POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

DESIGN AND ASSESSMENT OF A CLOUD-BASED REAL-TIME GEO-REFERENCED VEHICULAR NOTIFICATION SYSTEM

Supervisors

Prof. Claudio Ettore CASETTI

Dott. Marco RAPELLI

Candidate

Andrea GIORGI

Academic Year 2020-2021

Abstract

The recent need of a fully autonomous car has pushed researchers into developing applications, services and technologies capable of connecting the vehicle to its surroundings. A real-time geo-referenced notification system coupled with aggregation of data and optimization of fog-computations in the edge network allows for a fast and reliable traffic orchestration in critical areas. Modern literature showcases the efficiency and responsiveness of cellular vehicle-to-network in areas like collision avoidance and traffic control. This thesis aims at developing an architecture capable of aggregating and disseminating real-time notification messages through a multi-MAC infrastructure. Furthermore, an assessment on its efficiency and responsiveness is carried out through several metrics. After a quick overview on the state of the art, the overall architecture is discussed. This features a brief overview on the MEC infrastructure that recognizes and elaborates information on the MEC node. The focus of the analysis revolves around the cloud architecture algorithms and the message flows traveling from the road to the users. Due to stringent requirements, experimental data are collected on the end-to-end latency; this serves as a benchmark for evaluating the reliability and effectiveness of such a solution. The collected data are then aggregated in a live Citizen Application that displays information about vehicles and alerts on the critical zones where possible hazards have been detected.

Acknowledgements

A Mamma, Papà e Stefano,
a cui devo tutto.
A tutte quelle persone
che in un modo o nell'altro
hanno fatto parte di questo cammino.

First and foremost, I would like to thank Prof. Casetti for giving me this incredible opportunity and for providing me with thorough assistance throughout the entire work.

A special thanks goes to Dr. Rapelli, for guiding me through project with patience and diligence; his insights have been essential.

To everyone who has been a part of this directly or indirectly, each one of you has enriched me personally and professionally. To all of you goes a sincere thank you.

Table of Contents

List of Tables

List of Figures

Acronyms

ITS
Intelligent Transport Systems

DSRC
Dedicated Short Range Communication

RSU
Road Side Unit

V2X
Vehicle-to-everything

V2V
Vehicle-to-vehicle

V2N
Vehicle-to-network

C-V2X
Cellular Vehicle-to-everything

LiDAR
Light detection and ranging

GNSS
Global Navigation Satellite System

CSMA/CA
Carrier Sense Multiple Access with Collision Avoidance

SAE

Society of Automobile Engineers

CAN

Controller Area Network

Chapter 1

Introduction

In a world where the digital transformation is becoming a fundamental player in every technological sector, the vehicle is establishing itself as the prime drive for this change. The need of an interconnected mobility infrastructure is becoming crucially important in the development of a fully-autonomous mean of transportation. In the past years many important institutions have laid significant foundations towards achieving this goal. The *Intelligent Transport Systems* (**ITS**) refers to the set of information which have the aim of improving the transportation operations in order to increase **safety**, **reduce cost**, **save energy** and **time**.

ITS does not only introduce new information that were before unseen to the user but it plays the role of strengthening the ones that were already present. This is done in an effort to improve the existing network with accurate and complete traffic information, as well as new means of controlling it. The vehicle must be able to understand its existing infrastructure and must be able to improve its quality for all the other players in the network.

The correct functioning of ITS is possible thanks to the Vehicles/Users that provide constant exchanges of data. These, are then collected and managed by the road infrastructure which provides traffic control and services. Lastly, the communication networks (VANETs) work as a bridge between the two by allowing fast and reliable communication. To guarantee high mobility, network partition and fragmentation the information must be exchanged efficiently to avoid broadcasting issues (broadcast storm). The main characteristics of VANETs are the presence of fixed RSUs (Road Side Units) and vehicles which can move at low and high speeds, this poses a challenge due to the variable connection window. Nodes in the VANETs environment are restricted to follow the topology of the road (urban, rural and highway) leading to various challenges. Urban mobility is a complex network due to the sheer number of vehicles and obstacles and the presence of multiple users.

The increased number of VANETs application fueled the development of Dedicated Short Range Communication (**DSRC**) which exploit V2V and V2I exchanges of messages. The IEEE 802.11 covers several wireless standards. Among these, IEEE 802.11p WAVE covers the standardization of DSRC which uses the 5.85-5.925 GHz frequency range dedicated for vehicular communication. It relies on the communication transceivers mounted on the RSU and on board of the vehicles. However, performances are greatly affected by parameters such as vehicle **speeds** and **number of vehicles** that can be on a single channel. Latency and delivery success rate seem to be the most affected parameters when the number of vehicles increases over the channel. Whilst the DSRC protocol has been developed in the United States, the ITS-G5 one was developed by the European Telecommunication Standard Institute (ETSI) but serves a similar purpose and develops the same kind of problems seen in DSRC.

To improve the performance indicators of C-V2X, a newer solution has recently emerged exploiting 3GPP Long Term Evolution (LTE) Release 14 "PC5" (Device to Device). This allows to respect bounded low latency and a fair amount of volume of traffic with relative high speed[1]. In addition to this, the robustness is increased and fewer RSU are needed for coverage. While it may seem that C-V2X greatly differs from DSRC, it still exploit the same PHY/MAC layer. Although the most common mode of utilization of C-V2X generally fall into V2V, V2I and V2P it is also possible to implement a V2N (Vehicle to Network) communication mode that operates on the "Uu" interface that operates in the traditional mobile broadband licensed spectrum [2]. The use-case that will be dealt on this thesis greatly exploits these modes of communication, especially C-V2N.

Chapter 2

C-V2X

The benefits of the C-V2X are various and important. Typical fully-autonomous driving vehicles are equipped with a variety of sensors: Camera, LiDAR, Radar, GNSS, and CAN that are very well suited to analyze in depth the zones nearest to the vehicles. The advent of C-V2X makes possible the detection of hazards from a further distance. *Non in-line of sight objects* (NLOS) are impossible to be detected with the sole use of the on-board instrumentation. This makes this technology a compelling element in the improvement of road safety and comfort.

C-V2X can aggregate information that are being collected and can be periodically update onto a map. Other vehicles can then have access to these information from distances that would have been impossible before. Features like blind-spot detection, platooning and long-range perception can be achieved thus improving road capacity and efficiency. C-V2X has the novelty that is does not require a sim card or a licensed spectrum to function, it does not require an infrastructure to function in the case of V2V and V2I and above all it is has a scheduled reservation that does not require CSMA/CA. Compared to the 802.11p, it is far more powerful in its range and is more robust.

2.1 C-V2X stack

All the efforts performed by SAE, ETSI and IEEE in defining applications, messages/facilities, security services and transport/networking layers have all been leveraged by the C-V2X. Only the physical and the MAC layer have been replaced from the 3GPP to provide the end to end solution. The physical layer is based on the Single Carrier Frequency Division Multiplexing which supports 10 or 20 MHz channels. Each of these channels is divided into sub-frames, Resources Block (RBs) and subsequently sub-channels [1]. C-V2X defines two sub-channels that are a group of RB pairs in the same sub-frame. The sub-channels are then exploited

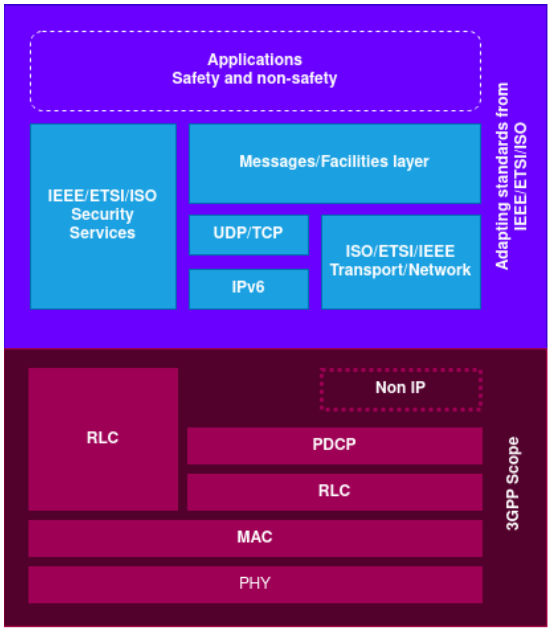

Figure 2.1: C-V2X stack

to transmit data and to control information. The physical channels are divided into Downlink and Uplink Channels. The data information is transmitted over the Physical Downlink Shared Channel (**PDSCH**). The majority of data channels are exploited for control operations. The Physical Broadcast Channel (**PBCH**) is used to transmit **MIB** (Master Information Block) and Radio Access Channels parameters. The Physical Control Format Indicator Channels (**PCFICH**) transmits the details on the format and the Physical Hybrid ARQ Indicator Channel (**PHICH**) transmits the ACK/NACK to the uplink frame. Finally the Physical Downlink Control Channels (**PDCCH**) controls the downlink resource scheduling, the uplink power control instructions and the uplink resource grant as well as the indication for paging/system information. As described before due to the similarity that LTE and C-V2X share, the physical layer is identical for both. The spectrum at which C-V2X operates varies depending on the region in which it is employed. Particularly in Europe, different channels are used to allow C-V2X to co-exist with the 802.11p standard. Differently, in North America the FCC has approved the exclusive use of the upper 5.9 GHz band for C-V2X applications.

The introduction of the Device-to-Device (**D2D**) in the LTE release 12 (commonly called **ProSe**) saw the addition of a UL/DL transfer mode called Sidelink transfer. Such a feature is also present in the V2X with the addition of enhanced ProSe interfaces (PC5) that enable Vehicle-to-Vehicle communication. This was specifically designed to tackle high speed and high density scenarios.

LTE-V2X uses centralized scheduling mode (Mode 3) and decentralized scheduling mode (Mode 4) that supports PC5. Mode 3 has the task of allocating cellular network resources, on the other hand Mode 4 does not require cellular coverage; this makes it possible for vehicles to autonomously select their radio resources using Semi-Persistent Scheduling (SPS) In these cases the Road Side Unit (RSU) and the On Board Unit (OBU) are sufficient to deploy C-V2X V2V, V2I, V2P [3].

2.2 C-V2X Architecture/ProSe

The ProSe architecture enables services that are provided by the 3GPP based on the UEs being in proximity one with the other. The 3GPP system enablers for ProSe include:

- EPC-level ProSe Discovery;
- Direct discovery;
- Direct Communication;
- UE-to-Network Relay;

The non-roaming architecture will be considered as it is the most common one. The architecture can be divided into the LTE Access Network and the Evolved

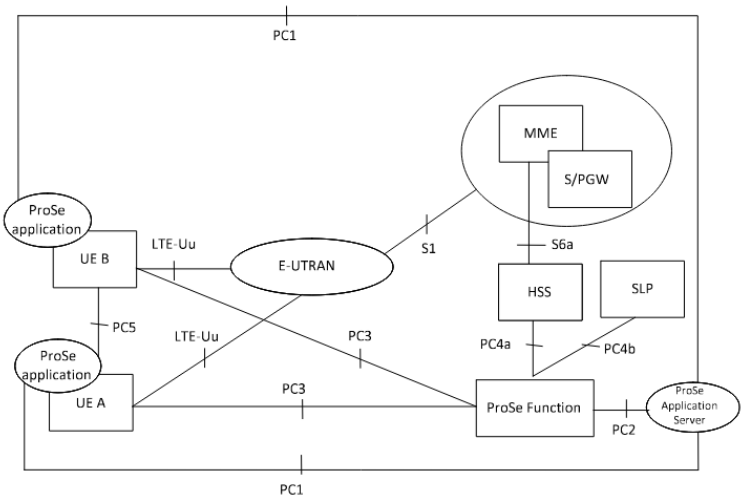

Figure 2.2: Non Roaming ProSe Architecture. Courtesy of ETSI

Packet Network (**EPN**) that describes the core network. It is possible to notice the various interfaces present in the architecture, namely:

- **PC1** The reference point between the ProSe application in the UE and in the ProSe Application Server.
- **PC2** The reference point between the ProSe Application Server and the ProSe Function. It is used to define the interaction between ProSe Application Server and ProSe functionality provided by the 3GPP EPS via ProSe Function for ProSe Direct Discovery and EPC-level ProSe discovery.
- **PC3** Which is the reference point between the HSS (hosting the client profile) and the ProSe function. It provides the subscription information to authorize access for ProSe Direct Discovery and ProSe Direct Communication on a per PLMN basis.
- **PC4b** Used by the ProSe Function in EPC-level ProSe Discovery Function.
- **PC5** It is the reference point between ProSe enabled-UEs for the control and user plane for ProSe functions.
- **S6a** In the case of ProSe S6a is used to download ProSe related subscription information to MME during E-UTRAN attach procedures or to inform MME subscription information in the case in which the HSS has changed.

The ProSe function is a logical function used for network-related actions that are required by ProSe such as service authorization and PLMN-specific information. Each PLMN has only one PSF available. The ProSe Function plays a different role depending on the features of ProSe. There are three subfunctions that are employed to provide UEs with data necessary for ProSe operations. The **Direct Provisioning Function** (DPF) is employed to allow UEs with the necessary parameters to execute PDD and PDC. The **Direct Discovery Name Management Function** used by PDD to allocate and process ProSe Application IDs; its main function is to protect the discovery messages. Finally, the **EPC-level discovery** includes a variety of functions among which storage of ProSe-related subscriber data and/or retrival of subscriber data from HSS; it also provides location services client, handle of EPC ProSe User IDs, Application Layer User IDs and much more [4].

2.2.1 UEs

The UEs, as evidenced in figure 2.2, must be able to exchange information through the PC3 interface and allow for open and restricted ProSe Direct Discovery of the other UEs across the PC5 interface. It may also support additional functions that are useful in vehicular applications such as:

- Procedures for one-to-many Direct Communication over PC5
- Procedures for one-to-one Direct Communication over PC5.

2.2.2 ProSe Application Server

The Applicaiton Server must be able to provide:

- Storage of EPC user IDs
- Mapping of Application Layer User IDs and EPC ProSe User IDs
- Interaction with UEs through PC1 interface.

2.2.3 Mobility Management Entity

The MME resides in the Core Network and is responsible for retrieving the information related to ProSe from the HSS and to inform the E-UTRAN that the UE is authorized to use ProSe.

2.2.4 P/S-GW

The P/S-GW are responsible for managing the mobile traffic and take care in receiving information related to the ProSe UE-Network Relay coming from the SGW and MME respectively.

2.3 ProSe High Level Functions

High Level Functions in ProSe are a set of functions that allow for Discovery and Communication between UEs [4]. Firstly the devices must be authorized to perform these tasks through the PC3 interface where the UE gets authorized on a per PLMN basis. In host PLMNs, the authorization for Direct Communication is requested from the Home PLMN (HPLMN). Direct Discovery is defined as the process capable of identifying and detecting another UE in proximity. The Discovery can either be open or restricted; the first allowing for no specific permission needed by the UE to be discovered whereas the second one requires a specific permission. There exist two models of Discovery:

- **Mode A** ("I am here"): where the UE announces certain information that can be used by proximity UEs and monitors information of interest in the proximity of announcing UEs.
- **Mode B** ("Who's there?"): UE transmits a request containing information on what is interested to discover and the reply is handled by another UE which is capable of providing the reply.

2.3.1 ProSe Direct Communication

Direct Communication allows a One-to-many on ProSe enabled Public Safety UEs that have been previously authorized and can be applied both in reach of an E-UTRAN and both outside E-UTRA coverage. The characteristics of the one-to-many communication is that it is connectionless (no signaling over the PC5 interface). Members of a group of devices share a secret from which a group of security keys can be derived. As shown in figure 2.3, the UE is configured with

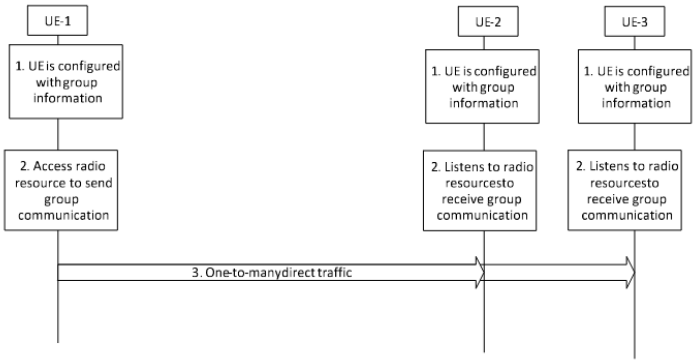

Figure 2.3: One-to-many ProSe Direct Communication transmission. Courtesy of ETSI

information for a one-to-many communication. The UE is then tasked with finding a proper radio frequency in which to conduct the transmission. The other devices are then equipped for listening and receive the group information. This is done through the ProSe Layer-2 Group ID, Group IP multicast address. The information are then filtered and if it matches the configured Group ID it is passed to the upper layers [4].

Chapter 3

Basic Set of Applications

ITS comprises a variety of use cases that are distributed over a multitude of ITS stations (ITS-S) that provide co-operating customer services. The most important services provided include *Cooperative Awareness* (**CA**) and *Road Hazard Warnings* (**RHW**). CA works by exchanging messages in the ITS network through ITS-S with the aim of creating awareness between vehicles and road users as well as supporting cooperative performance in the road network [5]. RHW applications have the objective of improving road safety and traffic efficiency through the V2V and V2I network. It exploits a Decentralized Environmental Notification application service that supports RHW applications [6].

3.1 Cooperative Awareness Services

As defined by ETSI, cooperative awareness services provides sending and reception of CAMs. Frequency of sending may vary and is dependent on the CA basic service originating ITS-S [5]. CA is a facilities layer entity in the ITS-S architecture, it interfaces with the application layer through the FA-SAP interface in order to collect information from CAMs. Data collection entities may be the Vehicle Data Provider (**VDP**), the Position and Time management (**POTI**) and the Local Dynamic Map (**LDM**). CA interfaces with the Network and Transport Layer through the **NF-SAP** to exchange messages with other ITS-S and with the management and security layer through the **MF-SAP** and **SF-SAP** respectively [5].

For sending and receiving CAMs, the following sub-functions must be provided:

- **Encode CAM**
- **Decode CAM**
- **Transmission management** that takes the task of managing the start and

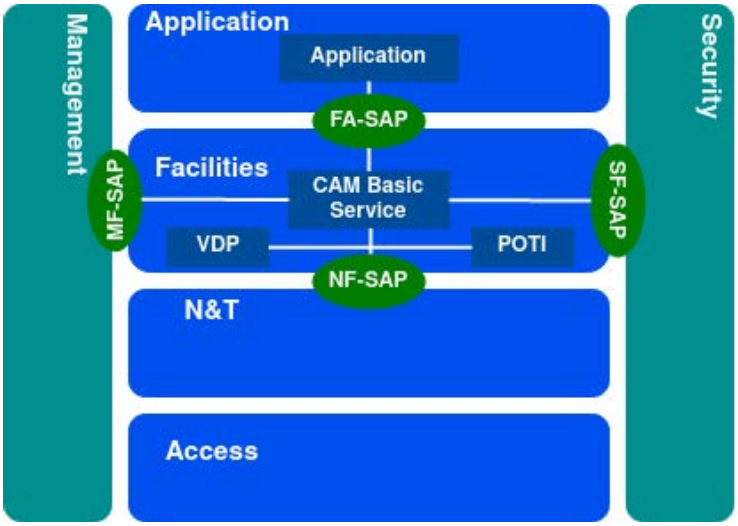

Figure 3.1: CA basic service architecture

end of a transmission and its generation frequency

- **Reception management** that triggers the decoding action

3.1.1 CAM dissemination

To successfully cover an area CA Messages are transmitted from the originating station towards all the receiving ITS-S in a single-hop that are in range. When a vehicle is present on the road, the stream of data is activated concurrently to the ITS-S activation. Similarly the stream of data is interrupted as soon as the ITS-S is deactivated.

The frequency at which messages are generated depends on the channel congestion levels. However, values belonging to the interval of minimum generation time $T_{GenCamMin} = 100ms$ and maximum $T_{GenCamMax} = 1000ms$ are accepted [5]. In the case of LTE-V2X the access layer manages channel congestion and subsequent generation times. Other conditions that may trigger a cam message can occur if:

- If the difference of the current heading and the one previously to the originating ITS-S varies by 4°
- The distance between the previously transmitted and originating station exceeds **4m**
- The speed difference from the originating ITS-S exceeds **0.5** m/s

3.1.2 CAM Format Structure

A CAM is composed of and ITS PDU header that provides information on the protocol version, message type and ITS-S ID from where the message has originated. The CAM can be composed by three types of containers [5]:

- **Basic container** that contains the basic information about the originating ITS-S
- **High frequency container** containing highly dynamic information about the ITS-S
- **Low frequency** containing static or slowly changing information
- **Special vehicle container** that contains info specific to a particular vehicle (i.e Public Transport, Special Transport)

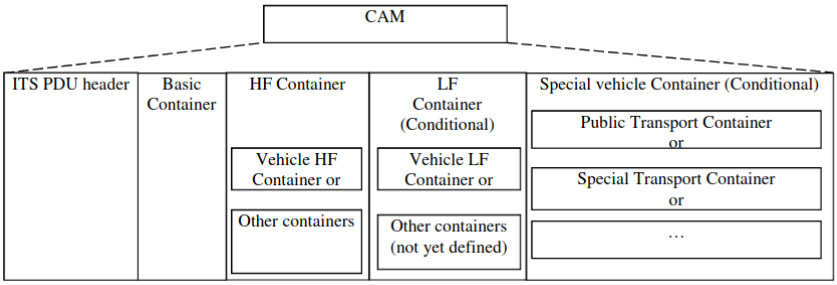

Figure 3.2: CAM structure. Courtesy of ETSI

3.2 Decentralized Environmental Notification Basic Service

Differently from the CA Services, DEN Services are used by ITS applications in order to alert users regarding a specific event detected on the road [6]. The DENM protocol deals with the exchange of these information. The way by which ITS uses DEN services, is through the following process:

- When an alert is raised the DENM disseminates itself across ITS-S that are contained in an area called zone of relevance.
- This transmission can be repeated and is persisted until the event has ceased
- Messages can be elaborated or simply forwarded from an ITS-S to another

- It may be possible to show information to the user through and HMI but it is not mandatory

The DENM contains information related to an event that may impact the road safety and traffic conditions. A DENM may be persistent for as long as the event exists. A particular feature of DEN Messages is that they are independent on the originating station, that is, if a vehicle detects an hazard on the road, the affected vehicles in the nearby zone will still be alerted even after the originating station has left the area of interest [6].

DENM types are of four categories:

- **New DENM**: Whenever a new event is detected for the first time. An action ID is assigned and the event provides attributes such as type, detection time and more
- **Update DENM**: This type of service is provided by the originating ITS-S whenever an information has to be updated
- **Cancellation DENM**: Whenever an event is terminated; The event is issued by the origination ITS-S.
- **Negation DENM**: When an event reaches its termination. For example an hazard that is no longer present on the road will be signaled with a negation DENM once its presence has been removed.

3.2.1 DENM ITS Architecture and Service

The DEN Service, just like for the CA service, interfaces with the Applications in order to receive and process DENMs. In the facility layer it interacts with the Local Dynamic Map that gets updated upon reception of messages. It interacts with the Security nd Management entities through the SF-SAP and MF-SAP respectively and with the lower layers through the NF-SAP.

3.2.2 DENM Dissemination

The event identification is enabled by the actionID that is linked to the originating station. This allows the distinction between various stations even in the case the event that has been generated is identical for all ITS-S. The stationID adds value to the actionID especially in the security level.

When the DEM Service receives a Trigger from the Application layer, it is requested to generate a DENM, at the same time an unused actionID shall be created by the DEN Service [6]. Similarly, when an event has received an update it

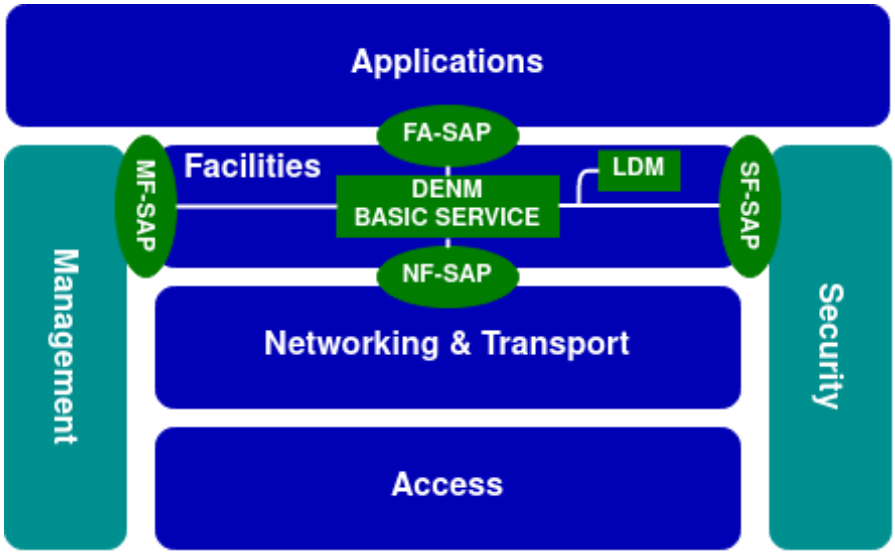

Figure 3.3: DENM architecture

must request to the DEN basic service to a newer *referenceTime* (greater than the originating one). The *actionID* has to remain unchanged as long as the ITS-S remains the same. A DENM repetition can be requested with a *repetitionInterval* and a *repetitionDuration*; in their absence no repetition is processed. The DENM termination may either be a negation or a cancellation. Terminations happen when an ITS-S stops generating DENMs after the *repetitionDuration*, cancellation happens when the *actionID* will be equal to the one set by the application upon triggering.

The **relevance area** is an important characteristics of the DENMs as it provides a region in which vehicles may be exposed to the danger on the road [6]. It must be included by the generating station and must include a *relevanceDistance* at which vehicles shall receive the alert and a *relevanceTrafficDirection* which indicates the direction at which the vehicles may encounter the alert. As an example, if a vehicular hazard happens in a motorway, the vehicles in the opposite lane would not be alerted but in the case of rural areas the alert must be extended to all directions of motion.

Location referencing acts as a complementary function to the relevance area and it provides information on the event position. If the hazard may be encountered form multiple directions the message must include all possible paths of encounter. A receiving ITS-S compares its route to the one provided by the DENM and takes relevant actions.

3.2.3 DENM Container

Similarly to CAMs, DENMs contain an ITS-PDU header that serves identical functions. The management container hosts information such as the *actionID, detectionTime, referenceTime, relevantDistance* and much more. The situation container holds information regarding the content of the event. *InformationQuality* is a range from 0 to 7 that evaluates the quality of the information provided (0 being unavailable info). *EventType* includes the *causeCode* and *subCauseCode* provide high and low level information on the causes of the event. The *linkedCause* and *eventHistory* describe the possible subsequent causes generated by the event and the previous position of it, respectively. This allows to describe a forward and backward map of all causes and reactions due to an event.

The location container includes information such as:

- Event speed;
- Event position heading;
- Traces;
- Road type;

The **à la carte** container is optional and includes additional information that have not been included in the other containers. As an example *externalTemperature* may be added to invigorate the information of adverse weather [6].

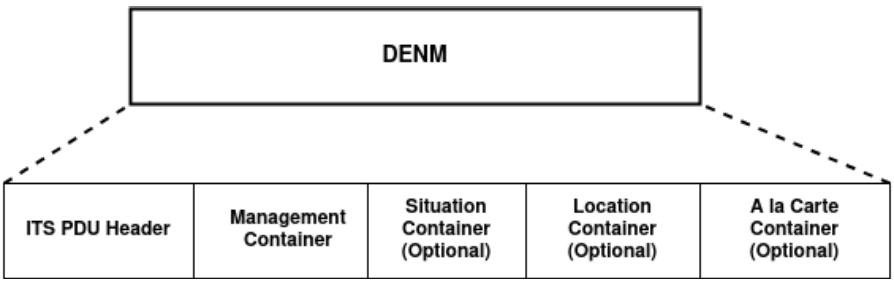

Figure 3.4: DENM container structure

Chapter 4

Multi Access Edge Computing

With the advent of Internet of Things the number of intelligent devices has increased rapidly. Having the possibility of relying on a cloud Computing interface enables rich a powerful solutions for the devices that may want to access it. In addition, the possibility of performing such operations at the network edge enables low latency and low congestion traffic. The MEC as defined by ETSI is divided into three reference points [7]:

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

A MEC infrastructure is composed of a host and of a MEC management that is crucial to run applications within an operator network. The MEC host can be further subdivided into a MEC platform which is a collection of fundamental functionalities that are required to run the MEC applications; such applications are instantiated on the Virtualization Platform [7].

The device applications interact with the MEC system and thus have a life-cycle management.

4.0.1 MEC system level management

At the base of the MEC system level management resides the the **multi-access edge orchestrator**. Its main functionality consists in keeping an overall view on the MEC Host, available resources, available services and topology. In addition

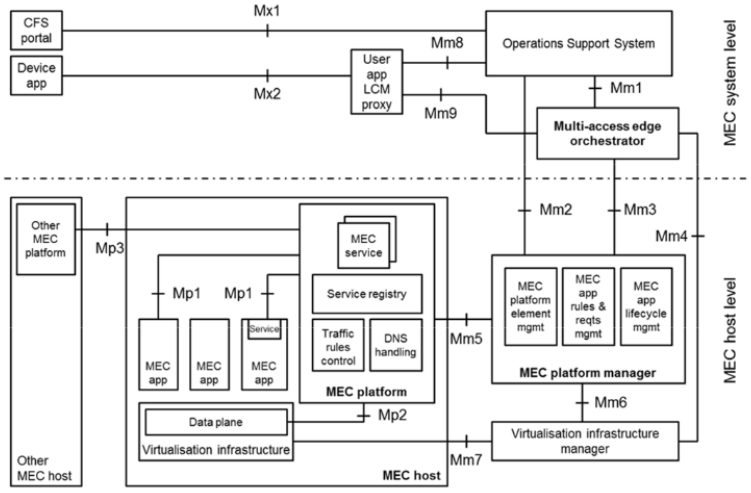

Figure 4.1: Multi-access edge system architecture [7]

to this it is responsible for selecting the correct MEC Host/s based on the needed application constraint such as latency, available resources and services [7].

4.0.2 MEC Host level management

In the Host management level the main task reside in the platform manager that is responsible for keeping track of the life cycle of applications as well as informing the MEC orchestrator about application related events. This platform also receives fault reports and performance measures.

The **Virtualization infrastructure manager** is responsible for allocating and realizing the compute, storage and networking resources for the virtualization infrastructure [7]. It is also tasked with running the image and reporting any possible issues.

4.1 MEC Services

The services provided and consumed by the MEC platform or MEC application takes the name of MEC service.

If an application is registered for that service it interacts through the Mp1 as shown in figure 4.1. A set of services is required in order to fulfill requirements as defined by ETSI. **Radio Network Information** enables up-to-date radio network information, measurements and statistics information related to the user plane as well as information related to the UEs that are being served by the radio node

associated to the MEC host [7].

Location services about UEs can be exploited by other UEs that are associated to the same MEC node thus providing a geolocalization platform that can achieve interesting ITS features with UEs that would otherwise be outside of range in normal DSRC applications. With the help of such a service it is possible to enable **Traffic Management Services** such as BandWidth Management (BWM) and Multi-Access traffic steering that seamlessly redirects application data traffic across multiple access network connections [7].

4.2 MEC Support for V2X Communication

The MEC finds most of its purposes in V2X vehicular communication scenarios. Safety, convenience, vulnerable road users and advanced driving assistance are some of the many applications in which MEC can be employed. The latter collects the most challenging aspects due to the amount of huge data that must fulfill reliable and fast constraints. In addition to this the vehicle can benefit from predictive reliability such as network availability to plan ahead.

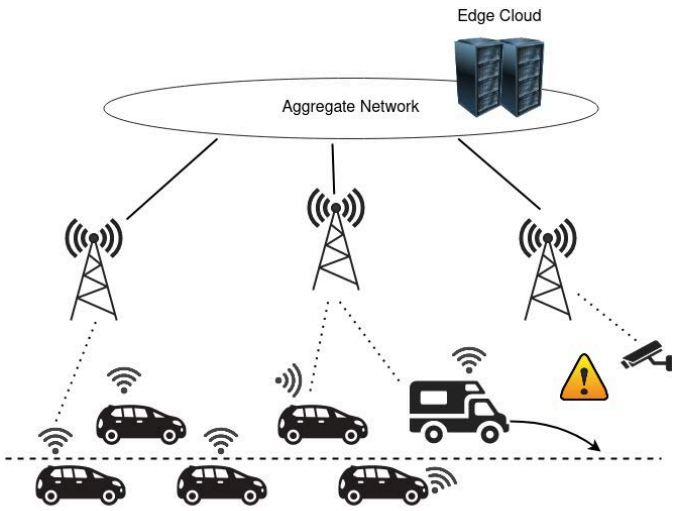

Figure 4.2: Multi-access edge Driver Assistance

Real-time situation awareness becomes of crucial importance in autonomous driving especially in segments of road that may be critical due to traffic or road layout. Vehicles can detect ahead hazards and information that populate a High Definition Local Map in real-time [8]. Local aggregation can be supported by connected nodes with very low latency. In addition, predictive quality must remain

stable in changing environments that may undermine latency and signal-strength parameters [8].

An interesting application described by *M. Malinverno, G. Avino et al.* [9] employs the use of a MEC server to perform collision avoidance algorithms. The architecture is composed of three entities: Users, POAs (Points of Access) and Collision Detection Servers. The Users can be either vehicles or vulnerable users equipped with OBUs capable of periodical broadcast of BSMs (Basic Safety Messages). These information include:

- position, speed and heading;
- lateral and vertical acceleration;
- vehicle length and width

BSMs are sent in broadcast so to allow each user to perform their own collision detection algorithms. In order to fullfill the MEC paradigm, the POAs are not solemnly constraint to collision avoidance, but can be employed for entertainment and leisure.

In the proposed architecture, collision detectors can be physical servers, virtual machines or containers that can run collision detection algorithms. The input provided to the server are BSMs generated by vehicles or users, which is then processed to understand whether a possible collision may happen and emit an alert.

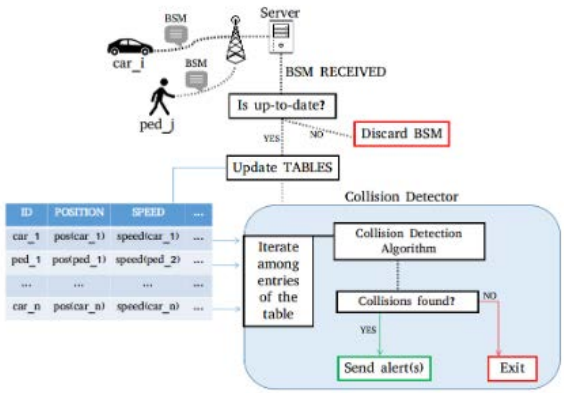

Figure 4.3: Internal architecture of the collision detector. [9]

The MEC offers significant flexibility in the number and placement of collision detectors. It is possible to place collision detectors at the edge of the network thus resulting in shorter delays and latencies or multiple ones in the core network that can cover a wider area.

4.3 End-to-end latency evaluation in MEC assisted C-V2X communications

With the advent with 5G technologies, collective vehicle awareness requires the satisfaction of various parameters. The most important one being End-to-End Latency (E2E) which is intended as the total time needed to transmit information from the moment a vulnerable user signals a dangerous situation until the moment in which the vehicle receives it and elaborates. MEC infrastructures play a crucial role in significantly reducing the E2E latency compared to common platforms.

In a study conducted by *M. Emara M. C. Filippou and D. Sabella* [10] a MEC-assisted V2X architecture has been tested and its results have been compared to a conventional one that does not employ MEC. The model comprised an highway environment with one lane per direction road where vehicles and non-vehicles exchange *Cooperative Awareness Messages* (CAMs) via the Uu radio interface. The latency model describes a one-way time interval that is the sum of:

$$T_{one\text{-}way} = T_{UL} + T_{BL} + T_{TN} + T_{CN} + T_{Exc} \tag{4.1}$$

where T_{UL} is the uplink transmission latency, T_{BH} is the backhaul network latency, T_{TN} the TN latency and T_{CN} the CN latency. The T_{Exc} is the processing time for the CAM messages. Due to this the end to end latency will be made up of:

$$T_{E2E} = T_{UL} + 2(T_{BH} + T_{TN} + T_{CN}) + T_{Exc} + T_{DL} \tag{4.2}$$

Thanks to the advantages of the MEC-assisted framework, the network latencies can be avoided by processing the CAM packets at the MAC host. This effectively removes T_{BH}, T_{TN}, T_{CN}

The results obtained take into consideration variations in the VRU and vehicle density. A greater number of VRUs leads to an increase in the traffic density but thanks to the use of MEC interfaces the exploitation of resources is improved by **66%-80%**. [10]

Chapter 5

Advanced Message Queuing Protocol

In IoT applications, the need of real-time communication systems is of crucial importance. There exist a variety of protocols currently available on the market. This research will only consider AMQP as it serves its purpose in the Rainbow use case.

AMQP is a **lightweight** M2M (Machine-to-machine) developed by John O'Hara at JPMorgan Chase in London in 2003. It is a corporate messaging protocol mainly used in business and it relies on reliable, secure, provisioning and interoperable infrastructure. It is capable of supporting both *request/response* and a *publish/subscribe* architecture. In addition to this it provides a varieties of features such as reliable queuing, topic-based publishing and subscribe messaging, flexible routing and transaction [11].

The AMQP is composed of multiple layers, the lowest comprises an efficient, binary, peer-to-peer, protocol for transporting messages between two processes over the network [12]. On top of that the messaging layer provides an actual messaging formatting that includes its own encoding.

5.0.1 AMQP Transport Layer

In AMQP there exist *nodes* that are connected through *links* and are responsible for the storage and/or delivery of the messages. The link is a **unidirectional** connection between nodes and is attached at the *terminus* point. It is responsible for tracking the delivery and origin of the messages. Once a message has been successfully sent, the receiving node is responsible for its safe storage.

Nodes exist within the concept of containers. A broker or a client, may be examples of containers that are made up of one or more nodes. Each node can

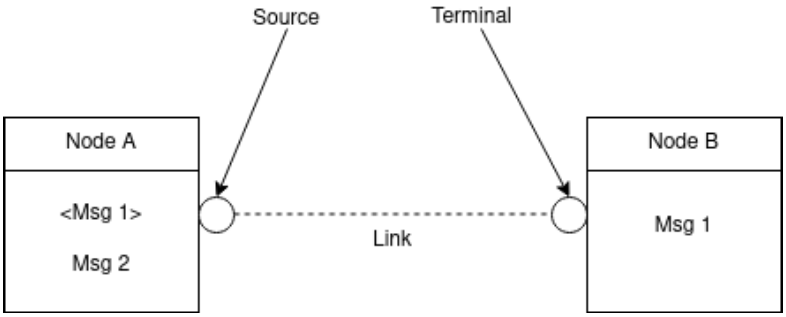

Figure 5.1: Transport between nodes

be a producer, a consumer or a queue. The first two take care of generating and processing messages whereas the third on simply stores them and forwards them.

For communication to happen between two nodes a connection must be established. AMQP defines a **full-duplex**, reliably ordered sequence of frames. A frame is a unit of work that is carried on the wire. If the n^{th} frame has been successfully sent then this means that all previous ones must have been correctly received as well [12]. A **connection** is composed of a series of independent unidirectional channels. A **session** is able to correlate two unidirectional channels to create a **bidirectional** one, capable of creating a conversation between nodes. The flow between nodes is controlled by *transfer frames* that have a maximum size for a specific connection. A connection may have several session, up until the maximum size of the channel.

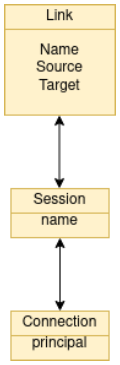

Figure 5.2: Communication between endpoints

5.0.2 AMQP Header

The AMQP header contains information regarding the protocol version that will be used for connection. header is composed by upper case ASCII letters "AMQP"

followed by an **id of zeros** followed by three unsigned bytes that represent the **major**, **minor** and **revision** of the protocol. For a protocol version 1, without revision, the code will look like (1 MAJOR, 0 MINOR, 0 REVISION) [12].

Once the role of client and server have been established, it is the task of the TCP client to immediately send the outgoing protocol header. On the other side, the TCP server may decide to wait a finite amount of time for the incoming header. Both TCP client and TCP server must agree on the version to be used. Any connection after that will refer to that protocol version.

In details the protocol version is chosen accordingly:

- The client must send upon socket connection to the server, the desired protocol header version.
- If the requested protocol is accepted by the server, it must be sent to the socket to successfully establish a connection.
- In the event of a protocol unavailability, the server must reply with the protocol of its convenience and then **terminate** the connection.
- Replies by the server must always choose the upper version, if available.
- If the received protocol header cannot be parsed, the server must reply with a supported protocol and, as before, close the socket connection.

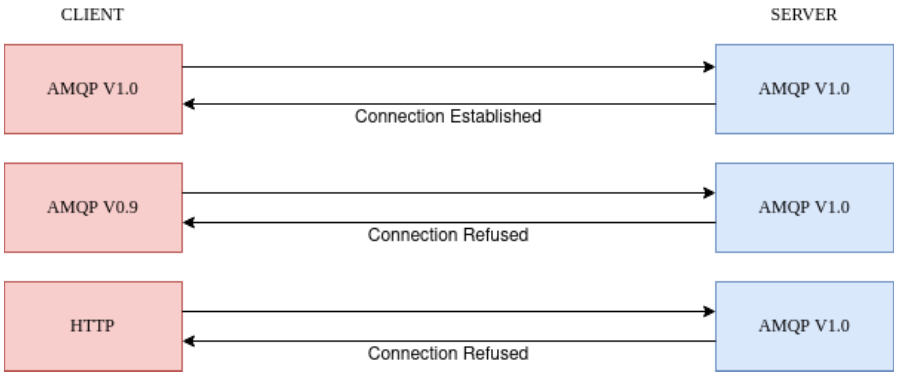

Figure 5.3: AMQP protocol negotiation

5.1 AMQP Frames

Frames can be divided into three categories: **fixed width frame header, variable width extended frame header, variable width frame body** [12]. The frame

header is composed of **8 bytes** which is fixed and is followed by an optional variable width frame header. The frame body, which is also optional, has a variable width as well, and it is the **last** part of the frame.

The rule by which sizes are decided depends on three main factors:

- **size**: bytes 0-3 contain the information regarding the size of the frame. This must contain an unsigned 32-bit integer that defines the sum of the frame header, extended header and frame body
- **doff**: Byte 4 contains the data offset. This gives the position of the body within the frame.
- **type**: Byte 5 indicates the format and the purpose of the frame. A 0x00 code indicates and AMQP frame whereas 0x01 indicates an SASL frame.
- Bytes 6-7 contain the information on the channel number
- The frame is composed of performative elements (*Open, Begin, Attach, Flow, Transfer, Disposition, Detach, etc.*) and of the payload. This last one is defined based on the structure of the performative elements.

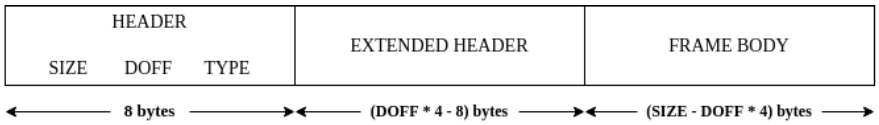

Figure 5.4: Frame layout

5.2 Connections

As previously state a connection is composed of many unidirectional channels that are connected by a session. The exchanges of information happens at the connection endpoints which also track the open and close connection. Endpoints are also responsible for mapping incoming and outgoing channels number and endpoints.

Due to the nature of the connection (unidirectional), incoming and outgoing channels are completely distinct from each other. This means that bidirectional connections do not share the same channel.

5.2.1 Connection opening

The connection is instantiated on the basis of the limitations. A **maximum frame size** of **512** and **maximum channel** of **0** are considered [12]. Such information

are used in the opening frame before beginning any transaction. Due to to the channel rule, opening frames can **only** be sent on channel **0**.

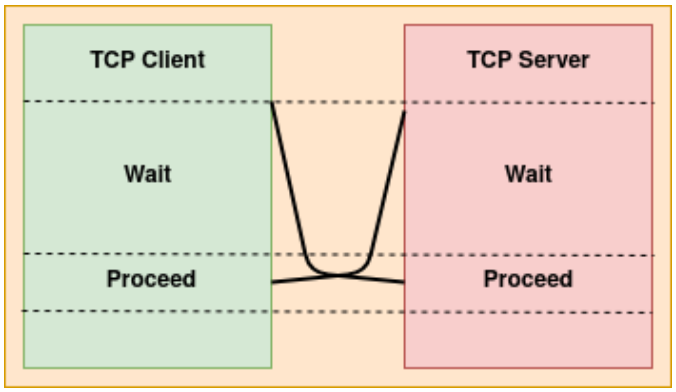

Figure 5.5: AMQP Connection Opening

5.2.2 Connection closing

Similarly to the connection opening, the closing requires both peers to instantiate this procedure. This is done through a specific closing frame that justifies the actions. Once this frame is sent, the peer can no longer send frames. It must, however, wait for the reply from the other peer with a closing frame acknowledging the reason for the termination. All acknowledges are considered within a timeout window to guarantee protection against malicious frames. Contrary to the opening, the closing can happen on **any channel**, however **channel 0** is preferential [12].

5.3 Sessions

Once a connection has been established, it is the task of the session to coordinate unidirectional connections in such a way to create a conversation between containers. Sessions serve the purpose of contextualizing links for communication. Sessions have a one-to-many property, that is, they can connect to multiple links at the same time. The opposite is **NOT** true. Links can interface with **at most** one session. Links within a session share common features, such as delivery features that allow the coordination of multi-link applications. This is used to simply the acknowledgment through the single use of sessions.

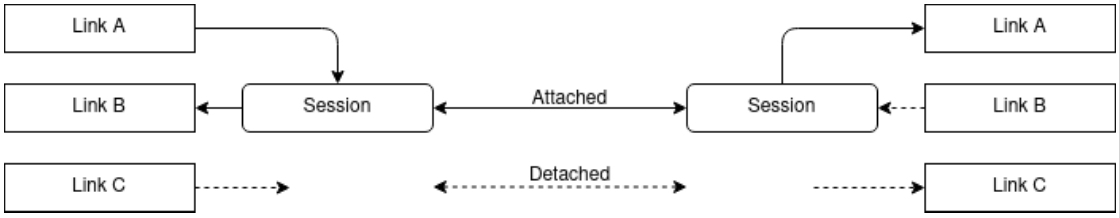

Figure 5.6: AMQP session description

5.4 Links

As described above, links provide a unidirectional method of communication between the Source and the Target. Link Endpoints have the task of interfacing the Terminus and the Session Endpoint. Links may come as *Senders* and *Receivers* respectively. When an sending application submits a message to be sent it must deliver a tag used by the Source to track state of the message during delivery. Endpoints have also the task of assigning univocal delivery ids that are used track subsets of outstanding deliveries on a session [12].

5.4.1 Establishing and Resuming a Link

When a link is first established the Link Endpoint assigned an unused handle and sends an attach frame. This includes the information of the local and remote termini that will eventually become the source and the target of the communication. The Remote session endpoint receives the *attach* and thus creates a Link Endpoint. When a link is resumed the conditions that were present before may have changed. The Source and Target can be different, this leads to conflict between the Termini properties in the source and target fields of the attach frame. In such a case it is the Sender that holds the properties of the Source and the Receiver the ones about the Target. In the case that the properties differ the link will detach, otherwise it will continue freely [12].

5.4.2 Closing a Link

The closing of a link is done by sending a *detach* frame with a close flag set to true. The peer partner will destroy its states and corresponding link endpoint. In the case in which the Peer and the Partner are sending conflicting information, the one that is sending the detach frame must issue a new attach frame that signals the intention of terminating the link [12].

5.5 Flow Control

To control the message transfer, a flow control mechanism is employed. This is done through the Link Endpoints that maintain the state of the flow. Flow control decides when a message can be sent over an attached link and controls information such as the *link-credit* [12]. This is used to understand when there are no more messages to be consumed or when there is not sufficient credit to sent the messages. Other flow control parameters are [12]:

- **Delivery Count**: It tracks the number of successfully sent messages. It can also be employed to track Received messages
- **Link-Credit**: The credit defines the amount by which the delivery count can be increased. Receivers can select this value whereas Senders must always attain to the chosen by the Receiver.
- **Available**: When a dedicated amount of link-credit is available, the senders signals to the receiver that it is **available** to utilize them. In the case in which the available is null, the sender can still send messages.
- **Drain**: When insufficient messages are available to be consumed the credit-flow the sender will increase the delivery count as much as possible in order to consume all credit.

If the link-credit is equal to zero then the delivery-count is equal to the delivery limit since [12]:

$$link - credit = delivery - count_{rcv} + link - credit_{rcv} - delivery - count_{snd}$$

5.6 Messages

5.6.1 Message format

It is important to distinguish between a message as seen from the sender and the one seen from the receiver. The first is usually called a *bare message* that contains an immutable payload that is handed to the messaging infrastructure. The receiver sees an *annotated message*, that is, the *bare message* plus annotated information that have been found along the infrastructure [12]. Annotated information are usually provided at the beginning and/or at the end of a message.

The bare message is immutable thus cannot be changed by an intermediary node that is found along the way. This is important because the encoded message cannot be modified. The message is composed of the following sections [12]:

- **Header**
- **Delivery** annotations
- **Message** annotations
- **Properties**
- **Application** properties
- **Body**
  - **data** section
  - **amqp sequence** section
  - **amqp value** section
- **Footer**

The **header** defines delivery details of the AMQP network. The *durable* field defines if the durability requirements; if an intermediary fails during the network it must guarantee that the message is not lost. The *priority* field indicates the priority of the message. A high number indicates an higher priority indicating that this message must be delivered before all the others with lower value. *Ttl* (time to live) defines the time in which the message should be considered to be alive. If a message lasts longer than defined in the ttl, it will discarded. A *first-acquirer* message indicates that no other link has acquired the message before . The *delivery-count* records the number of unsuccessful attempts; a message with non-null delivery may indicate the presence of **duplicates**.

Message and Delivery annotations consist in a series of information that are delivered to the receiver and are used for the delivery and message properties. In the case in which annotations are omitted, the equivalent section consists in an empty map of annotations. If they are present, they are used to describe properties which are immutable parts of the bare message.

The message **field details** contains among the most important information. The *message-id* defines the unique code to identify a message within a message system, it is not immutable as it can be easily modified by brokers and nodes. *User-id* on the other hand identifies the the user that originated the message. It may only be confirmed by intermediaries, but cannot be changed. The *content-encoding* holds information on the encoding properties of the message. If present it indicates that the application has been encoded with **additional encodings**; this requires specific decoding structures on the receiver side to decode the message.

The **body** can be made up of simple opaque binary data, AMQP sequence or AMQP value. The sequence contains an arbitrary number of structured data [12]. The AMQP value contains only a single value.

5.6.2 Transactions

AMQP transfers become entangled one to the other when a transaction occurs. One part of the transfer is considered as the *resource*, the other as the *controller* [12]. The controller is responsible for regulating the transaction by declaring it and discharging it. This is done through the use of an identifier that is confirmed by the *coordinator* (a resource target responsible for the transactions).

A transaction can be described with its state.

- posting or making available
- acquiring the message at the source level
- retiring or terminally ending it

The posting takes place when incoming transfer frames arrive at the controller. They then will take the acquired state when the incoming flow is initiated and terminated with the frame disposition. It needs to be specified that these flows can be **directional** therefore they can move from and to the controller.

5.7 Security

AMQP employs a security layer for encryption and authentication where the traffic can safely flow. In the case of Transport Layer Security, this has to be externally defined. On the other hand, SASL security layer has to be internally defined on its host protocol to provide framing [12].

5.7.1 TSL

As discussed in the beginning of the chapter peers begin their communication by exchanging a header frame containing protocol information. The major, minor bytes define respectively:

1 (TLS-MAJOR) 0 (TLS-MINOR) 0 (TLS-REVISION)

4 OCTETS	1 OCTET	1 OCTET	1 OCTET	1 OCTET
"AMQP"	%d2	MAJOR	MINOR	REVISION

Figure 5.7: TLS Security Layer Protocol

5.7.2 SASL

Similarly to the TSL security layer, the SASL must be established through an header frame that is sent at the beginning of the connection between peers. The schema is identical to the one shown in figure 5.7. The main difference from the TSL is that SASL needs to be defined internally through frames.

A SASL frame consists of an header, extended header and a body containing the SASL mechanisms, SASL init, SASL challenge and SASL response as well as the SASL outcome. At this point TCP client and TCP server begin a **negotiation** . This process defines the supported authentication using a SASL-mechanism that when agreed, can begin the exchange [12].

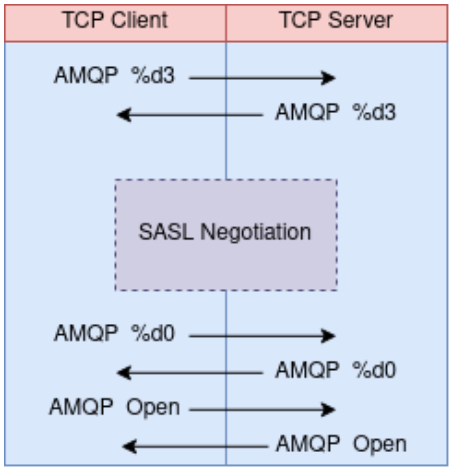

Figure 5.8: SASL establishment through negotiation

Chapter 6

The Rainbow Project

The increasing number of smart devices has pushed the massive use of cloud platforms; particular benefits can be obtained with little infrastructural investments. Given this unidirectional trend, it is easy to expect that almost all devices will exploit cloud services. However, this has raised several issues concerning **privacy, security, latency and bandwidth** [13]. In the automotive field, particularly for connected vehicles, these constraints can result in serious issues for the performance and the safety of the users. To overcome such a problem, it is becoming common sense to exploit the **network edge** to speed up the process of data transmission and elaboration [14]. This thanks to the possibility of completing tasks that would otherwise strain normal cloud data centers at the core network. In the modern vehicles camera sensors produce huge amount of data that can be utilized to perform precise evaluation on the status of the road and of the other vehicles. Edge computing not only provides direct services to the user but it can handle computational tasks such as processing, storage, caching and load balancing [14].

Fog computing enables computations at the extremes of the edge. They are used to store and compute data, coming from devices allocated at the network extremes. This data are then forwarded to the cloud infrastructure [15]. There is a subtle difference between **fog computing** and e**dge computing**. Although they perform similar tasks and their advantages are overall the same, edge computing takes place at the data source, **manipulating data before they are transferred to the Cloud**. Fog computing on the other hand is **part of the Cloud Computing power** that takes place locally, this indicates an extension of calculating power from the network core to the extremities of the edge. Even with the use of fog computing, challenges remain. This is further underpinned by the need of performing such computations on weaker hardware [16]. It is of great importance the need of considering fog nodes as not static entities fixed in position but as dynamic elements that interact with the IoT and Cloud Services. It is matter

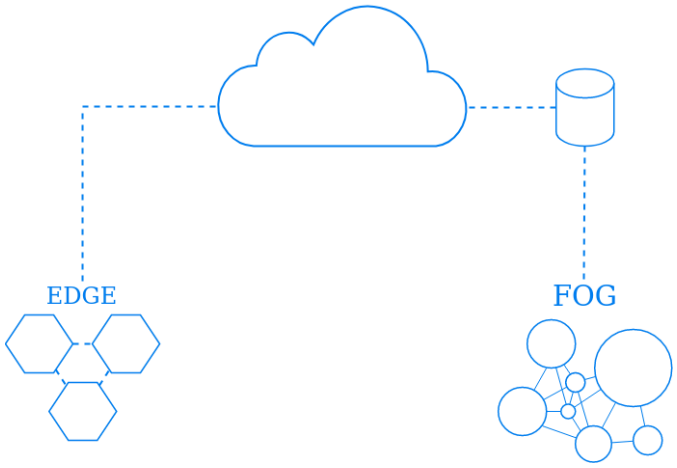

Figure 6.1: Fog Computing and Edge Computing

of growing discussion the need for optimizing efficiency, power consumption and manage their lifecycle. This requires a service capable of migrating computational resources and data management to better balance efficiency and power deployment. In the automotive sector in particular, it is of crucial importance the computational power modulation that can guarantee quality of data process and speed of transfer.

The aim of the RAINBOW project is to create an **open and trusted fog computing platform** that facilitates the **deployment and management of scalable, heterogeneous and secure IoT services and cross-cloud applications**[17]. This is done with the prospect of delivering a platform capable of managing hundred of edge devices, thousands of fog nodes and possibly millions of vehicles [17]. The project aims at demonstrating the possibility of providing deployment, orchestration and data management for edge applications. This is done by pushing computational powers to the network edge whilst also keeping security and reliability. Rainbow divides its implementation into three use cases:

- Human-Robot Collaboration in Industrial Ecosystems
- **Digital Transformation in Urban Mobility**
- Power Line Surveillance via Swarm of Drones

This research piece will only consider the second **use case**.

6.1 Digital Transformation of Urban Mobility

The goal of RAINBOW is to create a real-time geo-referenced notification system that can be employed by vehicles traveling in urban areas. Notifications will

empower vehicles that are in critical areas (i.e an hazard has been detected on the road). The steps needed to achieve this goal hinge on the creation of the following steps:

- Disseminating information across the traffic
- Aggregating data and making it available (**crowdsourcing**)
- Exploiting fog computation resources present in the edge network
- Identifying and forwarding alerts to the relevant users in the areas of interest

6.1.1 Alert detection, AHED

Roads can present several sources of hazard, each of them can be detected by vehicles, road side units, sensors or even users. Notifications can be of two types:

- **Automatic notifications**: detected by road side units, vehicle sensors or through sensor fusion.
- **Explicit notifications**: that are triggered manually by a vulnerable user.

In the case of automatic notifications, **data fusion** refers to the aggregation of Road Side Units and vehicles that send data directly to the Fog node, MEC or Cloud. For the identification, AI/ML algorithms are exploited to identify hazards and generate an appropriate notification flag associated with it. These alerts are conveyed to a Traffic Control Center (TCC), that holds information regarding **geo-localization, client digital signature, textual information and perhaps media data.** Due to the heavy computational resources required to administrate such a task, the service can be distributed at the MEC edge to guarantee low levels of latency and high performance metrics. The AHED (Automatic Hazardous Event Detection) is tasked with detecting and identifying the hazards [17].

The main challenges for such a configuration hold in the ability to:

- Create a trusted and secure virtualized orchestrator capable of **reporting and updating** the information. This encompasses not only the mere hazard identification but also being able correlate fused data with the manual notifications provided by users.
- Realizing an **optimization algorithm** capable of splitting the resources from local, the edge and the cloud backend.
- Identification of preferable MEC positions to balance **coverage** capability and service **stratification** (i.e possibility of generating multiple user services).

- Merging of the aforementioned needs with the expanding C-V2X PC5 technologies.

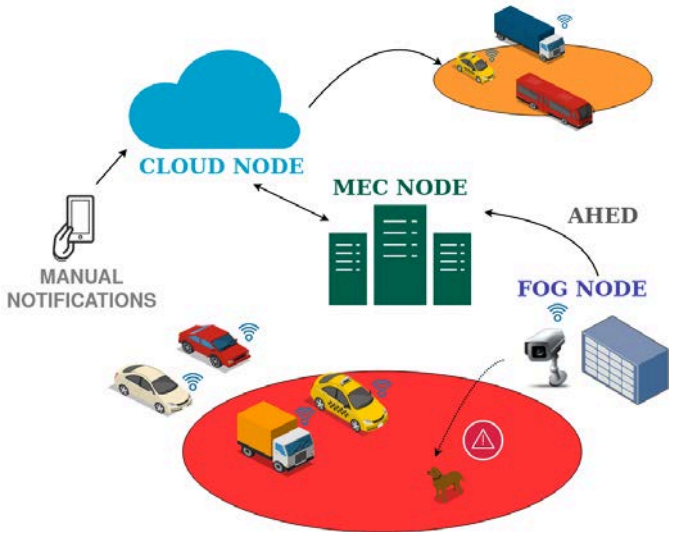

Figure 6.2: Automatic Hazardous Detection configuration

6.2 FOG node for hazard identification

The fog node utilizes computational capabilities and image recognition algorithms to understand the current status of the road. It is seen that fog nodes are extensions of the cloud level in order to provide services closer to the edge-network [18]. In evolving smart cities has pushed the need of the computational powers closer to the user. The geo-distributions of elements in cities makes it necessary for the big data to be processed closer to the sensor [19]. A scalable multi-layer approach described in [19] highlights the importance of stratifying services and resources in order to balance the need for coverage with the necessity of low latency services. Cloud computing elements present at top level provide monitoring and control [19]. Lower fog layers provide subsequent narrow coverage area but more reliable quickly adjustable gear. The fog node presented in 6.2 provides the example of a low level, low latency device capable of sensing hazards and producing alerts. In addition, these fog nodes represent the **closest access point** to which it is possible to interact with the other vehicles present on the road.

Due to the importance of such devices, it is crucial to discern malicious nodes from trusted ones. As described in [19], a third-party security provider can establish a validation procedure that can be used before data transfer occurs

between IoT devices and the node interact. FOG nodes among themselves can cooperate to provide backup and assistance in case of dynamical variations in the service provider.

6.2.1 Fog implementations: Camera

Interfaces with the road can be achieved with a variety of sensors. Cameras play a crucial part in this, as they can provide with a clear definition through image processing techniques capable of extracting peculiar information of an object almost instantaneously. To properly control a segment of the road, cameras would need to continuously, scanning for situations of interest and analyzing them in order to provide a clear an univocal description. This poses strains on the bandwidth of the system and would result in poor latency performances.

Fog nodes connected to cameras prove to be extremely effective in achieving a mass-scale coverage area, thanks to their **compact size** and **low cost** nature. The use of multiple sensors allows to detect of concurrent and independent events that can be promptly reported to any requesting application [20]. Primary detection is associated with the embedded system that provide computational power and storage [21]. Thanks to the nature of the fog infrastructure, it is possible to achieve highly accurate algorithms with far lower latency than conventional direct-cloud computing. In [22], it is shown how a possible real-time surveillance system can achieve powerful results when it is coupled with fog computing. Latency performances are **two orders of magnitude** slower than the direct-driven Cloud computing architecture.

In the Rainbow project [17], the RSUs are equipped with **IP cameras that work as fog computing nodes**. At the same time, the video stream is also reachable from the MEC node. Cameras are capable of implementing image detection algorithms in order to identify the presence of an animal on the road. Thanks to the dual nature of the infrastructure, it can either run on the Fog node or on the MEC node, depending on the appropriate situations. Good streaming quality is achieved at the Fog node thanks to its vicinity to the source, **this provides accurate identification results**. On the other hand, good accuracy requires high computational power that may lead to excessive **overheating**, resulting in a possible failure. The MEC node does not suffer this problem thanks to its powerful computational resources. However, this results in **higher latency** and **lower image quality**, leading to poorer identification results.

6.2.2 AMQP Message Flows

In the event of a detection, the on board cameras recognize an hazardous situation on the road. The RSU servers a double purpose. When hazards detected on

Running Node	Advantages	Disadvantages
Fog Node	• Low latency •High detection accuracy	• Low computational power • High power consumption
MEC Node	• High computational power • Low power consumption	• High latency • Low detection accuracy

Table 6.1: Fog computational advantages compared to MEC node

the road need to be broadcasted to the **Red Zone**, the PC5 interface is utilized. When the detected hazards need to be transmitted towards the **Orange Zone** an AMQP Client is employed which embeds DENMs/CAMs according to the structure described in chapter 5.

When the computational resources required are so great that the FOG node risks overheating, the MEC node is called into action. The latter has to perform DENM generation and act as an AMQP Client. Contrary to the Fog node, the MEC node has to generate information exclusively to the Cloud Data Center. This allows DENMs to be forwarded to vehicles in zones of less interest (Orange Zones) or be processed by the Cloud aggregating architecture for map visualization.

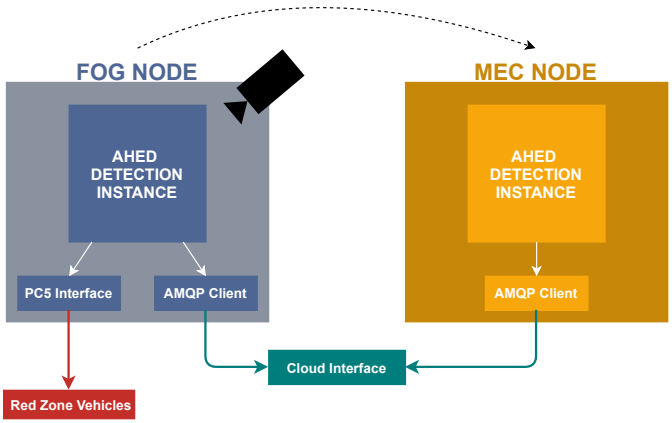

Figure 6.3: Fog/MEC messages generation

6.2.3 On-Vehicle Message Flows

Vehicles equipped with a C-V2X technology communicate with the nearby RSUs to exchange messages. On one side, vehicles receive DENMs generated by the RSU or forwarded from the Cloud Infrastructure, on the other, they act as clients for Cooperative Awareness Messages generation.

CAMs are exchanged between vehicles in order to notify them about their

presence, position and status in **single hop** distance [23]. CAM messages are generated according to the rules describe di chapter 3, the basic algorithm behind the information generation behaves accordingly [24]:

Algorithm 1 CAM generation algorithm as described by ETSI

Input: Position p; position history pHist; last CAM message sent lastCam
Output: New Cam message, lastCam updated; pHist updated with p

```
1: procedure CAM GENERATION(p,pHist,lastCam)
2:     ▷ p is the new position, pHist is the position History
3:     while true do
4:         time ←System.getTime()
5:         heading ← calcHeading(pHist,p)
6:         lastPos ←lastPosition(pHist)
7:         lastHist ←pHist/ lastPos
8:         lastHead ←calcHeading(lastHist,lastPos)
9:         speed ← calcSpeex(pHist,p)
10:        if p ≠ null then
11:            lastSpeed ← calcSpeed(lsatHist,lastPos)
12:            if distance(p, lastCam.pos) ≥ D-THRESHOLD or
13:                |heading − lastCam.heading| ≥ H-THRESHOLD or
14:                |speed − lastCam.speed| ≥ S-THRESHOLD then
15:                cam ← newCam(time, p, heading, speed)
16:                sendCam(cam)
17:                lastCam ←
18:            pHist ← pHist ∪ p
19:        else
20:            p ← lastPos
21:            heading ← lastHead
22:        if time - lastCam.time ≥ T-THRESHOLD then
23:            cam ← newCam(time, p, heading, speed)
24:            sendCam(cam)
25:            lastCam ←
26:        System.wait(CHECK-PERIOD)
27:        ▷ The system check guarantees the CAM generation frequency of 10 Hz
```

The values of D-THRESHOLD, H-THRESHOLD, S-THRESHOLD and T-THRESHOLD are the ones reported by ETSI:

- **D-THRESHOLD** = 5 m
- **H-THRESHOLD** +/- 4°

- **S-THRESHOLD** = 1 m/s
- **T-THRESHOLD** = 1 s

This guarantees that all vehicles can be properly mapped with a continuous flow of information that guarantees **reliability** and **accuracy**.

Information gathered at the sensor level allows to register speeds, and directions as well as other important parameters. Those are then routed through the CAN bus in order to be registered by the V2X management system. The AMQP Client is, too, connected to the CAN bus, it takes part in a bidirectional flow of information that comes from the Uu interface. Data collection and data generation happen mainly with respect to the Cloud Infrastructure that is a forwarder as well as a collector of CAM messages. These are useful to aggregate information on a Citizen Application containing DENMs as well as CAMs.

The PC5 interface on the other hand is employed in high risk areas (Red Zones) and directly communicates with the Fog RSU. In this case, Hazardous Notification Messages (DENM) are subject of interest for the vehicles, which after parsing the messages, is able to extrapolate the necessary information. If the vehicle is located inside the perimeter defined by the GeoNetworking header (**further details in successive chapters**) it notifies the drivers through an HMI.

In this sense the vehicle is a complex entity that has to manage bidirectional flows coming from Cloud and Fog interfaces. Processing speeds are limited by the CAN bus and the speed at which the V2X management unit can process data. The PC5 and Uu stacks manage the PC5 and the UU communication layer [17]. The AMQP client negotiates a connection with the cloud broker and then can either receive or send a stream of data (CAM, DENM) messages, **properly encoded** and **safely encrypted**. The AMQP client must also be able to extract sufficient information that have to be sent to the V2X management unit. The same piece of equipment manages and orchestrates all processed within the vehicle; sending and reception are in fact regulated by it. Finally the HMI sets-up a sequence of policies in order to guarantee a proper interaction with the end user.

6.2.4 Cloud Node

The role of the Cloud node is to bridge far communicating users in a **fast** and **reliable** manner. This is achieved through aggregating methods and smart traffic flow management. A **single broker** is employed, the use of topic categorization allows for reception of different type of messages depending on their source.

Signals issued from vehicles are collected, decoded and finally extrapolated. With this, the City Aggregator is able to implement the real-time geo-localization paradigm previously described. Another flow of message coming from the RSU is

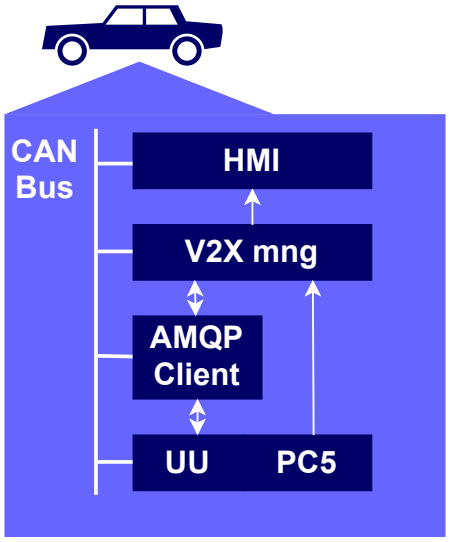

Figure 6.4: Vehicle information flow

forwarded directly to the vehicle. This contains alert messages that have been issued from the Red Zone and are directed for the vehicles belonging to the Orange Zone.

Finally, the Cloud Interface will collect data coming from the Vulnerable Road Users that are present on the road. Such information must be confirmed by RSU before being considered trust-worthy.

Figure 6.5: Simplified Cloud Node Flow

6.2.5 Use Case Scenario

After a brief description of the underlying architecture, it is important to describe the scenarios that take place.

From the Cloud management point of view, in real case implementations, the ecological impacts due to power consumption become a **non-negotiable** aspect. virtualization techniques have helped reducing the number of physical server thus decreasing **overall power consumption** and **utilization levels** [25], however this is not enough. A **Smart Orchestration** technique must be implemented to minimize the usage of such a powerful computational resource in non-necessary situations. The working conditions can be subdivided into two main categories:

- Non-hazardous situations
- Hazardous situations or Congested Network

In the case of **Non-hazardous situations**, when there are no detected hazardous events, the RSU will send **low-resolution** and **low frame-rate** footage directly to the MEC node. In this case, the AI image recognition algorithms will continuously analyze images. The RSU can be wireless or wired connected and are energized by the electric grid. In standard conditions the network bandwidth is enough for the video streams so the RSU can run is low-power mode. Most of the operations happen at the MEC node where they are already efficiently managed [17].

Differently, two possible situations can trigger another event:

- **Detection of hazardous situation:** In this case all computational resource powers must be switched to the zone where the event has been detected. This means moving the AHED algorithms from the MEC node towards the Fog node. Higher image resolutions and higher frame rate are needed to insure a more stable and reliable image detection. Furthermore thanks to the relatively short distance, the latency is significantly reduced. Alert messages are directly produced at the RSU level and are dispatched to the vehicles through the PC5 interface. Away vehicles are alerted exploiting the AMQP broker positioned at the Cloud level. This does not necessitate of **minimal latency** constraints due to the lower priority. After the event has ceased to exist all power resources are allocated back to the MEC node [17].
- **Congested network:** if extremely high levels of latency are detected at the MEC node due to congested traffic, the orchestration may decide to switch back and forth the computational power resourced towards the RSU [17].

6.2.6 Smart Orchestration

The role of the Rainbow platform is to manage and organize **load-shifting**, moving computational powers back and forth between elements of the architecture. This is

Scenarios	Fog resources	MEC resources
Non hazardous event	• Low quality • Low frames •Power saving mode	•AHED algorithms • Low power consumption •Efficient
Hazardous event	•Full resource allocation •Low latency •High definition and frame	•No AHED algorithms are employed •Power allocation towards the FOG node
Congested network	•Fog support towards MEC	•High level of latency •Reallocation of computational resources to the Fog nodes

Table 6.2: Resource orchestration for different use case scenarios

achieved by analyzing several metrics among which latency and bandwidth usage measured by service graph networks and policy editor [17].

These data are evaluated in an effort to satisfy as many constraints as possible:

- Bandwidth usage
- Latency
- Power consumption
- RSU temperature
- Event Detection

Powerful algorithms will balance the need for fast and reliable AHED without compromising too much power consumption and risking hardware overheating.

Load balancing is achieved by means of **Slicer** [26], a powerful general purpose sharding device developed by Google that is capable of monitoring load hotspots, server health and to dynamically allocate work over a set ot servers. This is done through the use of keys that that allow for re-balancing. On the other hand, **Accordion** technology achieves scalability resolving the issue of dynamic data placement [27].

6.2.7 Security and Data Storage

Due to the significant number of sensitive data, particular care has to be taken towards security measures. Security should be considered in both the communication, during data exchanges, as well as in the data itself by using authentication and encryption [28]. The means by which it is possible to guarantee such requirements is through the use of:

- SSL and TSL protocols coupled with the use of a VPN

- Digital Certificates

Encryption of data through **keys** and **authentication access controls** can be solutions exploited by the DBMS (Database Management System).

RAINBOW provides remote attestation by means of trust-aware service graph which can guarantee both authentication whilst keeping user data private [17].

6.2.8 RAINBOW Functionalities

Requirement	Monitoring network load
Goal	Control the network flow produced by the RSU and keep the power usage levels below a certain threshold
RAINBOW Functionalities	**Modelling** -Constraint and policy editor -Centralized application packaging **Orchestration** -ALC (Application Life Cycle) management -Runtime and resource adaptation
Workflow	**Configuring video streams from the RSU to the Edge Node** -Low resolution -Low frame per second **Begin AHED service** -Low accuracy -High latency **Bandwidth monitoring** -If above SLO then: i. Move AHED to RSU ii. Stop video stream towards edge node

Table 6.3: Network load administration from RAINBOW

Requirement	Ensuring hardware working regime
Goal	Control the RSU temperature to keep it under safety limits
RAINBOW Functionalities	**Modelling** -Constraint and policy editor -Centralized application packaging **Orchestration** -ALC (Application Life Cycle) management -Runtime and resource adaptation
Workflow	**Start AHED service on RSU** -High resolution -High frame per second -High accuracy -Low latency **Monitoring RSU temperature. If above SLO threshold** **Configure video stream to flow from the RSU toward the Edge Node** -Move AHED service on the Edge Node

Table 6.4: RSU temperature administration from RAINBOW

Requirement	Maximize available resources and accuracy in the case of hazardous events
Task	AHED will run on Edge Node up with low resolution. When Hazardous event is detected it will move to the Fog Node, high resolution and high frame rate. When event ends, all resources will be re-allocated to the Edge again.
RAINBOW Functionalities	**Modelling** -Constraint and policy editor -Centralized application packaging **Orchestration** -ALC (Application Life Cycle) management -Runtime and resource adaptation
Workflow	**Stream flows from RSU towards Edge Node** -High resolution -Low latency **Start AHED service on Edge Node. If hazard is detected:** -Move AHED service on the RSU i. High accuracy ii. Low latency **If event expires:** -Move AHED back to the Edge Node

Table 6.5: Resource administration from RAINBOW

Chapter 7

Cloud Node Framework

The Cloud node that has been described in section 6.2.4 highlights the main interfaces that play a role in message exchanges. The overall structure is composed by **single broker** that interacts with AMQP Producers and Consumers.

Three main flows of information can be highlighted:

- Fog/MEC node towards Cloud Node
- Vehicle to Cloud
- Citizen App towards User

For this reason, the node must be composed of three consumers, one per each direction, and one producer that forwards messages directly to the AMQP Client available on the vehicle. The node has its central element revolving around a message decoder that exposes the ETSI messages. The information contained inside are used to aggregate data in a centralized map, publicly accessible.

7.1 Message Emulator

The need for a message emulator originates in the ability to simulate the complete chain of transmission within the architecture. A message that has been created can be used to measure end-to-end latencies by comparing the **generation time** with the time taken for **transmission** and **decoding**.

Before being able to send a message across the channel, CAM/DENMs must be generated, this is done through the use of encoding techniques. In order to guarantee security during transmission, ETSI defines a set of Public Key Infrastructures (PKIs) for C-ITS environments [29]. However, no specific standard is defined regarding *Abstract Syntax Notation One* (**ASN1**) [30].

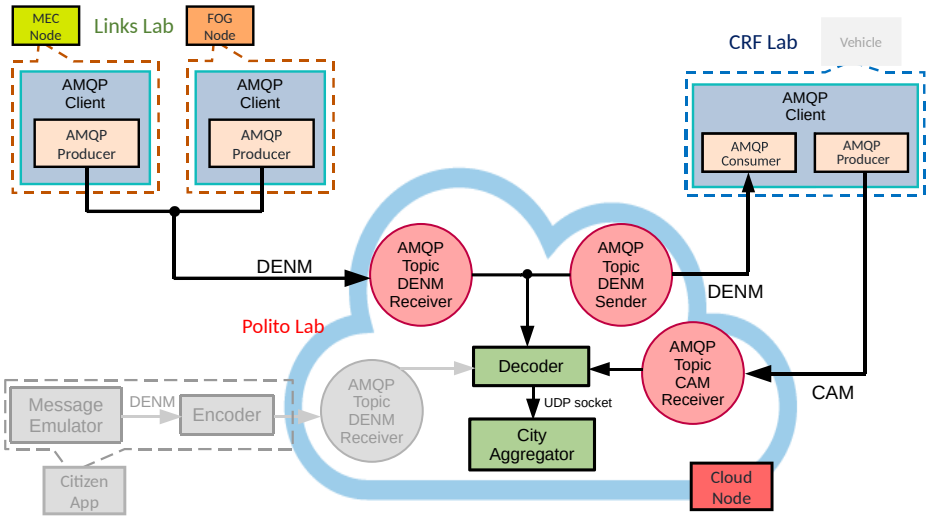

Figure 7.1: Cloud Node internal structure

Abstract Syntax Notation One is a standard widely used in telecommunication systems in order to specify the data used. The aims of ASN.1 is to specify data-structures that can be independent on the language and platform used. It, therefore, defines methods and rules by which these data-structures can be stored in a **binary fashion**. The final encoding that results after the operation is a **concatenation** of the encoding realized on all the structures composing the root ASN.1 definition [30]. Conversely, in the decoding phase, these structures must be predefined in order to successfully reconstruct the original message. The introduction of 3GPP saw the integration of the UPER encoding rule in order to describe signaling messages in the LTE-Uu interface protocol stack [31]. *Packet Encoding Rules* and *Unaligned Packet Encoding Rules* (PER, UPER) aim at eliminating redundancies contained in packet encoding rules such as in the case of BER. **The encoding of a structure yields an array of bytes**. Not all encoded structures represent a complete byte array value. A structure described in 6 bits can be represented with a single byte (8 bits) or just as an array of length 6. UPER version does not implement any padding in the unfilled octects, common practice usually employed in PER encoding [30].

In AMQP Senders the use of UPER encoding will be exploited. This procedure takes place at the beginning of the encoder where the ASN.1 file is **initalized**. This will be reused whenever a new message is created, making the procedure lightweight and reliable. The ASN.1 includes a ITS-PDU header description at

the beginning of the file which includes information on the protocol version, ITS station ID, and message type. In **appendix A** the ITS-PDU Header is defined with respect to the Cooperative Awareness Messages.

The emulation process begins with the transformation of a textual file containing the bare ETSI message. In the case of DENMs, this will hold information regarding the **cause code** and **subcause code** positioned at the *situation* container level. In the case of CAMs the critical information are those described in the **algorithm 1**. The textual message if firstly converted to string so that it can be managed by the encoding software. For this particular procedure, the Python library **ans1tools** will be employed [32]. On top of the encoded message, the Basic Transport Protocol and Geonetworking will be encapsulated [33], according to the structure in figure 7.2:

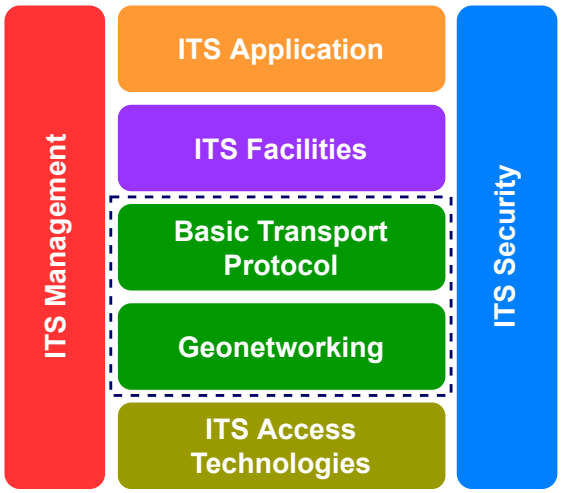

Figure 7.2: Geonetworking stack in ITS

Algorithm 2 Message emulator

Input: ASN.1 ITS-PDU *asn1*; latitude *lat*; longitude *long*; GeoNetworking *geo*; ETSI Message Body *msg*

Output: Encoded CAM *encoded*

```
1: procedure ETSI MESSAGE EMULATOR(asn1,lat,long,geo)
2:     time ←System.getTime()
3:     asn1.compiled← asn1tools.compile(asn1)
4:     msg_string ← string(msg)
5:     msg.latitude ← lat
6:     msg.long ← long
7:     while true do
8:         timestampITS ←convertEpoch(time)
9:         ▷ Converting internal clock time to ETSI reference time
10:        msg.timestamp ← timestampITS
11:        encoded ← encode(msg)
12:        msg_encoded ← geo ⋃ encoded
13:        ▷ Encoded body is encapsulated inside BTP+Geonetworking header
14:        AMQPSender(msg_encoded)
15: ▷ These procedures are not specific for a particular ETSI message type
```

To properly simulate the movement of vehicles within a certain geographical area, a **set of latitudes and longitudes** will be generated. Those will try, as best as possible, to emulate the probable path that a vehicle may undergone. Such a set of latitude and longitude are then included prior to the encoding phase within the ETSI CAM message. Similarly, a series of events will be generated and their position will be mapped onto the City Aggregator.

7.2 Qpid Proton: AMQP Sender/Receiver

The development of AMQP Senders and Receivers has been done exploiting **Qpid Proton**, high-performance, lightweight messaging library that can be employed to develop brokers, clients, routers, bridges, proxies and more [34]. As extensively described in section 5.4, AMQP message transfer occurs between peers upon the establishment of links. The sending peer calls the link *Sender*, at the receiving peer it takes the name of *Receiver*. Links may contain the *Source* or *Target* address where the message has to be sent. Links can begin only when sessions have been established and sessions take place only when connections are generated. To generate a connection, containers are used that are uniquely identified. A single connection may contain multiple session; generally this is a rare case but it is still

possible. Qpid generally ignores session unless explicitly required [34].

A message once sent over a link is said to be *in delivery*. The term *message*, as described in 5.6.1, contains an immutable payload containing all information of interest, header, application properties, and message annotations. The exchange is identified with the transfer of the data content.

To confirm a delivery either *Sender* or *Receiver* can settle it. Upon this action, the other side has to be promptly notified of the successful delivery. Further communications at that point are halted.

7.2.1 AMQP Sender

The AMQP Sender in the architecture described in 7.1 is mainly tasked with the information forwarding towards the vehicle. Due to the experimental nature of the analysis, the entire flow of messages has to be simulated.

AMQP Producers present at the Fog/MAC level are simulated with a **Qpid Sender instance** that generates flows of DENM data. This is done in an effort to simulate all levels of communication, even those outside the scope of the Cloud Node.

The basis of the process is based on the creation of an application logic in a class that handles various events [34]. Such events are handled by the **Proton event handler** and are defined within the application container with the following classes:

- *on_start*: When an event loop is **initialized**
- *on_sendable*: When the link has sufficient **credit**, thus permitting message flow
- *on_accepted*: When the remote peer **acknowledges** the message coming from the sending peer
- *on_disconnected*: Called only when the **connection socket** has been **closed**

All these events are contained inside a **container**, which is a class imported from Reactor. It allows for easy programming thanks to an **event loop** which is able to react to the events listed above. The event classes contained inside will then notify when their are being triggered.

The Container, upon initialization, is served with the address location of the broker that is usually defined according to the follow schematics:

```
1 IpAddress:5672/topic://rainbow.project.crf.polito.cam.0.0.1
```

Where the first element passed is the destination broker address and its port. AMQP defines a default port *5672* for all AMQP transmissions. Subsequently the topic is defined.

Although available, **queues** are **not preferred** over topic. A real-time notification service cannot be adopted due to their intrinsic nature to store messages in queues that are destined to specific consumers. This means that until ready, a notification message will remain stored within the broker until a consumer subscribes to the queue. A **non-defined delay time** will be included in the reception of the message which can lead to severe discrepancies between what the vehicles see and what is the actual situation of the road.

On the other hand, topic exchanges allow for multiple consumers, that are interested in the same topic, to receive the notification parallely. In the use-case scenario described in figure 6.2 all vehicles belonging to the Red Zone will be notified simultaneously regarding the same message:

topic://rainbow.project.redzone.*.*.*

Topics, in addition, employ powerful wildcards "*" that allows vehicles subscribed with topics such as

- **topic://rainbow.project.redzone.1.0.0**
- **topic://rainbow.project.redzone.0.1.0**

to receive the same hazardous notifications.

In addition to the message destination address, the encoded message is included in the initialization event belonging to the container. It is then sent along with a **unique** identification code through the *event.sender.message* command.

7.2.2 AMQP Receiver

Similarly to the AMQP Sender, the Receiver is instantiated inside a container where events are handled. Conversely to before the Receiver handles the following events:

- *on_start*: where variables are initialized
- *on_message*: where processes such as **decoding**, **Geonetworking extrapolation** takes place.

Thanks to the properties of the topic structure, the receiver will listen on:

```
1 IpAddress:5672/topic:rainbow.project.crf.polito.cam.*.*.*
```

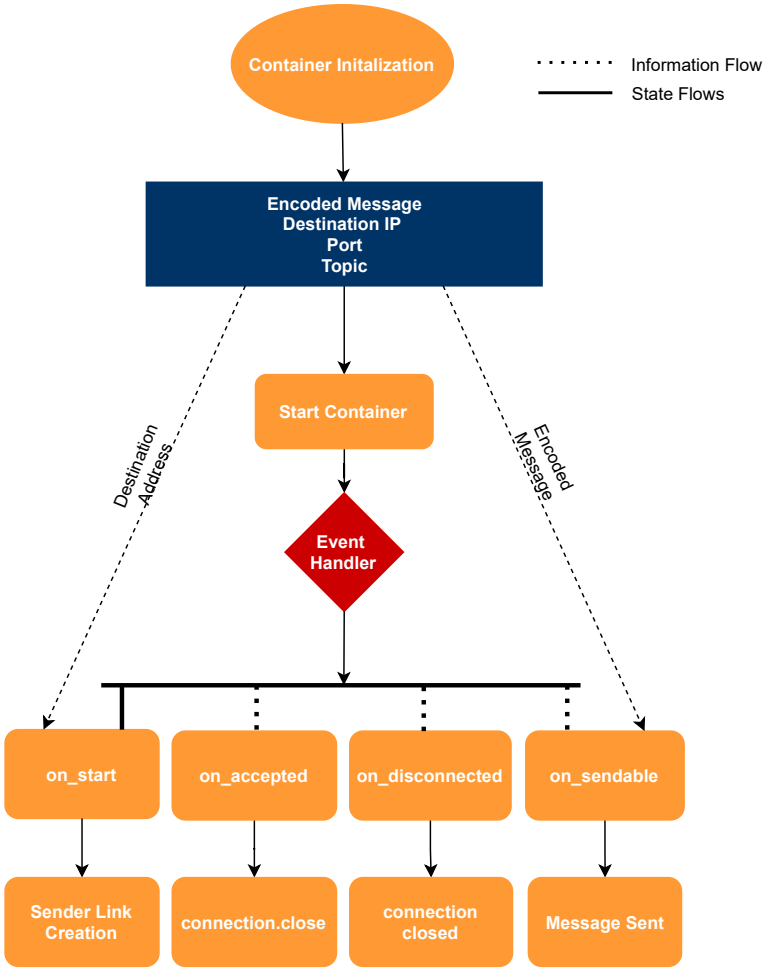

Figure 7.3: Qpid Proton Sender chart

This way, the Receiver consumes only messages that are associated with a particular direction of flow and belong to any subcategory within it.

From the Receiver the extrapolated messages are sent to the City Aggregator where they are processed to extract information relevant to the message type (CAM or DENM)

7.3 AMQP Broker

All message flows have the necessity of passing through a broker. AMQP Producers take advantages of those by sending messages over specified topics. On the other side, AMQP Receivers listen over specific topics to receive updates and critical information. It is therefore essential the use of a reliable and efficient broker

solution able to accommodate a sufficiently large traffic size without degrading in performance. Apache ActiveMQ, is an open source, multi-protocol, Java based message broker [35].

The way by which an ActiveMQ creates an application is through the following steps:

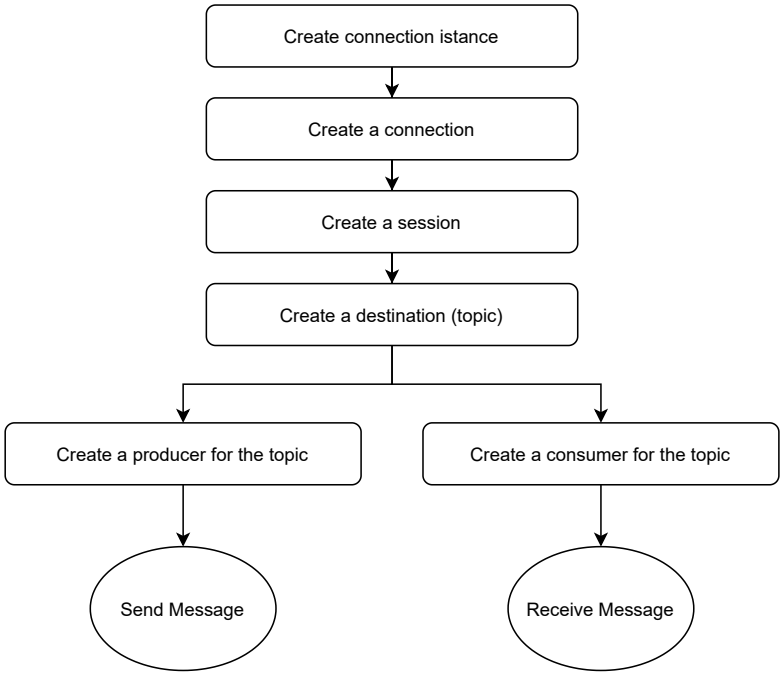

Figure 7.4: AMQP Broker initalization

For the current architecture **a single broker runs multiple instances** at the same time, this is done in order to reduce the overall complexity of the design. In a physical implementation several distributed brokers can concurrently work alongside to guarantee the correct functioning of the overall system. A specific broker takes care of the DENM flows from RSU/MEC nodes through the cloud interface whereas another one deals with the AMQP flows from and to the vehicles.

ActiveMQ performs much better than its competitors due to a low time for placement of messages over a topic. In addition it is far more efficient with lower number of clients [36].

7.4 Detailed message flow description

In order to complete the internal architecture of the system it is important to describe the **decoding phase** and how it interfaces with the City Aggregator.

When a message arrives at the AMQP Receiver side, it is routed depending on the information contained in the topic description. If a message contains simple DENMs notifications and it is coming from a RSU/MEC node, it is directly forwarded to the vehicles without any particular reprocessing. The bare message is forwarded to a successive topic that will be used by the vehicles in order to be informed regarding the zones of danger. The use of **direct forwarding** is done in an effort to reduce the latency time, thus **moving the decoding resources on-board of the vehicles**. In order to guarantee forwarding, a simple AMQP Sender is be employed; although different in its working principles, it maintains a similar structure to the basic AMQP Sender. Forwarding topic can be discerned from common ones by creating a **sub-topic category** for identification that points to the message final destination.

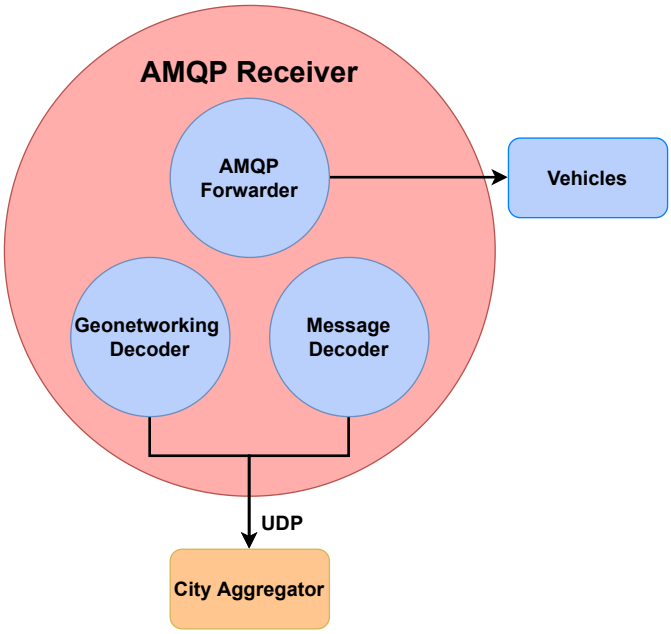

Figure 7.5: Forwarding processing inside AMQP Receiver

After the forwarding procedure has occurred, the message is firstly stripped of its BTP&Geonetworking part which is separately decoded. The remaining part of the message, containing the bare information, is firstly decoded using the inverse process shown in section 7.1, so as to convert a **bytearray into textual information**. Exploiting a **UDP Socket** messages are sent to the Map application system, ready for visualization.

In the case of vehicular messages (i.e **CAMs**) or citizen notification prompts, **no forwarding action takes place**. Firstly, the Rainbow architecture must validate user information prior to publishing. This is done in an effort to reduce

mischievous attempts that can severely undermine the correct functioning of the system. After validity cross-checks, User's messages may be included within the visualization platform and can be exploited by other vehicles.

Message flows	Message type	Forwarding	Interfaces
RSU/MEC to Vehicles	DENMs	Yes	AMQP Receiver & Forwarder
Vehicles to Cloud	CAMs	No	AMQP Receiver
Vulnerable User to Cloud Node	DENMs/CAMs	No	AMQP Receiver
Cloud node to City Aggregator	DENMs/CAMs	No	UDP Socket

Table 7.1: Detailed message flow within Cloud Node

7.5 Latency measurements

Although no explicit latency limit is specified in the ETSI standard, the **E-UTRAN must be able to transfer messages via 3GPP network between a UE and an application server with an end-to-end delay not superior to 1000 ms**. [37]. Considering recent works done with the use of 5G [38] in truck platooning, the latency measured from the Test User Equipment mounted on-vehicle board and the Base Station Equipment shows values around 2 ms of peak delay. This confirms the powerful nature of the technology which will be leveraged in this architecture.

Latencies measurements have been carried within the architecture by measuring the **time elapsed from the message generation, up until its reception and decoding at the vehicle side**. Latencies measured from the RSU/MEC/Vehicle node up until the decoding phase in the Cloud Node do not represent the limit case of the architecture, this is due to their easier construction and the presence of a single sender and receiver.

In the considered model, the end-to-end latency is, therefore the **union** of three type of delays:

- Time taken for the Message generation and AMQP sending procedure
- Time taken by the AMQP Receiver for consuming the message and forwarding it to the vehicle node
- Time taken for reception and decoding at the vehicle level

The sum of the three constitutes the total chain delays.

$$T_{\text{E2E}} = T_{\text{Emu}} + T_{\text{AMQP Send}} + T_{\text{AMQP Recv}} + T_{\text{AMQP Fwd}} + T_{\text{Vehicle Receive}} + T_{\text{Decoding}}$$

Where T_{Emu} represents the emulation time (encoding phase included), $T_{\text{AMQP Send}}$ is the AMQP Sender latency, $T_{\text{AMQP Recv}}$ is the Cloud Node AMQP Receiver latency, $T_{\text{AMQP Fwd}}$ is the AMQP forwarder latency, $T_{\text{Vehicle Receive}}$ is the AMQP Receiver latency mounted on board of vehicles and T_{Decoding} is the decoding time.

The preliminary work focuses mainly on the identification of the so called **latency bottleneck** within the architecture. This allows for a focused and efficient software optimization.

7.5.1 AMQP Sender/Receiver latency

Experimental results carried on the AMQP Qpid Proton Sender and Receiver measure the elapsed time from the container creation until the connection close procedure. This therefore include the **on_start phase**, the **on_sendable phase**, **on_accepted** and **on_disconnected** as described in section 7.2.1. The latency measured at the receiver side includes only the **on_message**. The receiver, in fact, is initialized only once and is kept alive for the entirety of the experiment.

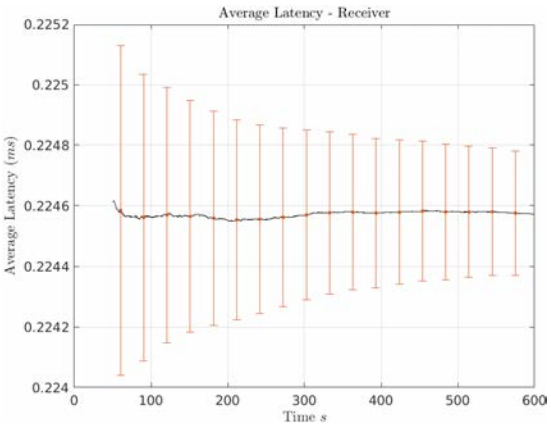

Figure 7.6: Average AMQP receiver latency

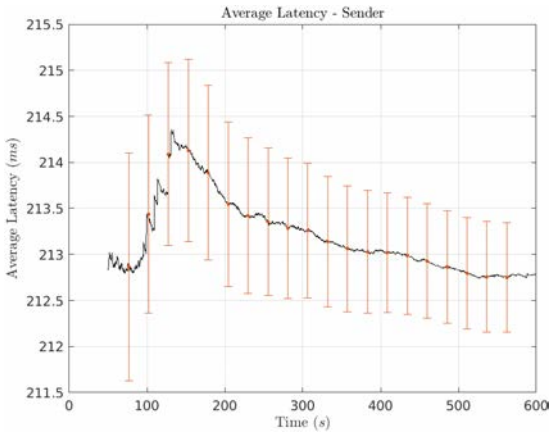

Figure 7.7: Average AMQP sender latency

Figures 7.6 and 7.7 are measures of **average latency** conducted on a set of data flow of 10 minutes. Data are plotted with the relative standard deviation. It is clear that the sending delay amounts for the majority of the taken time. This is justified by greater amount of procedures contained in the sending container compared to the receiving one. The AMQP receiver behaves particularly well and it is stable throughout the experiment.

Measurements conducted in this fashion tend to overestimate the actual latency; time retrieval within the container requires a set of n timestamps used

for comparison. This adds unwanted delay that undermines the correctness of the results.

7.5.2 Average transmission time CAM/DENM

The aforementioned latency measures all procedural times taken from the generation up until the sending. Similarly, the receiver considers only the time taken to process the incoming messages. **The time taken for the message to reach the broker and come back is not included**. For this reason a ping-based latency measurement experiment is devised. To simulate real packets transmission the data length is modified to accomodate for real CAM/DENMs packet sizes.

- CAMs: 99 bytes
- DENMs: 115 bytes

A series of a **thousands** packets is then sent across the channel towards the broker destination. This gives an estimate of the **time taken for the ITS packet to be transmitted and come back to its source**.

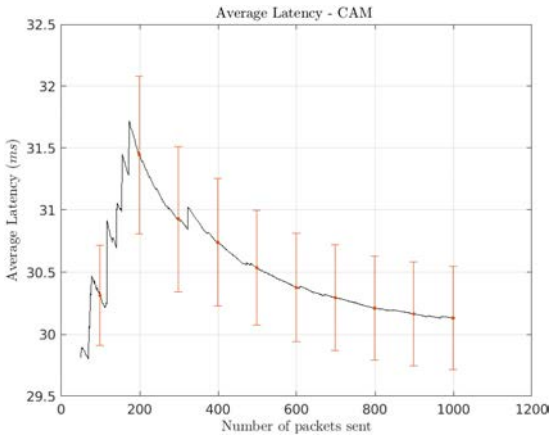

Figure 7.8: Average packet transmission latency (CAMs)

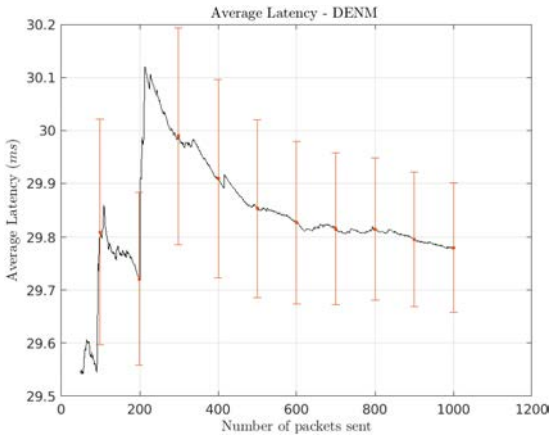

Figure 7.9: Average packet transmission latency (DENMs)

Figures **7.8** and **7.9** showcase similar results with average ping times that stabilize themselves around 30 ms. It is worth mentioning that standard utility ping instruments tend to **overestimate** latency measurements [39], the obtained results can then be considered as an **upper bound** for the overall transmission time.

7.5.3 End-to-end latency

In order to measure E2E latency in a **non-invasive** manner, the message structure is exploited. For CAMs **generationDeltatime** represents the time of reference position of the CAM, considered as the time for its generation [5]. It is considered wrapping a standard **TimestampITS** in a window of **65536**:

$$generationDeltaTime = \text{TimestampIts} \quad \% \quad 65536$$

The TimestampITS is an integer value considered since the epoch time `2004-01-01T00:00:00:000Z` [40]. For DENMs the **management container** holds information regarding the *actionID*, *eventPosition* and *detectionTime* as described in section 3.2.3. The latter can hold ITS-type timestamps [6].

Exploiting such container's sections, makes it possible to substitute these information with timestamp measurements obtained during the experimental phase. Upon reception, such results can be extrapolated during decoding and they can be **compared** to timestamps measured in that particular instance. This methods avoids the invasive comparing technique used in 7.5.1 providing accurate results.

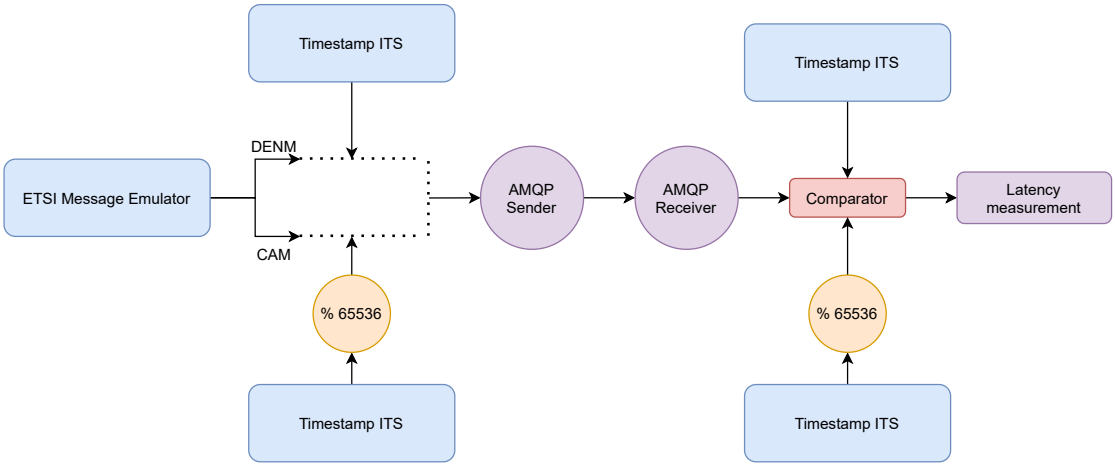

Figure 7.10: End-to-end latency working structure

Due to the great amount of data, **MySQL** database is used to store latency measurements computed with this method. **Grafana** is used for data visualization as it is a multi-platform interactive data aggregator that allows for compact and dynamic plotting [41].

Data are sent over to MySQL through **UDP socket** [**UDPSocket**], before data pre-processing. The MySQL listens over port 5005 commonly employed for **Real-time Transport Protocol** applications.

Data pre-processing consists in stripping valuable information from the message body and computing the **punctual latency**. In addition, an average latency value

is computed in a time window of 5 s; such a value **attenuates latency peaks** that can originate due to the oscillating performances of both hardware and wireless systems used during experimentation.

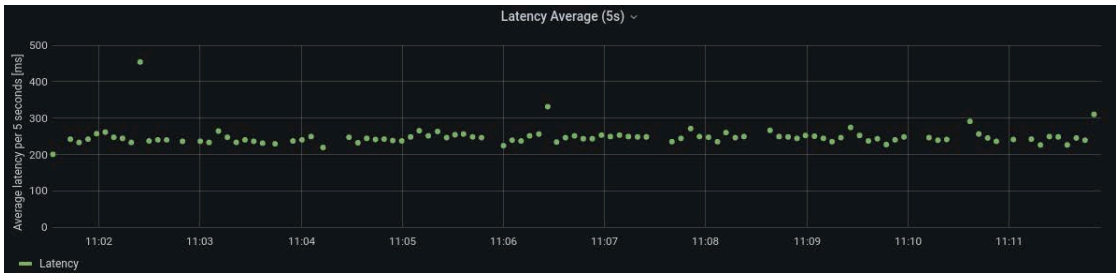

Figure 7.11: E2E average latency per 5s - CAMs flow

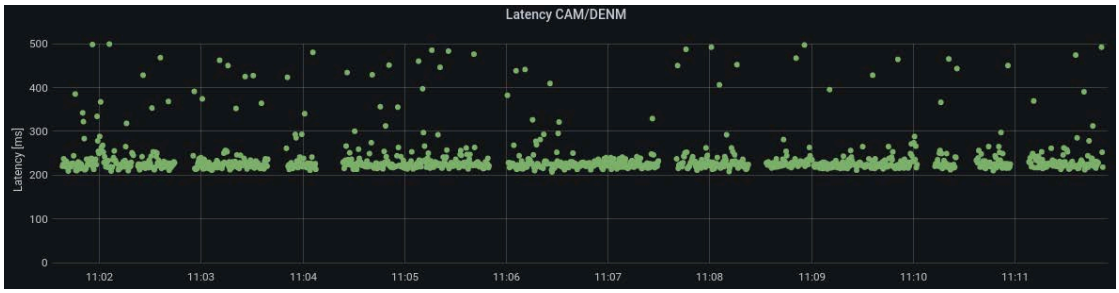

Figure 7.12: E2E punctual latency - CAMs flow

The *Cooperative Awareness Message* flow of figures **7.11**, **7.12**, shows an average latency around 200 ms with limited fluctuations. This is the **minimum latency** observed in the architecture as it does not require forwarding; it is directly processed within the Cloud Node.

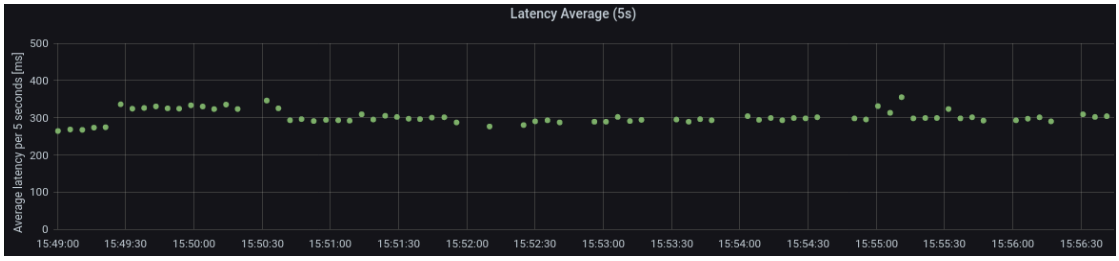

Figure 7.13: E2E average latency per 5s - DENMs flow

The average DENMs flow of figures 7.13, 7.14 highlights the effect of the AMQP forwarder. Due to the additional AMQP sender and AMQP receiver, the

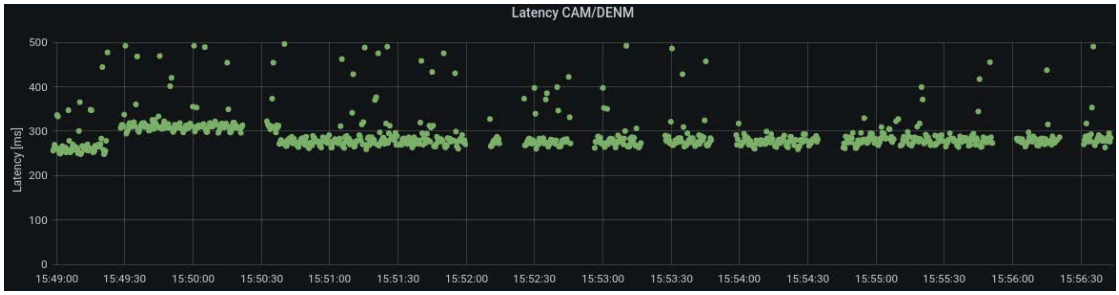

Figure 7.14: E2E punctual latency - DENMs flow

latency values revolve around 300 ms (100 ms more than the CAMs), this quantifies the average AMQP sending latency around 100 ms, different from the one obtained with invasive analysis in section 7.5.1.

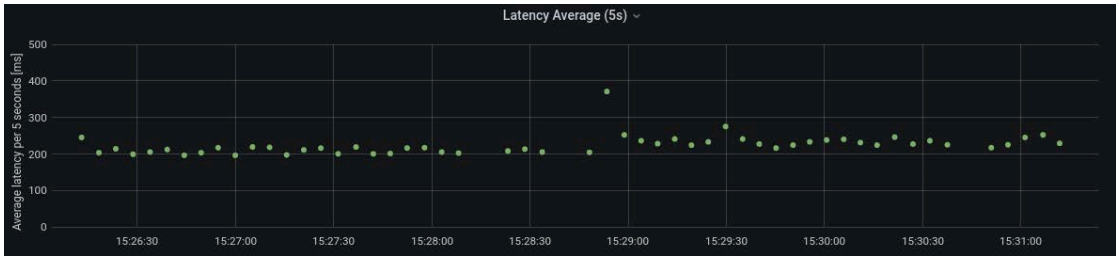

Figure 7.15: E2E average latency per 5s - CAMs/DENMs flow

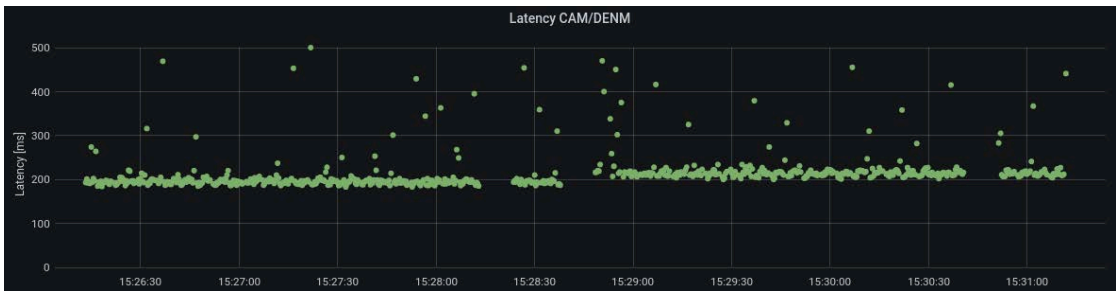

Figure 7.16: E2E punctual latency - CAMs/DENMs flow

The CAMs/DENMs mixed flow of figures **7.15**, **7.16** mimics a possible Citizen App scenario where users can send manual notifications. As evidenced in figure **7.15** the oscillating behavior is caused by the presence of a DENM message every five CAMs.

7.6 NTP: Latency measures for remote peers

It is important to remark that the obtained results are relevant only if timestamp encapsulation and extraction are performed on the **same machine**. Remote operations necessitate of a **machine-to-machine** synchronization rule. The literature offers an ample selection of flavors through which synchronization may occur across devices. *Precision Time Protocol* (**PTP**) defines a protocol for precise and accurate synchronization of real-time clocks of devices in a network of distributed systems. It does so enabling clocks of various inherent precision, resolution and stability to synchronize to a grandmaster clock [42]. *Reference Broadcast Synchronization* (**RBS**) employs nodes to send reference beacons to their neighbors using physical-layer broadcasts [43]. The most common an well-established synchronization system remains the *Network Time Protocol* (**NTP**) [44]. It has established itself as an Internet Standard Protocol by using a set of time servers and transmission paths as a synchronization subnet [45]. The precision of the NTP protocol heavily relies on the assumption of a **symmetric transmission** between client and server. If this does not hold, the synchronization may result erroneous [46].

NTP employs a stratified architecture level made up of servers. Each level, also known as **stratum**, is ordered in terms of its accuracy level, that is, the topmost level being the most accurate [45]. As the levels increase, the accuracy performances will degrade depending on the network path and local-clock stability [45].

A simple tools that can guarantee time synchronization through NTP is the *NTP Pool Project* [47]. Devised by Bjørn Hansen, it exploits a public pool of servers to achieve NTP synchronization worldwide. A domain name server is initialized, after which, multiple sub-domains are created under this address according to the geographical location [48].

```
server 0.it.pool.ntp.org
server 1.it.pool.ntp.org
server 2.it.pool.ntp.org
server 3.it.pool.ntp.org
```

As of today, **there exist no explicit method to synchronize two machines outside a common LAN**. As a matter of fact, remote computers fetch from a random pool of servers that changes every hour [47]. The choice usually tends to optimize three parameters:

- **Delay**: represents the time take for a request to reach and come back the server , this is used to calculated the network delays in the synchronization phase

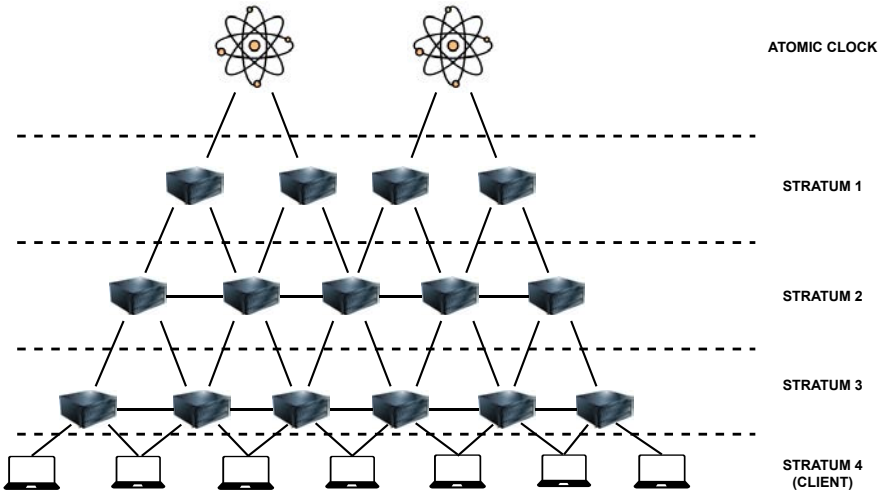

Figure 7.17: NTP stratum levels

- **Jitter**: Indicates to what extent does the dispersion affects the signal
- **Offset**: Which indicates the time difference of the local machine with respect to the external server.

The offset value depends on the specific server to which the machine is synchronized and on the machine itself.

Synchronization of two remote machines is only possible by forcing the remote machine to synchronize to the same server. Due to the intrinsic nature of the NTP protocol, this becomes extremely challenging. A server connection may be accurate for a specific machine and totally inaccurate for the other remote machine. In addition, connection's faults are frequent and unpredictable.

Synchronization is therefore considered as a **random error** with standard deviation σ affecting the overall latency. A set of offset data is retrieved from a specific NTP server connected to two remote machine. The overall NTP offset difference is calculated as the absolute difference between the two remote machines. Offset values can, in fact, be positive in the case of a machine delay with respect to the server and a negative value in the case of a machine offset advance.

$$NTP_{\text{off}} = |OFF_{\text{S1}} - OFF_{\text{S2}}|$$

As it is visible in 7.18 **positive skewness** properties can be evidences in the normal distribution graph. This characteristic describes the lack of symmetric of a normal distribution function around an average μ and standard deviation σ. As reported by *Doane & Seward* [49] skewness is defined through the second and third

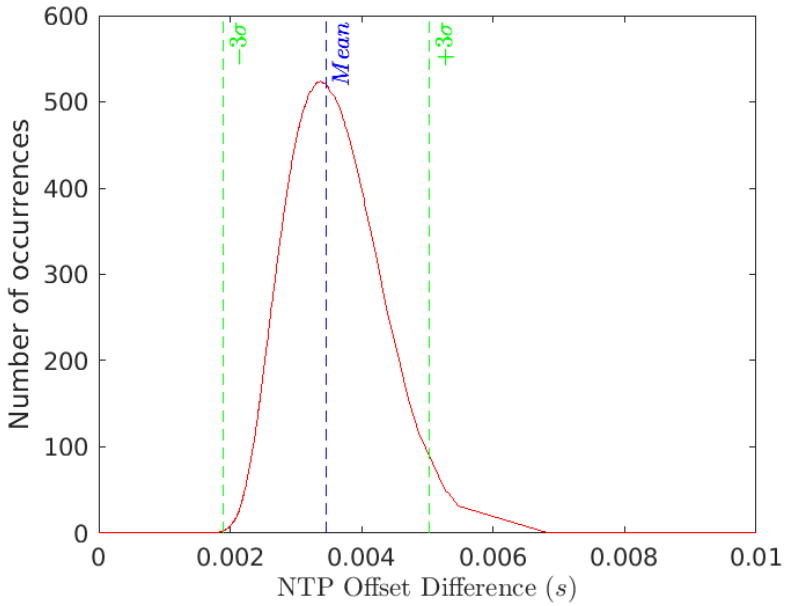

Figure 7.18: Standard distribution of offset between remote NTP clients

	Mean (s)	**Standard deviation (s)**
NTP Offset	0.0034664	0.0005226

Table 7.2: Mean and standard deviation for NTP offset measurements

moments around the mean

$$m_2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \qquad m_3 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3$$

According to *Fisher-Pearson* [50] the measure of skewness is obtained through:

$$g_1 = \frac{m_3}{m_2^{3/2}} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^{3/2}}$$

As mentioned above, synchronization to the same NTP server is difficult and unstable, this results in increasing number of offset measurements on both remote machine; the resulting effect is the positive skewness property. However, thanks to the obtained statistical results, it is possible to account for the direct effect on the E2E latency results of section 7.5.3

7.6.1 City Aggregator

The city aggregator completes the real-time notification system infrastructure; it introduces data conglomeration in order to visualize a live map publicly accessible. The platform is developed on **Mapbox**, a custom online map provider exploiting the Mapbox open source GL-JS JavaScript library [51].

The application is defined through JavaScript Express [52], used to create HTTP and HTTPS server connections. The connection is handled through IOSecureSocket. AMQP packet transmission occurs through the UDP socket interface. All messages entering the application are obtained from a single AMQP Receiver running on the server.

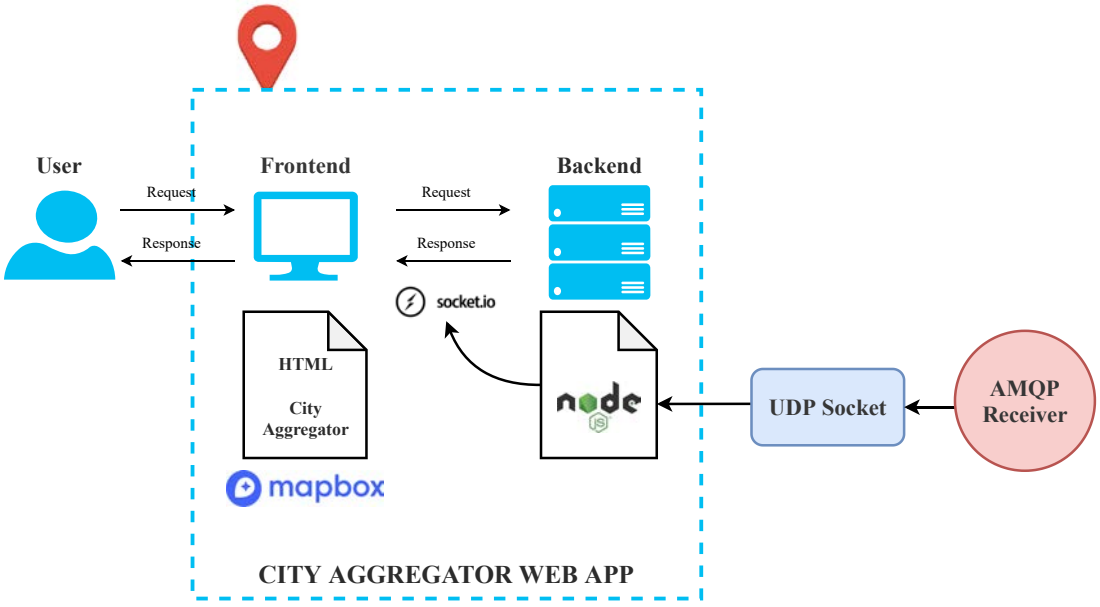

Figure 7.19: City Aggregator web app architecture

Messages received at the broker level are processed and sent to the server logic running node. Its aim is to read object information from the AMQP receiver and pass it to the client through **socket.io**. The frontend shows the relevant information on the map constructed with Mapbox whilst it implementing aggregating, filtering and data interpretation algorithms.

7.6.2 Geonetworking Geographical Area Definition

GeoNetworking is a network layer that provides packet routing in an ad hoc network [53]. Single-hop broadcast is mainly used in the transmission of periodic CAMs, whereas multi-hop distribution of events within an area is exploited in DENMs

[54].

Targeted areas are identified through geometrical shapes. ETSI defines a set of three different areas:

- circular area
- rectangular area
- elliptical area

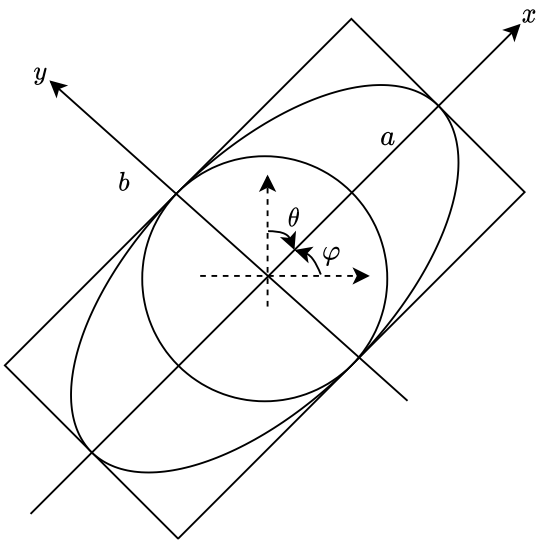

Figure 7.20: GeoNetworking geographical area definition

The value of a corresponds to the semi-axis of the ellipse and rectangle, b corresponds to the shorter semi-axis length (in the case of circular area b and a coincident with the radius r). The angle θ corresponds to the azimuth angle with respect to the major semi-axis, φ its complementary.

Areas are plotted onto the map by firstly drawing the shape through a series of interpolating point that respect the geometry shape. The figure is then rotate exploiting a simple rotation matrix around the z-axis:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The geometry is then converted into latitude and longitude coordinates with respect to a central point on the map. Considering the number of kilometers per degree of

longitude corresponds to:

$$\left(\frac{\pi R_{\text{earth}}}{180^\circ}\right)\cos\left(\frac{\theta\pi}{180^\circ}\right)$$

A displacement in meters around a point on the map corresponds in a latitude and longitude variation given by:

$$\text{Lat}_c - \text{Lat}_i = \frac{\Delta x}{R_{\text{earth}}} \cdot \left(\frac{180^\circ}{\pi}\right)$$

$$\text{Long}_c - \text{Long}_i = \frac{\Delta x}{R_{\text{earth}}} \cdot \left(\frac{180^\circ}{\pi}\right) \cdot \frac{1}{\cos(\text{Lat}_i \cdot \frac{\pi}{180^\circ})}$$

The calculations hold for small displacements, at least comparable to the earth's radius.

GN defines an additional parameter, useful for identifying the position of the object within the area. Function F determines whether a point P(x,y) is:

$$F(x,y)\begin{cases} = 1 & \text{for} \quad x = 0 \quad \text{and} \quad y = 0 (\text{centerpoint}) \\ > 0 & \text{inside geographical area} \\ = 0 & \text{at the border of the area} \\ < 0 & \text{outside the area} \end{cases}$$

For **circular** area F is defined as:

$$F(x,y) = 1 - \left(\frac{x}{r}\right)^2 - \left(\frac{y}{r}\right)^2$$

For **rectangular** areas the function F assumes the form:

$$F(x,y) = min\left(1 - \left(\frac{x}{a}\right)^2, 1 - \left(\frac{y}{b}\right)^2\right)$$

For **elliptical** area:

$$F(x,y) = 1 - \left(\frac{x}{a}\right)^2 - \left(\frac{y}{b}\right)^2$$

The results obtained in the City Aggregator are reported in the **appendix C**.

Chapter 8

Conclusions

The possibility of a real-time geo-referenced notification system brings the reality of a fully autonomous system, one step forward. The efforts made toward this goal are strengthened by the recent developments in the automotive and telecommunication fields. The Rainbow project is an example such effort made by governing agencies towards the enhancement of fog computing capabilities. This development will open up further researches, aiming at extending this platform to larger aspects of mobility.

The aim of this thesis has been that of designing the Cloud Node infrastructure and test its performance in terms of reliability and efficiency. A preliminary part has served the purpose of introducing the technological window where the project lays its foundations. C-V2X technologies and MEC infrastructures introduced the advantages of edge computing in terms of its reduced latency and greater scalability. The BSA opened up the description of *Cooperative Awareness Messages* and *Decentralized Environment Notification Messages*, crucially important in achieving hazard notification and full-scale dissemination. Detailed descriptions have been carried out regarding message transfer and flows. Furthermore, the key concepts of AMQP protocol have been exploited to manage a messaging infrastructure capable of rapidly moving and aggregating information that can instantaneously available in the palm of our hands.

The Rainbow Project has then been introduced, its underlying structure, basic functioning and use case scenarios have been described in-depth. Each architectural component has been analyzed, particularly in terms of its relation with the central Cloud Node.

The central part of the discussion has been devoted to the Cloud Node Framework, this includes an overall description of the message emulator algorithms, used for generation of ETSI messages. The structure of the AMQP Sender and Receiver have continued the discussion, particularly on the topic generation and

handling. The experimental part saw the description of the latency model and its role in the AMQP chain. Message transmission latency through variable data length ping has quantified the amount of latency measured between AMQP entities. Finally measurements of E2E latency have shown the overall message flow latency from its origination up to its decoding at the vehicle level. NTP considerations have been useful to quantify the average random error present in the E2E measures due to the lack of synchronization between remote peers.

The City Aggregator internal architecture has been exposed. Message information have been used exploiting the GN Geographical Area Definition to identify an area where hazards and vehicular communications should be delivered. The final demo has merely served the purpose of confirming the correct functioning of the architecture but has provided with a final visual reference.

Appendix A

ITS-PDU Header for CAMs

```
1   CAM-PDU-Descriptions {
2       itu-t (0) identified-organization (4) etsi (0) itsDomain (5)
    wg1 (1) en (302637) cam (2) version (2)
3   }
4
5   DEFINITIONS AUTOMATIC TAGS ::=
6
7   BEGIN
8
9   IMPORTS
10  ItsPduHeader, CauseCode, ReferencePosition, AccelerationControl,
    Curvature, CurvatureCalculationMode, Heading, LanePosition,
    EmergencyPriority, EmbarkationStatus, Speed, DriveDirection,
    LongitudinalAcceleration, LateralAcceleration,
    VerticalAcceleration, StationType, ExteriorLights,
    DangerousGoodsBasic, SpecialTransportType, LightBarSirenInUse,
    VehicleRole, VehicleLength, VehicleWidth, PathHistory,
    RoadworksSubCauseCode, ClosedLanes, TrafficRule, SpeedLimit,
    SteeringWheelAngle, PerformanceClass, YawRate,
    ProtectedCommunicationZone, PtActivation, Latitude, Longitude,
    ProtectedCommunicationZonesRSU, CenDsrcTollingZone FROM ITS-
    Container {
11      itu-t (0) identified-organization (4) etsi (0) itsDomain (5)
    wg1 (1) ts (102894) cdd (2) version (2)
12  };
13
14
15  -- The root data frame for cooperative awareness messages
16
17  CAM ::= SEQUENCE {
18      header ItsPduHeader,
```

```
19          cam CoopAwareness
20      }
21
22      CoopAwareness ::= SEQUENCE {
23          generationDeltaTime GenerationDeltaTime,
24          camParameters CamParameters
25      }
26
27      CamParameters ::= SEQUENCE {
28          basicContainer BasicContainer,
29          highFrequencyContainer HighFrequencyContainer,
30          lowFrequencyContainer LowFrequencyContainer OPTIONAL,
31          specialVehicleContainer SpecialVehicleContainer OPTIONAL,
32          ...
33      }
34
35      HighFrequencyContainer ::= CHOICE {
36          basicVehicleContainerHighFrequency
        BasicVehicleContainerHighFrequency,
37          rsuContainerHighFrequency RSUContainerHighFrequency,
38          ...
39      }
40
41      LowFrequencyContainer ::= CHOICE {
42          basicVehicleContainerLowFrequency
        BasicVehicleContainerLowFrequency,
43          ...
44      }
45
46      SpecialVehicleContainer ::= CHOICE {
47          publicTransportContainer PublicTransportContainer,
48          specialTransportContainer SpecialTransportContainer,
49          dangerousGoodsContainer DangerousGoodsContainer,
50          roadWorksContainerBasic RoadWorksContainerBasic,
51          rescueContainer RescueContainer,
52          emergencyContainer EmergencyContainer,
53          safetyCarContainer SafetyCarContainer,
54          ...
55      }
56
57      BasicContainer ::= SEQUENCE {
58          stationType StationType,
59          referencePosition ReferencePosition,
60          ...
61      }
62
63      BasicVehicleContainerHighFrequency ::= SEQUENCE {
64          heading Heading,
65          speed Speed,
```

```
66          driveDirection DriveDirection,
67          vehicleLength VehicleLength,
68          vehicleWidth VehicleWidth,
69          longitudinalAcceleration LongitudinalAcceleration,
70          curvature Curvature,
71          curvatureCalculationMode CurvatureCalculationMode,
72          yawRate YawRate,
73          accelerationControl AccelerationControl OPTIONAL,
74          lanePosition LanePosition OPTIONAL,
75          steeringWheelAngle SteeringWheelAngle OPTIONAL,
76          lateralAcceleration LateralAcceleration OPTIONAL,
77          verticalAcceleration VerticalAcceleration OPTIONAL,
78          performanceClass PerformanceClass OPTIONAL,
79          cenDsrcTollingZone CenDsrcTollingZone OPTIONAL
80      }
81
82      BasicVehicleContainerLowFrequency  ::= SEQUENCE {
83          vehicleRole VehicleRole,
84          exteriorLights ExteriorLights,
85          pathHistory PathHistory
86      }
87
88      PublicTransportContainer ::= SEQUENCE {
89          embarkationStatus EmbarkationStatus,
90          ptActivation PtActivation OPTIONAL
91      }
92
93      SpecialTransportContainer ::= SEQUENCE {
94          specialTransportType SpecialTransportType,
95          lightBarSirenInUse LightBarSirenInUse
96      }
97
98      DangerousGoodsContainer ::= SEQUENCE {
99          dangerousGoodsBasic DangerousGoodsBasic
100     }
101
102     RoadWorksContainerBasic ::= SEQUENCE {
103         roadworksSubCauseCode RoadworksSubCauseCode OPTIONAL,
104         lightBarSirenInUse LightBarSirenInUse,
105         closedLanes ClosedLanes OPTIONAL
106     }
107
108     RescueContainer ::= SEQUENCE {
109         lightBarSirenInUse LightBarSirenInUse
110     }
111
112     EmergencyContainer ::= SEQUENCE {
113         lightBarSirenInUse LightBarSirenInUse,
114         incidentIndication CauseCode OPTIONAL,
```

```
115         emergencyPriority EmergencyPriority OPTIONAL
116     }
117
118     SafetyCarContainer ::= SEQUENCE {
119         lightBarSirenInUse LightBarSirenInUse,
120         incidentIndication CauseCode OPTIONAL,
121         trafficRule TrafficRule OPTIONAL,
122         speedLimit SpeedLimit OPTIONAL
123     }
124
125     RSUContainerHighFrequency ::= SEQUENCE {
126         protectedCommunicationZonesRSU ProtectedCommunicationZonesRSU
    OPTIONAL,
127         ...
128     }
129
130     GenerationDeltaTime ::= INTEGER { oneMilliSec(1) } (0..65535)
131
132     END
```

Appendix B

AMQP WireShark capture

The capture reported in figure B.1 represents the different phases of an AMQP message. This is complementary to the description of chapter 5. The transmission is initiated with the **Protocol-Header 1-0-0** and **Protocol-Header 1-0-0 sasl.mechanism**, followed by **sasl.init and outcome**. The beginning of the flow is performed by the **open begin attach flow** followed by the **transfer** and **disposition**. Finally when the transmission is ended (either from the client or the broker) the **close** procedure occurs.

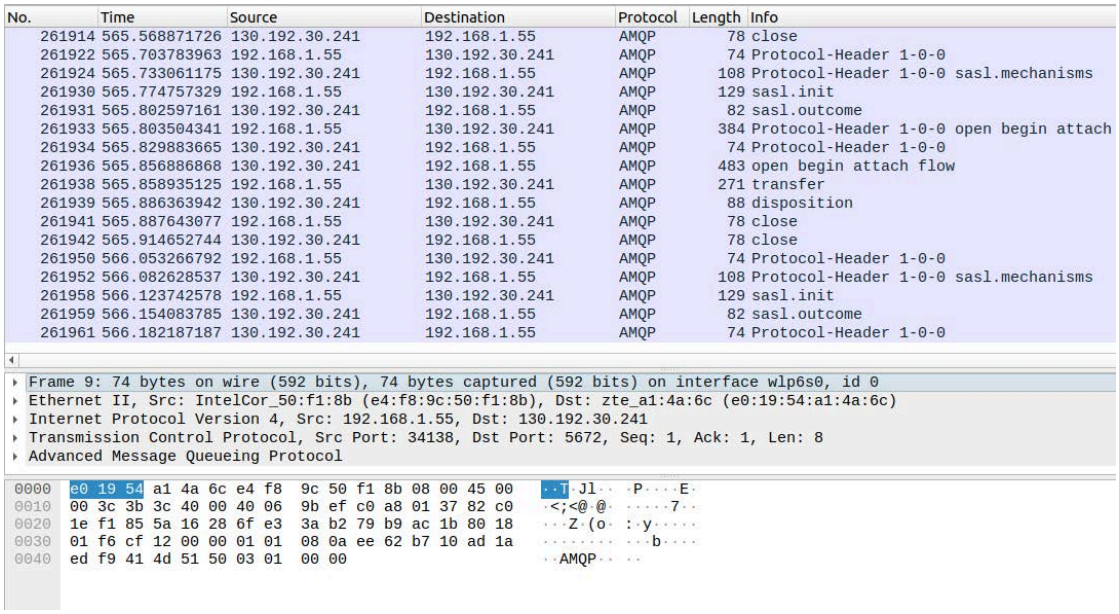

Figure B.1: WireShark capture of AMQP message flows

Appendix C

City Aggregator - DEMO

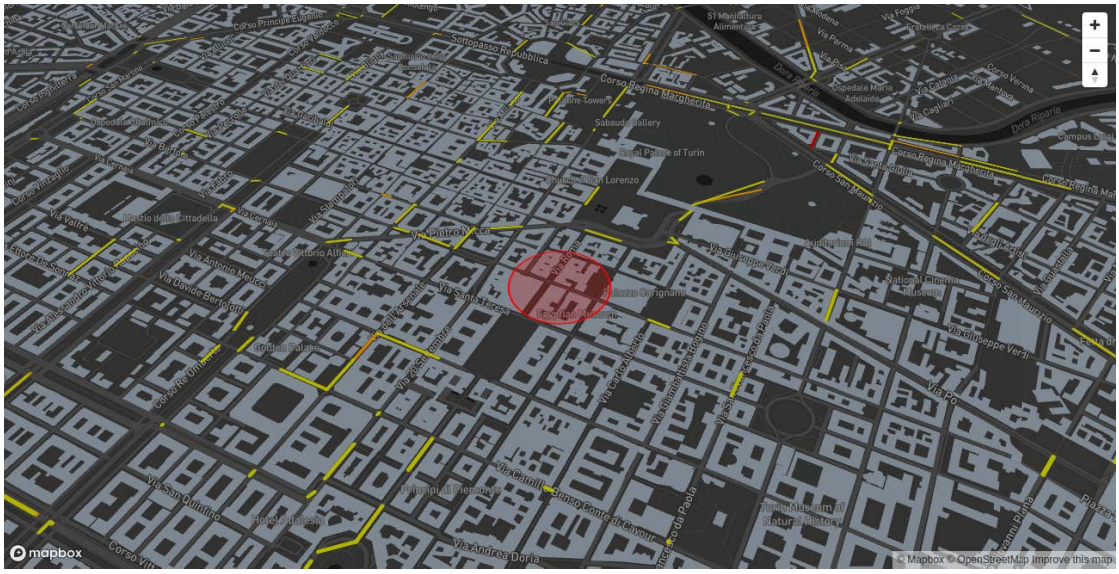

Figure C.1: Geometrical Area Definition Geonetworking - Circle

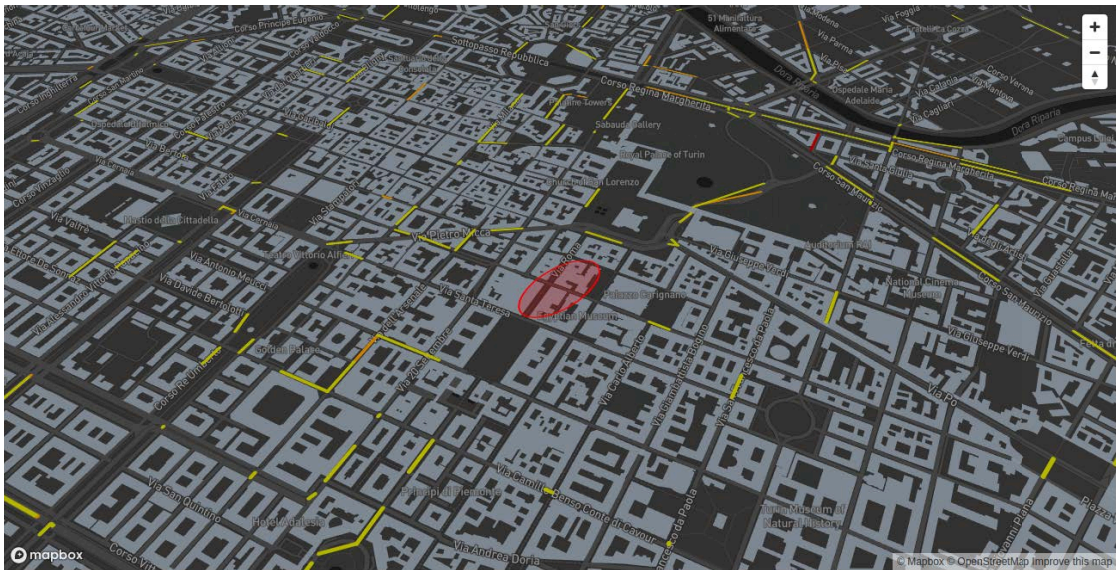

Figure C.2: Geometrical Area Definition Geonetworking - Ellipse

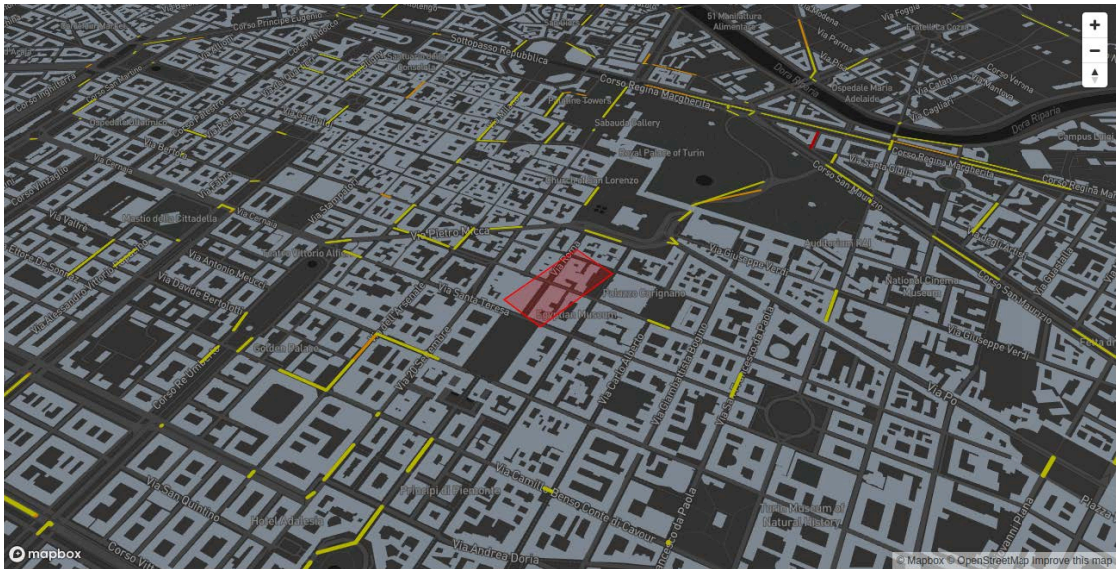

Figure C.3: Geometrical Area Definition Geonetworking - Rectangle

Bibliography

[1] Valerian Mannoni, Vincent Berg, Stefania Sesia, and Erica Perraud. «A Comparison of the V2X Communication Systems: ITS-G5 and C-V2X». In: *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*. 2019, pp. 1–5. DOI: 10.1109/VTCSpring.2019.8746562 (cit. on pp. 2, 3).

[2] Taajwar Bey and Girma Tewolde. «Evaluation of DSRC and LTE for V2X». In: *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. 2019, pp. 1032–1035. DOI: 10.1109/CCWC.2019.8666563 (cit. on p. 2).

[3] Lili Miao, John Jethro Virtusio, and Kai-Lung Hua. «PC5-Based Cellular-V2X Evolution and Deployment». In: *Sensors* 21.3 (2021). ISSN: 1424-8220. DOI: 10.3390/s21030843. URL: https://www.mdpi.com/1424-8220/21/3/843 (cit. on p. 5).

[4] ETSI. *Universal Mobile Telecommunications System (UMTS);LTE; Proximity-based services (ProSe); Stage 2*. Technical Specification (TS) 123.303. Version 14.1.0. European Telecommunications Standards Institute (ETSI), May 2017. URL: https://www.etsi.org/deliver/etsi_ts/123300_123399/123303/14.01.00_60/ts_123303v140100p.pdf (cit. on pp. 6–8).

[5] ETSI. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. Tech. rep. 302 637-2. Version 1.4.1. European Telecommunications Standards Institute (ETSI) (cit. on pp. 9–11, 55).

[6] ETSI. *Intelligent Transport Systems (ITS);Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*. Tech. rep. 302 637-3. Version 1.2.2. European Telecommunications Standards Institute (ETSI) (cit. on pp. 9, 11–14, 55).

[7] ETSI. *Multi-access Edge Computing (MEC); Framework and Reference Architecture*. Group Specification (GS) 003. Version 2.2.1. European Telecommunications Standards Institute (ETSI), Dec. 2020. URL: https://www.etsi.org/

deliver/etsi_gs/MEC/001_099/003/02.02.01_60/gs_MEC003v020201p.pdf (cit. on pp. 15–17).

[8] ETSI. *Multi-access Edge Computing(MEC); Study on MEC Support for V2X Use Cases*. Group Report (GR) 022. Version 2.1.1. European Telecommunications Standards Institute (ETSI), Sept. 2018. URL: https://www.etsi.org/deliver/etsi_gr/MEC/001_099/022/02.01.01_60/gr_MEC022v020101p.pdf (cit. on pp. 17, 18).

[9] Marco Malinverno, Giuseppe Avino, Claudio Casetti, Carla Fabiana Chiasserini, Francesco Malandrino, and Salvatore Scarpina. «Edge-Based Collision Avoidance for Vehicles and Vulnerable Users: An Architecture Based on MEC». In: *IEEE Vehicular Technology Magazine* 15.1 (2020), pp. 27–35. DOI: 10.1109/MVT.2019.2953770 (cit. on p. 18).

[10] Mustafa Emara, Miltiades C. Filippou, and Dario Sabella. «MEC-Assisted End-to-End Latency Evaluations for C-V2X Communications». In: *2018 European Conference on Networks and Communications (EuCNC)*. 2018, pp. 1–9. DOI: 10.1109/EuCNC.2018.8442825 (cit. on p. 19).

[11] Nitin Naik. «Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP». In: *2017 IEEE International Systems Engineering Symposium (ISSE)*. 2017, pp. 1–7. DOI: 10.1109/SysEng.2017.8088251 (cit. on p. 20).

[12] *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0*. 2012 (cit. on pp. 20–29).

[13] Ben Zhang, Nitesh Mor, John Kolb, Douglas S. Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. «The Cloud is Not Enough: Saving IoT from the Cloud». In: *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*. Santa Clara, CA: USENIX Association, July 2015. URL: https://www.usenix.org/conference/hotcloud15/workshop-program/presentation/zhang (cit. on p. 30).

[14] Weisong Shi and Schahram Dustdar. «The Promise of Edge Computing». In: *Computer* 49.5 (2016), pp. 78–81. DOI: 10.1109/MC.2016.145 (cit. on p. 30).

[15] Koustabh Dolui and Soumya Kanti Datta. «Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing». In: *2017 Global Internet of Things Summit (GIoTS)*. 2017, pp. 1–6. DOI: 10.1109/GIOTS.2017.8016213 (cit. on p. 30).

[16] Tuan Nguyen Gia, Amir M. Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. «Fog Computing Approach for Mobility Support in Internet-of-Things Systems». In: *IEEE Access* 6 (2018), pp. 36064–36082. DOI: 10.1109/ACCESS.2018.2848119 (cit. on p. 30).

[17] Luisa Andreone Marco Marchetti. *D1.3 Use-Cases Descriptions*. Project Deliverable. Version V 1.0. URL: https://rainbow-h2020.eu/wp-content/uploads/2021/02/RAINBOW_D1.3-Use-Cases-Descriptions-Version-1.0.pdf (cit. on pp. 31, 32, 34, 37, 39–41).

[18] Zain Ashi, Mohammad Al-Fawa'reh, and Mustafa Al-Fayoumi. «Fog computing: security challenges and countermeasures». In: *International Journal of Computer Applications* 175 (Aug. 2020), pp. 30–36. DOI: 10.5120/ijca2020920648 (cit. on p. 33).

[19] Bo Tang, Zhen Chen, Gerald Hefferman, Shuyi Pei, Tao Wei, Haibo He, and Qing Yang. «Incorporating Intelligence in Fog Computing for Big Data Analysis in Smart Cities». In: *IEEE Transactions on Industrial Informatics* PP (Feb. 2017). DOI: 10.1109/TII.2017.2679740 (cit. on p. 33).

[20] «On the Use of Cameras for the Detection of CriticalEvents in Sensors-Based Emergency Alerting Systems». In: *Journal of Sensor and Actuator Networks* () (cit. on p. 34).

[21] «Wide area video surveillane based on edge and fog computing concept». In: *2017 8th International Conference on Information, Intelligence, Systems Applications (IISA)*. 2017, pp. 1–6. DOI: 10.1109/IISA.2017.8316451 (cit. on p. 34).

[22] «Fog Computing Based Intelligent Security Surveillance Using PTZ Controller Camera». In: *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2019, pp. 1–5. DOI: 10.1109/ICCCNT45670.2019.8944815 (cit. on p. 34).

[23] «Experimental evaluation of CAM and DENM messaging services in vehicular communications». In: *Transportation Research Part C: Emerging Technologies* 46 (Sept. 2014), 98–120. DOI: 10.1016/j.trc.2014.05.006 (cit. on p. 36).

[24] F. Berens V. Martinez. *Survey on ITS-G5 CAM statistics*. Technical Report TR2052. Version 1.0.1. CAR 2 CAR (cit. on p. 36).

[25] Hien Van, Frédéric Tran, and Jean-Marc Menaud. «Performance and Power Management for Cloud Infrastructures». In: *2012 IEEE Fifth International Conference on Cloud Computing* 0 (July 2010), pp. 329–336. DOI: 10.1109/CLOUD.2010.25 (cit. on p. 39).

[26] Atul Adya et al. «Slicer: Auto-sharding for datacenter applications». In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 739–753 (cit. on p. 40).

[27] Marco Serafini, Essam Mansour, Ashraf Aboulnaga, Kenneth Salem, Taha Rafiq, and Umar Farooq Minhas. «Accordion: Elastic scalability for database systems supporting distributed transactions». In: *Proceedings of the VLDB Endowment* 7.12 (2014), pp. 1035–1046 (cit. on p. 40).

[28] Demetris Trihinas. *D4.1 Data Management Services*. Project Deliverable. Version V 1.0. URL: https://rainbow-h2020.eu/wp-content/uploads/2021/02/RAINBOW_D4.1-Data-Management-Services-Early-Release_V1.0.pdf (cit. on p. 40).

[29] ETSI. *Electronic Signatures and Infrastructures (ESI); Policy and security requirements for Trust Service Providers issuing certificates*. European Standard (ES) 319 411-1. Version 1.2.1. European Telecommunications Standards Institute (ETSI), Feb. 2018. URL: https://www.etsi.org/deliver/etsi_en/319400_319499/31941101/01.02.01_30/en_31941101v010201v.pdf (cit. on p. 43).

[30] Badis Hammi, Jean Philippe Monteuuis, Eduardo Salles Daniel, and Houda Labiod. «ASN.1 Specification for ETSI Certificates and Encoding Performance Study». In: *2017 18th IEEE International Conference on Mobile Data Management (MDM)*. 2017, pp. 291–298. DOI: 10.1109/MDM.2017.47 (cit. on pp. 43, 44).

[31] Xiaolin Li, Xiaoqiang Wen, and Xuesong Li. «Design and application of the ASN.1 module in the LTE-Uu interface protocol stack». In: *2010 International Conference on Educational and Information Technology*. Vol. 3. 2010, pp. V3–149–V3–152. DOI: 10.1109/ICEIT.2010.5608403 (cit. on p. 44).

[32] Erik Moqvist. *asn1tools*. https://github.com/eerimoq/asn1tools. Version 0.158.0. 2017 (cit. on p. 45).

[33] ETSI. *Intelligent Transport Systems (ITS);Vehicular Communications;GeoNetworking;Part 3: Network architecture*. Technical Specification (TS) 102 636-3. Version 1.1.1. European Telecommunications Standards Institute (ETSI), Mar. 2010. URL: https://www.etsi.org/deliver/etsi_ts/102600_102699/10263603/01.01.01_60/ts_10263603v010101p.pdf (cit. on p. 45).

[34] *Apache Qpid Proton*. https://qpid.apache.org/proton/index.html. Accessed: 2021-06-28 (cit. on pp. 46, 47).

[35] *Apache ActiveMQ*. https://activemq.apache.org/. Accessed: 2021-06-29 (cit. on p. 50).

[36] Valeriu Manuel Ionescu. «The analysis of the performance of RabbitMQ and ActiveMQ». In: *2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER)*. 2015, pp. 132–137. DOI: 10.1109/RoEduNet.2015.7311982 (cit. on p. 50).

[37] ETSI. *ETSI TS 122 185 V14.3.0 (2017-03)LTE;Service requirements for V2X services (3GPP TS 22.185 version 14.3.0 Release 14)*. Technical Specification (TS) 122 185. Version 14.3.0. European Telecommunications Standards Institute (ETSI), Mar. 2017. URL: https://www.etsi.org/deliver/etsi_ts/122100_122199/122185/14.03.00_60/ts_122185v140300p.pdf (cit. on p. 52).

[38] Kohei Moto, Manabu Mikami, Koichi Serizawa, and Hitoshi Yoshino. «Field Experimental Evaluation on 5G V2N Low Latency Communication for Application to Truck Platooning». In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. 2019, pp. 1–5. DOI: 10.1109/VTCFall.2019.8891450 (cit. on p. 52).

[39] Gary Jackson, Pete Keleher, and Alan Sussman. «A Ping Too Far: Real World Network Latency Measurement». In: *2015 IEEE 11th International Conference on e-Science*. 2015, pp. 580–588. DOI: 10.1109/eScience.2015.74 (cit. on p. 54).

[40] ETSI. *ETSI TS 102 894-2 V1.2.1 (2014-09)Intelligent Transport Systems (ITS);Users and applications requirements; Part 2: Applications and facilities layer common data dictionary*. Technical Specification (TS) 102 894-2. Version 1.2.1. European Telecommunications Standards Institute (ETSI), Sept. 2014. URL: https://www.etsi.org/deliver/etsi_ts/102800_102899/10289402/01.02.01_60/ts_10289402v010201p.pdf (cit. on p. 55).

[41] *Grafana*. https://grafana.com/. Accessed: 2021-06-30 (cit. on p. 55).

[42] «IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems». In: *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)* (2020), pp. 1–499. DOI: 10.1109/IEEESTD.2020.9120376 (cit. on p. 58).

[43] Jeremy Elson, Lewis Girod, and Deborah Estrin. «Fine-grained network time synchronization using reference broadcasts». In: *ACM SIGOPS Operating Systems Review* 36.SI (2002), pp. 147–163 (cit. on p. 58).

[44] David L Mills et al. «Network time protocol (NTP)». In: (1985) (cit. on p. 58).

[45] D.L. Mills. «Internet time synchronization: the network time protocol». In: *IEEE Transactions on Communications* 39.10 (1991), pp. 1482–1493. DOI: 10.1109/26.103043 (cit. on p. 58).

[46] Faten Mkacher and Andrzej Duda. «Calibrating NTP». In: *2019 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. 2019, pp. 1–6. DOI: 10.1109/ISPCS.2019.8886646 (cit. on p. 58).

[47] *NTP Pool Project*. https://www.ntppool.org/en/. Accessed: 2021-07-01 (cit. on p. 58).

[48] Chien-Chi Chao, Shih-Ping Huang, Wu-Shun Jwo, and et al. «Development of the NTP Pool Project in Taiwan». In: (Jan. 2009) (cit. on p. 58).

[49] David P. Doane and Lori E. Seward. «Measuring Skewness: A Forgotten Statistic?» In: *Journal of Statistics Education* 19.2 (2011), null. DOI: 10.1080/10691898.2011.11889611. eprint: https://doi.org/10.1080/10691898.2011.11889611. URL: https://doi.org/10.1080/10691898.2011.11889611 (cit. on p. 59).

[50] Karl Pearson. «Contributions to the mathematical theory of evolution». In: *Philosophical Transactions of the Royal Society of London. A* 185 (1894), pp. 71–110 (cit. on p. 60).

[51] *Mapbox*. https://www.mapbox.com/. Accessed: 2021-07-01 (cit. on p. 61).

[52] *Express*. https://expressjs.com/. Accessed: 2021-07-01 (cit. on p. 61).

[53] ETSI. *ntelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality*. European Standard (ES) 302 636-4-1. Version 1.2.1. European Telecommunications Standards Institute (ETSI), May 2014. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/3026360401/01.02.01_30/en_3026360401v010201v.pdf (cit. on p. 61).

[54] Sebastian Kuhlmorgen, Ignacio Llatser, Andreas Festag, and Gerhard Fettweis. «Performance Evaluation of ETSI GeoNetworking for Vehicular Ad Hoc Networks». In: *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. 2015, pp. 1–6. DOI: 10.1109/VTCSpring.2015.7146003 (cit. on p. 62).