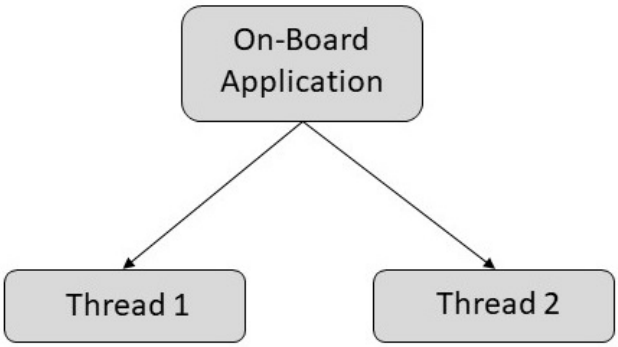

FIGURE 8.4: On-board Application

- **Thread 1** It connects to the mosquitto broker using the Paho-Mqtt library. Receives data from the CAN network and the GPS of the vehicle, then converts these data into json strings using json.dumps() function and publishes the json CAM packets every 100 ms at /topic/turnlights. In the local simulation the

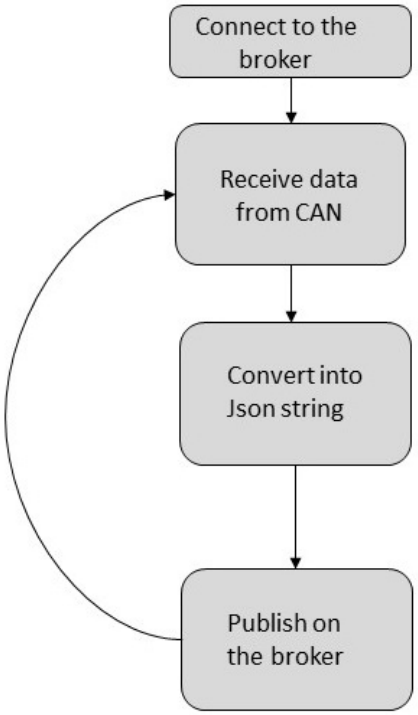

FIGURE 8.5: Thread 1 Flow Chart

packets are taken from the log file of the IVE experiment. Below is possible to see an example of the json datagram.


```
{"eventName" : "CAM",
 "Timestamp" : "1569227751102",
 "MsgCount" : "18679",
 "utcCreationTimestamp" : "20190923-083551102",
 "payload" : {
    "decodedCAM" : {
            "stationID" : 140,
            "ipAddress" : "192.168.1.140",
            "stationType" : 5,
            "vehicleRole" : 0,
            "date" : 20190923,
            "time" : 83551102,
            "lat" : 45.105403,
            "lon" : 7.645323,
            "speed" : 0,
            "heading" : 0,
            "brakePedalStatus" : 0,
            "turnLights" : 0,
            "longitudinalAcceleration" : 0,
```

```
            "lateralAcceleration" : 0,
            "verticalAcceleration" : 0,
            "int1UDPPort" : 6001,
            "int2UDPPort" : 6002,
            "out1UDPPort" : 7001,
            "out2UDPPort" : 7002,
            "yawRate" : 0,
            "vehicleWidth" : 2,
            "vehicleLength" : 2,
            "GPS_antenna_X_pos" : 1,
            "GPS_antenna_Y_pos" : 2,
            "HDOP" : 0,
            "hydroPlanning" : 0,
            "friction" : 0
            }
    }
}
```

- **Thread 2**

  Using the same library mentioned before, Paho-Mqtt, this thread subscribes

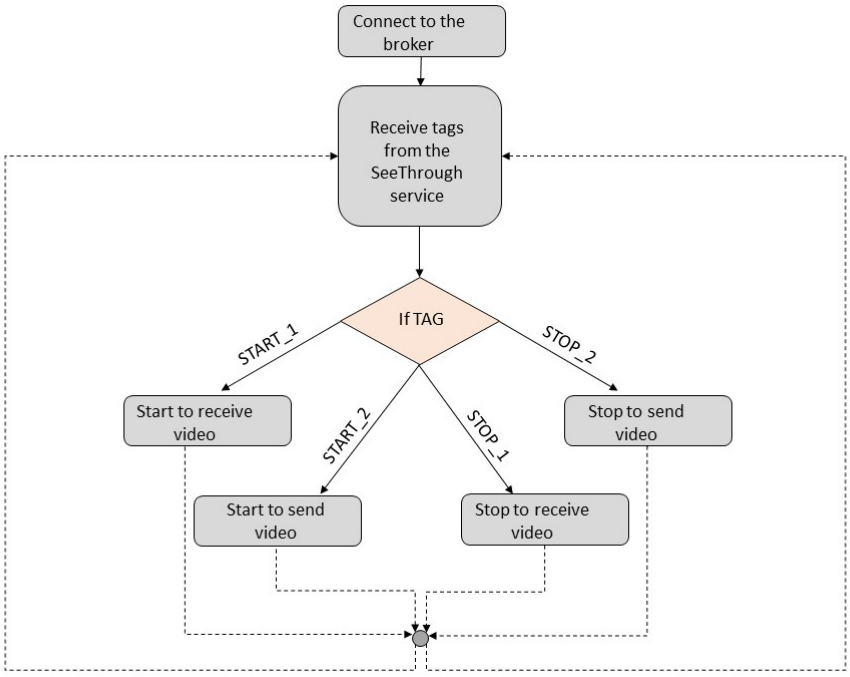

FIGURE 8.6: Thread 2 Flow Chart

to /topic/video and waits until a tag is sent by the see through service. It can discriminate between four types of tags and behave as consequence:

- START_1
  open the pipeline to receive a video stream
- START_2
  begin to send the video ata known IP address, the one of the vehicle behind
- STOP_1
  kill the receive process and wait for the new tag
- STOP_2
  kill the sending process and wait for the new tag

The video streaming has been implemented in a separate scripts using gstreamer.

Gstreamer

GStreamer is a framework for creating streaming media applications. With Gstreamer's framework is possible to develop any type of streaming multimedia application since it is able to manage quite easily audio, video or both. Gstreamer, moreover, can process any type of data flow, thus it is used not only for handle multimedia data stream. The pipeline design is made to have little overhead introduced by the applied filters. This makes GStreamer a good framework for designing even high-end audio applications which put high demands on latency. One of the most obvious uses of GStreamer is using it to build a media player. GStreamer already includes components for building a media player that can support a very wide variety of formats, including MP3, Ogg/Vorbis, MPEG-1/2, AVI, Quicktime, mod, and more. GStreamer, however, is much more than just another media player. Using its pluggable components matched and mixed in arbitrary way is possible to create several types of pipelines that can be utilized to build different kind of applications. The framework is based on plugins that will provide the various codec and other functionality. The plugins can be linked and arranged in a pipeline. This pipeline defines the flow of the data. The GStreamer core function is to provide a framework for plugins, data flow and media type handling/negotiation. It also provides an API to write applications using the various plugins.
In the vehicle applications have been used the two following strings to build the media player. Of course, as I said before, the components and the plugins can be arranged in several ways, so there are many other possible solution at this problem [13].

- **Sender**
  exec gst-launch-1.0 -m filesrc location=sampleVideo.mp4 ! decodebin ! x264enc ! rtph264pay ! udpsink host=127.0.0.1 port=5000
  In this case, since we are doing a local simulation, we send a video in the filesystem and the destination host is the local one (127.0.0.1). In a real scenario the video should be taken from the dash cam location while the destination

host can be given as parameter or can be send in broadcast and received only from the waiting user.

- **Receiver**
  exec gst-launch-1.0 -v udpsrc port=5000 caps="application/x-rtp, media=video, clock-rate=90000, encoding-name=H264, payload=96, ssrc=3394826012, timestamp-offset=2215812541, seqnum-offset=46353" ! rtph264depay ! decodebin ! videoconvert ! autovideosink sync=false

8.3.2 MEC Applications

MQTT Broker

An MQTT Broker is a software that can run either on premise or in the cloud used to exchange informations. It is usually compared to a post office in which the users don't use the address but any one who want to send or receive a message use a specific subject line called "Topic". The communication through the broker is many-to-one in the sense that many users can publish their informations on the same topic and also one-to-many, because many users can be subscribed to the same topic and get the same informations at the same time. Moreover each client can both publish and receive messages at the same time (MQTT is a bidirectional protocol). MQTT enables also a Transport Layer Security (TLS) encryption with username and password, is mandatory to have these informations in order to be subscribed or to publish on a topic. For this application we used the Mosquitto MQTT Broker, an open source Eclipse project that can be easily downloaded and configured. Than, having it running in background, through the paho.mqtt library we enable the communication between the MEC server and the road users.

MQTT Protocol

MQTT stands for MQ Telemetry Transport. It is a protocol that relies on TCP protocol for data transmission. we have also other variants like MQTT-SN used over other transport protocol like UDP or Bluetooth. It is a lightweight messaging protocol based on the publish/subscribe paradigm. Its principles match with the requirements of the Internet of Things and the Machine to Machine communications, thus nowadays is very useful for the actual applications.
It defines two entities:

- **MQTT Broker**
  As explained before the message broker is a software that receives messages and route them to the proper destination

- **MQTT Client**
  It is any device that by running an MQTT library is connected to an MQTT Broker over the network

Publisher and subscriber are completely unaware one of the other, publishers do not need to know the number or the location of the subscribers and at the same time the latter do not need any information about publishers. An MQTT control message can range from 2 up to 256 megabytes if needed. Moreover there are fourteen defined messages used to connect, disconnect, publish, subscribe or acknowledge the broker. MQTT does not have an encryption security procedure but it can be implemented using the underling TCP protocol.

SeeThrough Service

As we can see from the flow chart in Figure 8.7, this Python application, which is the one that contains the see-through concepts, first of all subscribes to the topic: /topic/turnlights and receives the json packets from the vehicles. Using the ipAddress field it is able to discriminate which car has sent the information.

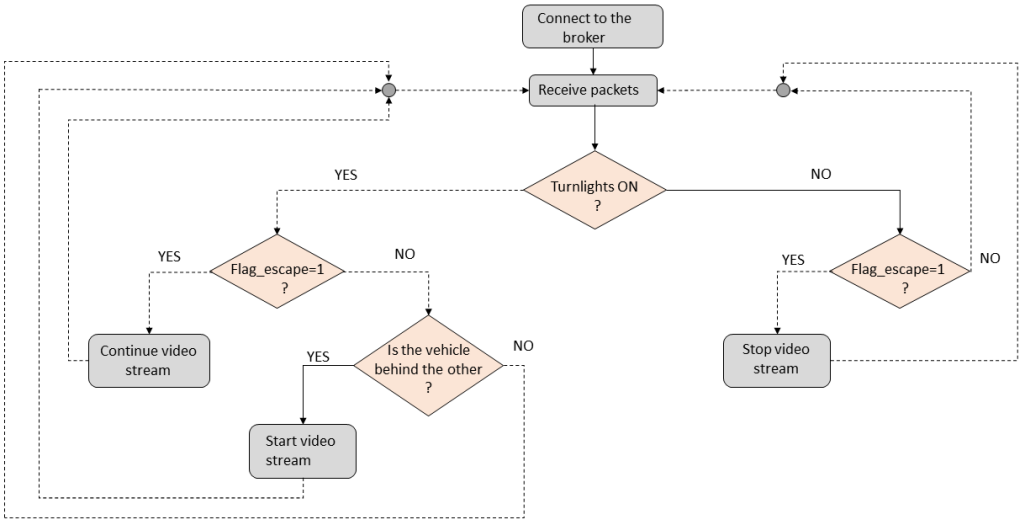

FIGURE 8.7: See-Through Service Flow Chart

The application stores the most recent useful informations retrieved from the packet i.e. latitude, longitude and heading. Then checks for the turn lights status. When the left turn light is inserted in the dedicated field of the json packet is written "turnLights=2" and, moreover, knowing that the vehicles send a packet every 100 ms and that the frequency of the turn light signal is 1 Hz, is possible to establish what is the exact pattern to find if the left turn light is on. In order to do this it has been used a three cells vector; as soon as the application finds the value 2 in the packet it starts summing these values in the first cell and after ten packets moves to the second cell, does the same for the following ten packets and repeat the procedure for the third cell. If after this procedure in the dedicated vehicle vector is written "20 0 20" it means that the left turn light is on. At this point the geo-mapping algorithm starts, using the lasts saved informations regarding longitude, latitude and heading of the two vehicles, the application, through a linear transformation, discriminates if the vehicle that has inserted the turn light is the head of the line or not. If it is the

application clears the vector and start again the research. If, instead, the vehicle is behind the other the See-Through service sends the specific tags on the /topic/video to let the forwarding video start and raises a flag (Flag_escape) meaning that from now on it has no more to look for the turn light "ON pattern" but for the "OFF pattern" that is every pattern different from "20 0 20" or "0 20 0". As soon as the signal is detected Off the application publishes the proper tags to trigger the end of the forwarding video.Then drops the relative Flag_escape, clears the vector and start again the searching of the "ON pattern".

Chapter 9

Conclusions

As is clear from the previous chapter the See-Through service implemented is still an initial version, not ready to be tested in an urban environment. Anyway during the implementation all the important topics regarding the development of this application target to be deployed on a MEC server came out. In particular the algorithm should match the with low response latency provided by the edge computing, thus, when possible is better to avoid long cycles and huge data structure. Moreover we faced also the transmission of a multimedia contents which is not trivial even with a valid support like Gstreamer.

9.1 Further Improvements and Future Developments

The application, first of all, should be scaled in an urban environment, where much more than two users are involved in the communication, since the See-Through service create an instance of a *class vehicle* for each user this part must be quite easy, in the sense that each time it receives a message from a new client it can create a new instance, or the classes can be managed referring to a database where each active user in the covering area has an entry.
A crowded environment lead also to a more tricky detection of the head-of-the-line vehicle. In my opinion one solution can be the one related to the image processing and the object recognition to evaluate which is the vehicle who has more vision of the road in the line, of course in a limited space range.
Furthermore we need to implement a way to recover from packet lost or delayed, mostly for what concern the detection of the turn light patterns. At the moment the algorithm is not able to find the ON pattern if even a single frame is lost. Anyway these improvements must be thought with the knowledge of the simulation environment and the network condition. The See Through is an important Multi-Access edge computing Use Case strictly related to the road security that must be completely reliable before being utilized.

Bibliography

[1] Seung-Hoon Hwang Amir Haider. "Adaptive Transmit Power Control Algorithm for Sensing-Based Semi-Persistent Scheduling in C-V2X Mode 4 Communication". In: *electronics* (July 2019).

[2] "C-V2X Edge Computing: The Winning Technologies for Connected Vehicles and Autonomous Driving". In: (Sept. 2018). URL: https://5gaa.org/news/c-v2x-edge-computing-the-winning-technologies-for-connected-vehicles-and-autonomous-driving/.

[3] Altiostar Cisco Rakuten. "Reimagining the End-to-End Mobile Network in the 5G Era". In: *Cisco public* (2019).

[4] Intel Corporation. "OpenNESS Architecture and Solution overview". In: (2019). URL: https://github.com/open-ness/specs/blob/master/doc/architecture.md.

[5] Pekka Kuure Sami Kekki Zheng Zhou Alice Li Christoph Thein Edwin Fischer Ivan Vukovic John Cardillo Valerie Young Soo Jin Tan Vince Park Michaela Vanderveen Stefan Runeson Stefano Sorrentino Dario Sabella Hassnaa Moustafa. "5GAA - Toward fully connected vehicles: Edge computing for advanced automotive communications". In: (Dec. 2017).

[6] "Docker overview". In: (2019). URL: https://docs.docker.com/engine/docker-overview/.

[7] Multi access Edge Computing (MEC) ETSI Industry Specification Group (ISG). "Phase 2: Use Cases and Requirements". In: *ETSI GS MEC 002 «Mobile Edge Computing (MEC)»* (Oct. 2018).

[8] Multi access Edge Computing (MEC) ETSI Industry Specification Group (ISG). "Study on MEC Support for V2X Use Cases". In: *ETSI GR MEC 022 «Mobile Edge Computing (MEC)»* (Sept. 2018).

[9] "Family of Standards for Wireless Access in Vehicular Environments (WAVE)". In: *IEEE 1609* Retrieved 2014-11-14 (Apr. 2013).

[10] "Framework and Reference Architecture". In: *ETSI GS MEC 003 «Mobile Edge Computing (MEC)»* (Mar. 2019).

[11] "Multi-access Edge Computing (MEC)". In: (). URL: https://www.etsi.org/technologies/multi-access-edge-computing.

[12] Konstantinos Alexandris Navid Nikaein Chia-Yu Chang. "Mosaic5G: Agile and Flexible Service Platforms for 5G Research". In: (Aug. 2018).

[13] "Overview". In: (2017). URL: https://gstreamer.freedesktop.org/documentation/additional/design/overview.html?gi-language=c.

[14] "SAGUNA VEDGE". In: (2019). URL: https://www.saguna.net/saguna-vedge/.

[15] Yonggang Fang Pekka Kuure Alice Li Debashish Purkayastha Feng Jiangping Danny Frydman Gianluca Verin Kuo-Wei Wen Kwihoon Kim Rohit Arora Andy Odgers Luis M. Contreras Salvatore Scarpina Sami Kekki Walter Featherstone. "MEC in 5G networks". In: *ETSI White Paper* 28 (2018).

[16] "Technical Requirements". In: *ETSI GS MEC 003 «Mobile Edge Computing (MEC)»* (Mar. 2016).

[17] Vodafone. "First MEC Hackathon – Exemplary case study". In: (2018). URL: https://forge.etsi.org/mec/mec_case_study.PDF.

[18] "Why Edge Computing is key for the automotive industry". In: (Apr. 2019). URL: https://www.teraki.com/blog/why-edge-computing-is-key-for-the-automotive-industry/.

[19] Jun Zhang Kaibin Huang Yuyi Mao Changsheng You and Khaled B. Letaief. "Mobile Edge Computing: Survey and Research Outlook". In: *IEEE Communications Surveys Tutorials* (Jan. 2017).