



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Tècnica Superior d'Enginyeria  
de Telecomunicació de Barcelona



# Performance analysis of V2X technologies 802.11p and LTE-PC5

---

Master Thesis  
submitted to the Faculty of the  
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona  
Universitat Politècnica de Catalunya  
by  
Adrià Pons Serra

In partial fulfillment  
of the requirements for the master in  
**MASTER'S DEGREE IN ADVANCED TELECOMMUNICATION  
TECHNOLOGIES ENGINEERING**

Advisor: Jordi Casademont Serra  
Barcelona, January 2023

# Contents

<b>List of Figures</b>	<b>3</b>
<b>List of Tables</b>	<b>4</b>
<b>1 Introduction</b>	<b>11</b>
1.1 Scope . . . . .	11
1.2 Work Plan . . . . .	12
1.3 Budget . . . . .	13
<b>2 State of the art</b>	<b>14</b>
2.1 Vehicle-to-Everything Communications (V2X) . . . . .	14
2.2 V2X Protocol Architecture . . . . .	16
2.3 IEEE 802.11p . . . . .	20
2.4 LTE-PC5 . . . . .	21
<b>3 Deployment of simulation environments and results</b>	<b>28</b>
3.1 Simulation environments . . . . .	28
3.1.1 SUMO . . . . .	28
3.1.2 OMNeT++ . . . . .	31
3.2 Creation of scenarios . . . . .	34
3.2.1 Collisions scenario . . . . .	34
3.2.2 OMNeT++ scenario . . . . .	39
3.3 Results . . . . .	45
3.3.1 Collisions scenario . . . . .	45
3.3.2 OMNeT++ scenario . . . . .	48
<b>4 Performance Evaluation. 802.11p and LTE-PC5</b>	<b>51</b>
4.1 Highway scenario . . . . .	51
4.2 Simulation Parameters . . . . .	52
4.2.1 Standard Configuration . . . . .	52
4.3 Results . . . . .	54
4.3.1 Packet Error Rate . . . . .	55
4.3.2 Distance errors . . . . .	59
4.3.3 Neighborhood Awareness Ratio . . . . .	63
<b>5 Conclusions and future development:</b>	<b>66</b>
<b>References</b>	<b>68</b>

## List of Figures

1	Types of Vehicle-to-Everything communications. . . . .	12
2	Types of Vehicle-to-Everything communications. . . . .	15
3	ITS reference architecture. . . . .	16
4	GeoNetworking routing schemes. . . . .	19
5	V2X communication interfaces. . . . .	22
6	V2X communication modes. . . . .	23
7	Structure of the LTE-PC5 resource grid with 10 subchannels per subframe. . . . .	24
8	LTE subchannelization schemes. . . . .	25
9	Semi-Persisten Scheduling in C-V2X. Sensing and Selection Windows. . . . .	27
10	Netedit's visual interface. . . . .	29
11	CARLA and SUMO co-simulation. . . . .	31
12	Simulation framework overview. . . . .	32
13	Artery simulation framework. . . . .	32
14	Map view of Aragó with Passeig de Gràcia streets. . . . .	34
15	OpenStreetMap Web Wizard tool. . . . .	35
16	Junction with prohibited lane changes. . . . .	35
17	SUMO simulation of the collision scenario. . . . .	39
18	Netedit view of Town05 scenario. . . . .	40
19	Netedit interface to define new routes. . . . .	41
20	Scenario Town05 running. . . . .	42
21	ROS scheme of the nodes and topics involved in the transmission of CAM messages. Example with 2 vehicles. . . . .	44
22	Example of CAM message in JSON format. . . . .	45
23	Simulation results. Example of the collisions.xml and the positions.csv files. . . . .	47
24	Simulation results. A csv file with the actualized LDM from all the vehicles on the simulation. . . . .	50
25	5 km highway scenario. . . . .	51
26	PER results with standard configurations and 2 vehicles. . . . .	56
27	PER results with modified configurations and 2 vehicles. . . . .	56
28	PER results. . . . .	58
29	Error distance perceived by vehicle 1 before receiving the 2n CAM from vehicle 2. . . . .	60
30	Distance error results. . . . .	61
31	NAR results. . . . .	64

## List of Tables

1	Mapping between ITS and OSI architectures. . . . .	17
2	ITS-G5 Traffic priority classes . . . . .	21
3	Types of vehicles used in the SUMO scenario. . . . .	37
4	IEEE 802.11p, OMNeT++ simulation parameters. . . . .	43
5	List of parameters in "collisions.xml". . . . .	46
6	List of parameters in "positions.csv". . . . .	47
7	Personalized parameters of the LDM. . . . .	49
8	OMNeT++ configuration parameters for IEEE 802.11p standard. . . . .	52
9	OMNeT++ configuration parameters for C-V2X LTE Mode 4 standard. . . . .	54
10	Coverage modifications to compare 802.11p and LTE PC5 technologies. . . . .	57

---

## Abbreviations

<b>3GPP</b>	3rd Generation Partnership Project
<b>AC</b>	Access Categories
<b>ACK</b>	Acknowledgment
<b>AIFS</b>	Arbitration Inter-Frame Space
<b>BPSK</b>	Binary Phase Shift Keying
<b>BSA</b>	Basic Set of Applications
<b>BTP</b>	Basic Transport Protocol
<b>C-V2X</b>	Cellular V2X
<b>CAM</b>	Cooperative Awareness Message
<b>CBR</b>	Channel Busy Ratio
<b>CCH</b>	Control Channel
<b>CPM</b>	Collective Perception Message
<b>CSMA/CA</b>	Carrier Sense Multiple Access with Collision Avoidance
<b>CSR</b>	Candidate Single-Subframe Resources
<b>CW</b>	Contention Window
<b>D2D</b>	Device-to-Device
<b>DCC</b>	Decentralized Congestion Control
<b>DENM</b>	Decentralized Environmental Notification Message
<b>DIFS</b>	Distributed Inter-Frame Space
<b>DSRC</b>	Direct Short Range Communications
<b>EDCA</b>	Enhanced Distributed Coordination Access
<b>eNB</b>	evolved Node B
<b>ETSI</b>	European Telecommunications Standards Institute
<b>ITS</b>	Intelligent Transport System
<b>KPI</b>	Key Performance Indicators
<b>LDM</b>	Local Dynamic Map
<b>LLC</b>	Logical Link Control
<b>LTE</b>	Long-Term Evolution
<b>MAC</b>	Medium Access Control

---

**NAR** Neighbor Awareness Ratio

**OBU** On-Board Unit

**OFDM** Orthogonal Frequency Division Multiplexing

**OSI** Open System Interconnection

**OSM** Open Street Maps

**PDSCH** Physical Downlink Shared Channel

**PER** Packet Error Rate

**PHY** Physical Layer

**ProSe** Proximity Services

**PSCCH** Physical Sidelink Control Channel

**PSSCH** Physical Sidelink Shared Channel

**PUSCH** Physical Uplink Shared Channel

**QAM** Quadrature Amplitude Modulation

**QPSK** Quadrature Phase Shift Keying

**RB** Resource Block

**RC** Resource Counter

**ROS** Robot Operating System

**RRI** Resource Reservation Interval

**RSRP** Reference Signal Received Power

**RSSI** Received Signal Strength Indicator

**RSU** Roadside Units

**SC-FDMA** single carrier frequency division multiple access

**SCI** Sidelink Control Information

**SIFS** Short Inter-Frame Space

**SL** SideLink

**SNR** Signal to Noise Ratio

**SPAT** Signal Phase and Timing Message

**SPS** Semi Persistent Scheduling

**SRM** Signal Request Message

**SSM** Signal Status Message

**SUMO** Simulation of Urban Mobility

**TB** Transport Block

**TBS** Transport Block Size

**TCP** Transmission Control Protocol

**TraCI** Traffic Control Interface

**UDP** User Datagram Protocol

**UE** User Equipment

**V2I** Vehicle-to-Infrastructure

**V2N** Vehicle-to-Network

**V2P** Vehicle-to-Pedestrian

**V2V** Vehicle-to-Vehicle

**V2X** Vehicle-to-Everything

**VAM** VRU Awareness Message

**VRU** Vulnerable Road User

**WLAN** Wireless Local Area Network

**WP** Work Package

## Revision history and approval record

Revision	Date	Purpose
0	14/11/2022	Document creation
1	03/02/2023	Document revision

### DOCUMENT DISTRIBUTION LIST

Name	e-mail
Adrià Pons Serra	adria.pons.serra@estudiantat.upc.edu
Jordi Casademont Serra	jordi.casademont@upc.edu

Written by:		Reviewed and approved by:	
Date	03/02/2023	Date	03/02/2023
Name	Adrià Pons Serra	Name	Jordi Casademont Serra
Position	Project Author	Position	Project Supervisor



## Abstract

Nowadays, Vehicular-to-Everything (V2X) communications are becoming an essential element to improve safe driving conditions and autonomous driving. This thesis presents a comparison of two V2X communication technologies: IEEE 802.11p, and Cellular-V2X. The objective of this study is to evaluate the performance of both technologies in terms of the Medium Access Control (MAC) layer, especially in a congested environment. Therefore, we analyze the different schemes used on these technologies to access shared channel resources and avoid interferences. The study is conducted using several simulation tools: SUMO which allows us to create personalized scenarios, and OMNeT++ used to simulate the network and transmit all the V2X messages between the vehicles. With SUMO we created a highway scenario that can support a high density of vehicles. And OMNeT++ is used to change the main simulation parameters, and obtain results such as all the packets received and sent through the network. Finally, we defined some performance metrics to analyze the results and observe how the technologies react over a congested scenario, with high densities of vehicles.

---

## Acknowledgements

I would like to express my sincerest gratitude to my advisor, Jordi Casadement, for his guidance, and support throughout my Master's journey. I would also like to thank the foundation i2CAT for providing me with the resources and opportunities to pursue my studies, as well as my family, friends, and colleagues for giving me encouragement and support through the development of this project.

# 1 Introduction

Over the past decade, significant advancements have been made in cellular communications and the automotive industry. The goal to increase safety and achieve fully autonomous cars has led to a focus on the role of cellular communications. To encompass these requirements, a new term, "Vehicle to Everything (V2X)," has emerged. V2X technology is designed to enable vehicles to communicate with other vehicles, infrastructure, pedestrians, and networks, and can be used for a variety of applications such as cooperative awareness, advanced driver assistance systems, and even fully autonomous driving. Overall, V2X technologies have the potential to revolutionize the way we think about transportation and make our roads safer for everyone.

This project is performed in the framework of the foundation i2CAT, a research and innovation center based in Barcelona, more specifically, in the department of Mobile Wireless Internet with the focus set on V2X technologies. As will be seen in section 3, some of the topics that have been developed in this thesis are involved with several Spanish and European projects like FEM-IoT, INTEGRA, and CPSoSAAware.

**FEM-IoT:** is a consortium of 12 Catalan research centers to boost the emerging sector of the Internet of Things. One of their research sub-projects is "Connected Street Infrastructure", whose objective is to achieve an integrated, flexible, and multi-technology ICT infrastructure. With that mindset, V2X technologies are used to analyze the mobility and latency requirements of an urban scenario.

**INTEGRA:** is a Spanish project focused on the development of digital instrumentation applicable to assisted driving to increase safety through anti-collision sensory systems and intelligent networks. So, in this project, i2CAT is leading the actions related to intelligent networks, specifically using simulation tools to develop connectivity systems.

**CPSoSAAware:** CPSoSAAware is a European project aimed to develop the models and software tools to allocate computational resources to the cyber-physical systems' end devices. This project extends to the automotive sector with a mixed traffic environment with semi-autonomous connected vehicles. Here, i2CAT is responsible to implement the security stack for vehicle communications and to develop simulation tools to test those services.

## 1.1 Scope

This project is divided into two main sections with different objectives. The first one is devoted to studying the main simulator tools related to V2X technologies. In this section, we analyze and combine the simulators to create different scenarios related to specific V2X use cases. These scenarios are used to obtain a high variety of results which had been used as a contribution to several research projects. Therefore, the implementation of these use cases is a really good opportunity to study the strengths and weaknesses of the simulators, which are used to develop the next section of this thesis.

The second and principal objective of this study is to analyze and compare the main

standards used in V2X technologies. So, in this section, we use the previous simulation tools to analyze the performance of the IEEE 802.11p standard, usually named Dedicated Short-Range Communications (DSRC), and the later introduced Cellular V2X technology, which is specifically designed to achieve the main V2X requirements and use cases. In this case, the target is to compare their performance related to the Medium Access Control (MAC) layer, so modified some parameters of the physical layer for a better analysis.

## 1.2 Work Plan

The development of this thesis had a duration of 5 months corresponding to the first semester of the academic year 2022/2023. It started in mid September 2022 and lasted until January 2023. Nevertheless, we started this project with some background knowledge due to a previous internship in i2CAT, where we begin working with some of the simulators. In figure 1 we can see a Gantt Diagram of the work plan followed, which is divided into several work packages (WP) explained above:

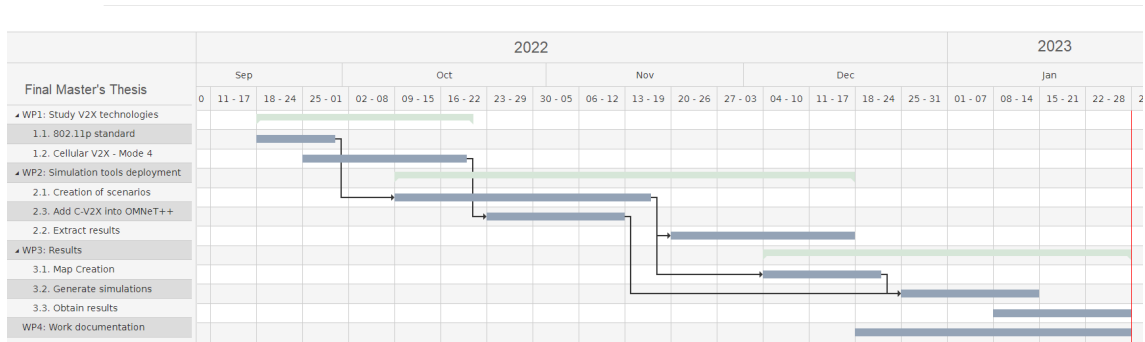


Figure 1: Types of Vehicle-to-Everything communications.

- WP1 - Study V2X technologies:** The first WP lasted a month and was dedicated to studying the state of the art of the technologies that are analyzed in this thesis:
  - 1.1. IEEE 802.11p
  - 1.1. Cellular V2X - Mode 4
- WP2 - Simulation tools deployment:** The second WP is related to the first objective of this thesis, to study the simulation tools and obtain some initial results. And since all the initial simulations were done using only 802.11p technology, we added a sub-WP dedicated to introducing LTE-PC5 Mode 4 into the simulator OMNeT++.
  - 2.1. Creation of scenarios: In this section, we created the two main scenarios with SUMO simulator, which will be later analyzed on OMNeT++. Since these scenarios are used to collaborate in several research projects, we took more time in order to obtain feedback from the partners involved, so it lasted around a month and a half.

- 2.2. Extract results: Once created the scenarios, we can simulate the network elements with OMNeT++ and obtain different results. And as happened in the previous section, we also needed to obtain feedback from the partners of the project to arrange a valid format for the results, so it lasted more than a month.
  - 2.3. Add C-V2X into OMNeT++: And last, since all the previous simulations were done with 802.11p technology, we had to modify the OMNeT++ simulator in order to include a module named “open C-V2X”, which implements the LTE-PC5 protocol.
- **WP3 - Results:** This WP is focused on the second and main objective of this project, which is to compare the two most used V2X protocols: 802.11p and C-V2X.
    - 3.1. Map creation: First, we need to create a new scenario that can contain a higher density of vehicles in order to study both technologies in a congested environment.
    - 3.2. Generate simulations: Then, we need to execute several simulations with both technologies, and with different densities of vehicles. This part was specifically time-consuming since each simulation could last from 12 to 36 hours to be completed (depending on the density of vehicles used).
    - 3.3. Obtain results: In this last section, we will collect the results of all the previous simulations and compute some performance parameters to compare them. We plot the results and then extract conclusions from them.
  - **WP4 - Work documentation:** Finally, the last WP is used to document all the results and write the final version of this master’s thesis.

### 1.3 Budget

Regarding the budget of this project, it has been developed in the framework of the foundation i2CAT. All the software used for the simulator was open source, so no cost is involved. However, i2CAT provided us with a computer to do all the simulations, which has an estimated price of 1500€.

In terms of manpower, the project had a duration of 20 weeks from mid-September 2022 to the end of January 2023, and it required one person working full-time. Therefore, an 8-hour workday for 20 weeks corresponds to 800 hours.

## 2 State of the art

In this section, we explain the main characteristics of the V2X technologies, how are implemented, and which are their objectives. Also, we introduce which are the two principal technologies used to implement V2X communication: IEEE 802.11p and Cellular V2X or LTE-V.

### 2.1 Vehicle-to-Everything Communications (V2X)

The automotive industry is increasingly using wireless communications to enhance the driving experience, improve safety and make transportation more efficient. In this context, many companies and organizations have started to develop a new type of technology called V2X, referred to as Vehicle-to-Everything communication. Some examples of advancements regarding V2X technologies include the development of 5G networks that provide faster and more reliable communication, the use of machine learning to improve traffic flow, and the use of blockchain technology to secure communication between vehicles.

V2X technologies denote various forms of communication that allow vehicles to communicate with other vehicles, infrastructure, and devices. The main communication is classified as follows, and can be seen in Figure 2:

- V2V (Vehicle-to-Vehicle) Communication: This technology allows vehicles to communicate with each other in real-time, and share information such as their position, speed, and direction. This information can be used to prevent collisions and improve traffic flow.
- V2I (Vehicle-to-Infrastructure) Communication: This technology allows vehicles to communicate with infrastructure such as traffic lights, road signs, and weather sensors. This information can be used to optimize traffic flow, reduce emissions, and improve safety.
- V2P (Vehicle-to-Pedestrian) Communication: This technology allows vehicles to communicate with pedestrians, and alert them of the presence of a nearby vehicle.
- V2N (Vehicle-to-Network) Communications: This technology allows vehicles to communicate with other internet-based networks in order to access real-time information and services. This can include things like weather updates, traffic reports, and even parking availability.

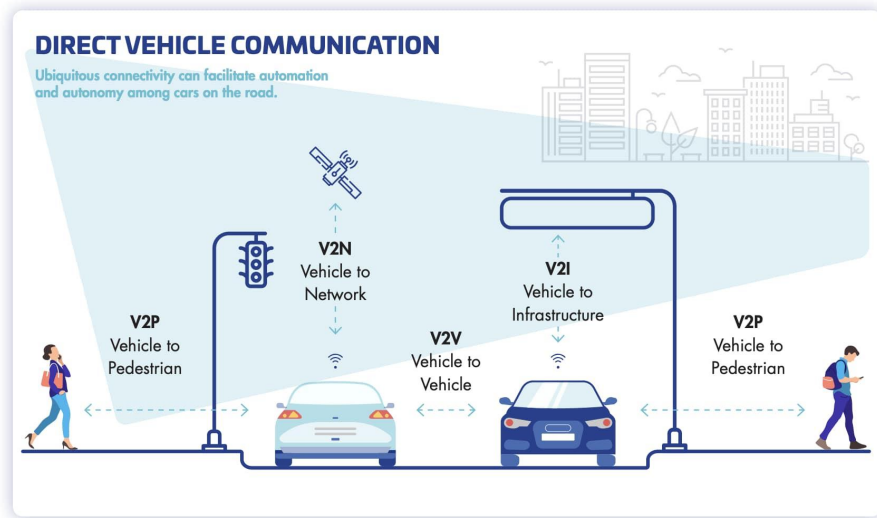


Figure 2: Types of Vehicle-to-Everything communications.

However, there are still challenges that need to be overcome in order for V2X technologies to reach their full potential. These include issues related to data privacy and security, as well as the need for standardization and cooperation between different stakeholders. Despite these challenges, V2X technologies have the potential to significantly improve safety, reduce emissions, and optimize traffic flow, making them a key area of focus for the future of transportation.

In this thesis, we focus on analyzing the two main technologies used to implement V2X communications: IEEE 802.11p and C-V2X LTE-V.

The first technology to be implemented was the IEEE 802.11p, a wireless communication standard known as Direct Short Range Communications (DSRC) [1], and the ITS-G5 standard (Intelligent Transport System specified in Europe by ETSI). This standard was made in 2009, so it has been widely used in the past years. It operates in the 5.9 GHz frequency band and is characterized by high-speed, low latency, and secure communication between vehicles. However, the initial intention of DSRC was to transmit short-range (around 300 m) basic safety messages between vehicles and therefore it is not intended for high bandwidth requirements of V2N applications.

The second technology considered is Cellular-V2X, based on the 3GPP Long-Term Evolution (LTE) Release 14 standard [2]. This standard is an evolution of the LTE Release 12 proximity services (ProSe), which was focused on public safety and some V2V communications. However, to cover all the requirements needed for V2X communications, Release 14 includes some new features, for instance, Device-to-Device (D2D) communications as an evolution of the previous ProSe. C-V2X implements several modes, and with Mode-4 we can also operate at the ITS 5.9 GHz band, and is centered on supporting day-1 safety services. In future releases, 3GPP has introduced 5G into vehicular communications with Release 16 NR-V2X (5G-New Radio). However, this new release is focused in cover new and advanced use cases like adding services using licensed spectrum.

## 2.2 V2X Protocol Architecture

The protocol architecture for V2X technologies is guided by the Cooperative Intelligent Transport Systems (C-ITS), which is composed of several ITS stations:

- Personal ITS station: Regarding the devices used by pedestrians, such as smartphones.
- Vehicle ITS station: Used on On-Board Units (OBUs) installed inside the vehicles.
- Central ITS station: For traffic management centers.
- Roadside ITS station: Used on Roadside Units (RSUs), located on traffic infrastructure like traffic lights and signals.

With all these types of ITS stations, we can establish different communication modes, which follow a common reference architecture, as seen in Figure 3. This protocol has the same functionalities as the Open System Interconnection model (OSI), which is divided into seven layers. However, the ITS protocol combines some of these layers resulting in four main blocs: application layer, facilities layer, network & transport layer, and access layer, and on top of that, there are two vertical entities responsible for the management and the security of the system. A comparison between both architectures can be seen in Table 1:

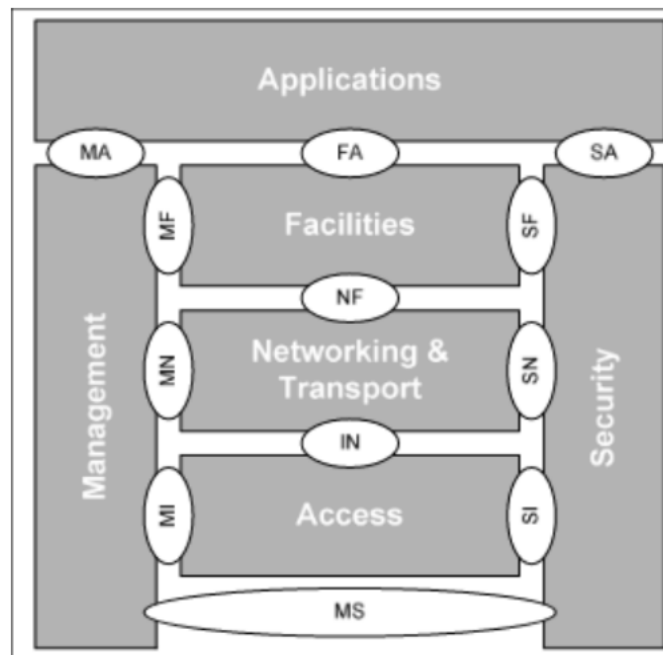


Figure 3: ITS reference architecture.



ITS Reference Architecture	OSI Model
Application	-
Facilities	Application
	Presentation
	Session
Network and Transport	Transport
	Network
Access	Data Link
	Physical

Table 1: Mapping between ITS and OSI architectures.

Now we explain in detail the main characteristics of these layers.

## Applications

The ITS application layer has been added to incorporate new applications and uses cases into the stack. A group of applications is called Basic Set of Applications (BSA) [3], and is divided into three main classes, each with different objectives and requirements:

1. Active road safety: The goal of this class is to improve traffic safety by exchanging periodically information between vehicles to create cooperative awareness, and as a result, prevent road casualties.
2. Cooperative traffic efficiency: The goal of this class is to improve road traffic management, and increase traffic efficiency in terms of travel times, fuel consumption emissions, etc. Implementing these use cases is usually needed to obtain information from the infrastructure.
3. Other applications: These include applications providing other services such as those for infotainment.

## Facilities

The ITS facility layer covers the application, presentation, and session layers of the OSI model, so it covers all those functionalities. Since the main objective of this layer is to provide information to the upper layer, ITS applications, the ITS facility layer is also known as basic service. These facilities can be classified regarding the type of support provided, where we have:

- Application support facilities: to support the application functionalities.
- Information support facilities: to provide data and manage the functionalities of a database.
- Common support facilities: to provide services for communication and session management.

Moreover, the facilities layer implements several types of messages to enable V2X communications, which are standardized under the ETSI ITS protocol (European Standardization). We have a huge variety of messages centering on different purposes, such as:

- Cooperative Awareness Messages (CAMs): Periodic messages sent by all stations giving information about the status and attribute information of the originating device. It contains information like the time, position, speed, direction, etc. . .
- Decentralized Environmental Notification Messages (DENMs): Event-driven messages sent by OBUs and RSUs upon detection of an abnormal situation (traffic jam, break down vehicle, roadworks ...) to alert other vehicles or road users within a geographical area.
- Vulnerable Road User (VRU) Awareness Message (VAM): Enhance the protection of road users such as pedestrians, bicyclists, motorcyclists as well as animals that may pose a safety risk to other road users.
- Collective Perception Message (CPM): Contains information about the onboard sensors (like the range or the field of view), and the detected vehicles or objects (with information like their position, speed, and size).
- Signal Phase and Timing (SPAT) Message: These messages contain information about the state of traffic lights, including the current phase and timing information. They are used to provide vehicles with information that can be used to optimize traffic flow and improve safety.
- Signal Request Message (SRM): Used to request services from an intersection signal controller.
- Signal Status Message (SSM): Contains information about the internal state of the controller.

In the framework of this thesis, we are simulating CAM messages to study the performance of different technologies

## Network and Transport

The transport layer is responsible for ensuring that data is transmitted correctly and reliably between different devices. The ITS architecture uses the Basic Transport Protocol (BTP) which provides an end-to-end transport service. It is responsible for multiplexing the messages from the different processes at the ITS facilities layer, and demultiplexing the messages received through the GeoNetworking protocol. The multiplexing technique used is based on ports, where each port corresponds to a different identifier associated with a specific process. For example, to implement the facilities for CAMs and DENMs the ports 2001 and 2002 are selected respectively. Additionally, the BTP allows the Facilities layer to access the services provided by the GeoNetworking protocol.

There are two types of BTP headers (BTP-A and BTP-B) and their selection is specified in the GeoNetworking Common header. BTP-A is for interactive packet transport and contains information about the source and destination ports. On the other hand, BTP-B is designed for broadcast communications and only contains information about the destination.

Next, the network layer is responsible for routing and relaying information between different devices and is implemented through the GeoNetworking protocol. This protocol uses

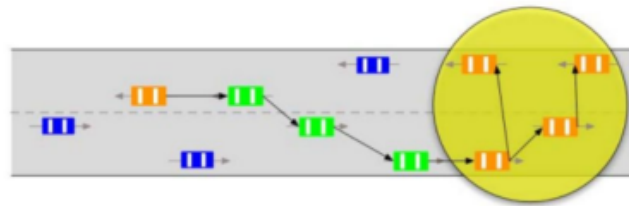
geographical positions and areas to route packets across the ITS ad hoc network. It enables infrastructure-less communication and meets the vehicle networking requirements, such as support for high node mobility and continuously changing network topology [4].

The GeoNetworking protocol has two main functions: geographical addressing, which is based on sending a packet to a specific node, or selecting a specific geographical area and sending the packet to all the nodes in it. And geographical forwarding, which uses the network topology information stored on the nodes to make forwarding decisions based on the selected destination.

And regarding the routing techniques of the GeoNetworking protocol, there are three different packet forwarding schemes, as can be seen in Figure 4:



(a) GeoUnicast



(b) GeoBroadcast



(c) Topologically-scoped broadcast

Figure 4: GeoNetworking routing schemes.

- GeoUnicast: A single destination is defined and the packet is continuously forwarded by intermediate nodes.
- GeoBroadcast: Once selected a destination geographical area, the packet is continuously forwarded until reaching that area, and all the nodes of that area will re-broadcast that packet.
- Topologically-scoped broadcast: We define a certain number of hops and the packet is continuously re-forwarded until it reached the n-hop nodes.

## Access

And last, the ITS Access Layer covers the Medium Access Control (MAC), the Logical Link Control (LLC), and the Physical Layer (PHY). This layer is responsible for the physical communication between devices and has defined several standards to implement wireless V2X communications. In this case, we are comparing two technologies, first, the 802.11 Wireless Local Area Network (WLAN), developed by the IEEE, and second the Cellular V2X proposed by the 3GPP. Both technologies will be analyzed in more detail in the following sections.

### 2.3 IEEE 802.11p

IEEE 802.11p was the first implementation of V2X technologies and is the basis for the DSRC and the ITS-G5 (specified in Europe by the ETSI) standards. This standard is the direct evolution of the IEEE 802.11a, which was improved in order to handle high node mobility and gradually changing vehicular environments.

The Physical layer of the IEEE 802.11p is really similar to its predecessor. IEEE 802.11a used a frequency bandwidth of 20 MHz in the 5 GHz band, however, the new standard works at the 5.9 GHz band with a channel bandwidth of 10 MHz, which makes it more robust to fading and other propagation effects. Also, IEEE 802.11p incorporates OFDM (Orthogonal Frequency Division Multiplexing) into the physical layer and uses the same modulation and coding scheme as IEEE 802.11a. It can reach data rates from 3 to 27 Mbps, using coding rates 1/2, 2/3, or 3/4, and it implements several modulations like BPSK (binary phase shift keying), QPSK (quadrature phase shift keying), 16-QAM (quadrature amplitude modulation), or 64-QAM.

Now, with regards to the MAC layer, the IEEE 802.11p uses the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) algorithm, where the nodes have to sense the medium to check if it is idle, and a mechanism based on random backoff is performed to reduce the probability of collisions. If the medium is sensed idle during a period of time called the Distributed Inter-Frame Space (DIFS), the station is allowed to transmit. And, if the channel is sensed as busy, that node can not transmit a packet until the current transmission has ended. To define if the medium is busy or not, the received power has to surpass a sensitivity threshold, which must be higher than the receiver's sensitivity level. Once the busy period has finished, the node needs to wait a random backoff time to avoid two nodes to start transmitting at the same time. That backoff time is computed by multiplying a constant  $aSlotTime$  and a random integer selected from a Contention Window (CW). The CW is standardized as 15, and the  $aSlotTime$  is defined as  $13\mu s$ . Moreover, the backoff timer is decreased as long as the channel is sensed idle, paused when a transmission is in progress, and resumed when the channel returns to idle for more than the DIFS. After each data frame is successfully received, the receiver transmits an acknowledgment (ACK) frame after a Short Inter-Frame Space (SIFS) period, which is implemented on unicast communications. However, in vehicular Ad-Hoc networks, we can not transmit ACKs since the vehicles may be out of range before receiving them. Therefore, broadcast communications are used, which don't require ACKs and remove their exponential backoff added to unicast communications. A deeper analysis of these

protocols is shown in [5].

Additionally, IEEE 802.11p improves this system by adding different access categories (ACs) with different priorities, which is known as Enhanced Distributed Coordination Access (EDCA). These changes were defined in standard 802.11e, which improved the quality of service by substituting the DIFS with 4 Arbitration Inter-Frame Space (AIFS). The AIFS is a method of prioritizing one AC over the other, and functions by shortening or expanding the period a wireless node has to wait before it is allowed to transmit its next frame. This system is used in vehicular communications to define different priority levels among distinct types of V2X messages, as can be seen in table 2:

AC	TC ID	CW (min)	CW (max)	AIFS	Intended Use
AC-VO	0	3	7	58 $\mu s$	High priority DENM
AC-VI	1	7	15	71 $\mu s$	DENM
AC-BE	2	15	1023	110 $\mu s$	CAM
AC-BK	3	15	1023	149 $\mu s$	Multi-hop DENM and other data

Table 2: ITS-G5 Traffic priority classes

In an ITS ad hoc network, there isn't any central infrastructure, and all the moving vehicles are connected to each other forming a mesh-like structure. This makes the network topology vary constantly, and have an unpredictable number of vehicles within the communication range. Consequently, in the case of high-density scenarios, the vehicles may require a number of resources beyond the channel capacity. As such, the Decentralized Congestion Control (DCC) mechanism is used to avoid channel congestion and allow fairer access to limited resources. The way DCC works is that the vehicle adapts its transmission parameters according to the measured channel load. There are different DCC access techniques to help balance the channel, like the followings:

- Transmit Power Control: Reducing the output transmitted power to reduce the resulting interferences.
- Transmit Rate Control: Increases the time between consecutive packets.
- Transmit Data rate Control: Reduces the transfer rate.

However, the DCC mechanism is not used in the scope of this project, since we pretend to study the performance of the MAC layer, and how it manages highly congested scenarios (without using any congestion control scheme).

## 2.4 LTE-PC5

The second access technology that we study is C-V2X, which is based on the 3GPP LTE Release 14 standard. This standard has a lot of improvements compared to its previous releases making it achieve all the requirements needed to perform vehicular communications. In this section, we analyze these improvements and all the characteristics of the resulting LTE-PC5 technology.

One of the main improvements made in Release 14 was the addition of sidelink (SL) communications, which allows the vehicles to directly communicate with each other without the need for any infrastructure. As a result, LTE-V can work with two different interfaces, as shown in Figure 5:

- **E-UTRAN Uu radio interface:** This is the cellular interface that is present in all LTE systems. It is an interface between the evolved Node B (eNB) and the user equipment (UE). This interface has unicast communications over the physical channels of LTE (like Physical Uplink Shared Channel, PUSCH, or Physical Downlink Shared Channel, PDSCH), and it can be used to implement V2I communications.
- **PC5 interface (V2X Sidelink):** This new interface allows us to support direct communication between vehicles (V2V) based on the LTE SL. Also, this interface supports multicast and broadcast communications since all the vehicles can transmit a message to all the vehicles in their range.

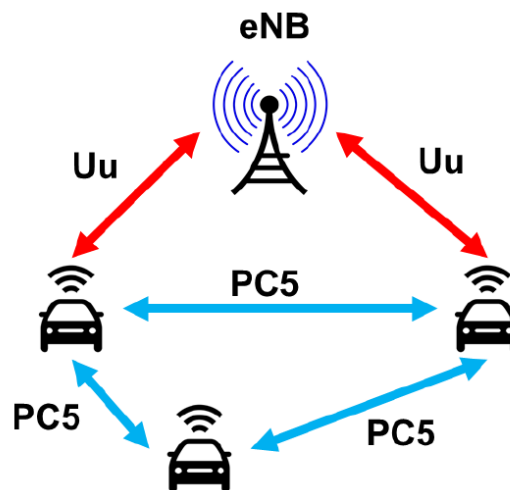


Figure 5: V2X communication interfaces.

Considering that the Uu interface implements V2I communications, and the PC5 interface implements V2V communications, both can be used simultaneously and independently for transmitting and receiving information. In this case, PC5 is explicitly addressed to support a scenario with a high density of vehicles, moving at high speed. It also works at the 5.9 GHz band, and it is appropriate to transmit periodic information like the positions and the speed of the vehicles. Moreover, the SL transmissions avoid the need to cross the whole network, which significantly improves the latency.

Now, we analyze the possible D2D communications made over the PC5 interface. First, there were implemented two different modes (mode 1 and mode 2) whose objective was to improve the battery of the devices when they were out of coverage. But that also produced an increase in latency when the vehicles recover the coverage. Then, with LTE Release 14, there were included two new modes to implement V2V communications. A representation of the modes can be seen in Figure 6:

- **Mode 3:** In this mode, the cellular network (eNB) is responsible to select and

manage the radio resource that will be used in the V2V communications. Therefore, it will dynamically assign the resource to each vehicle using the Uu interface, and then the vehicles will communicate with each other using the PC5 interface. So, the vehicles will have to request the resources needed by the eNB before communicating to other vehicles. For obvious reasons, this mode is only available when the vehicles are on coverage.

- Mode 4:** This mode is the direct competitor for the IEEE 802.11p DSRC protocol. In this case, the vehicles will select autonomously the radio resources needed to establish direct V2V communications. It is based on a distributed scheduling scheme where vehicles select their resources from a specific resource pool. It also allows the addition of the DCC to reduce the congestion of the channel. With mode 4 the vehicles can operate without the coverage of an eNB, which makes it one of the most important features of the LTE-V. For these reasons, we base this thesis on comparing the LTE-PC5 Mode 4, with the IEEE 802.11p protocol.

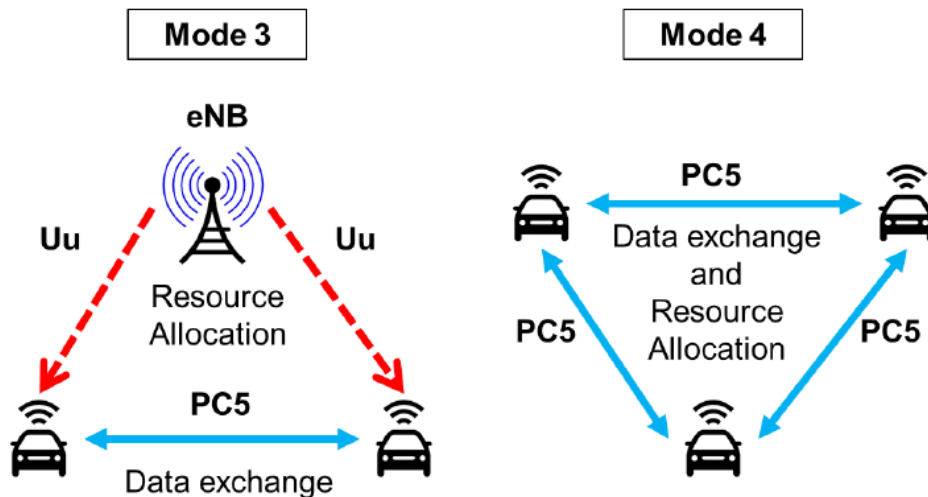


Figure 6: V2X communication modes.

Another one of the changes added in LTE Release 14 was the introduction of a new resource grid that combines the time and the frequency domain. As seen in Figure 7, the time domain is divided into subframes of 1 ms, and the frequency domain is divided into physical Resource Blocks (RB). These are all the parameters involved to define the physical layer of the protocol:



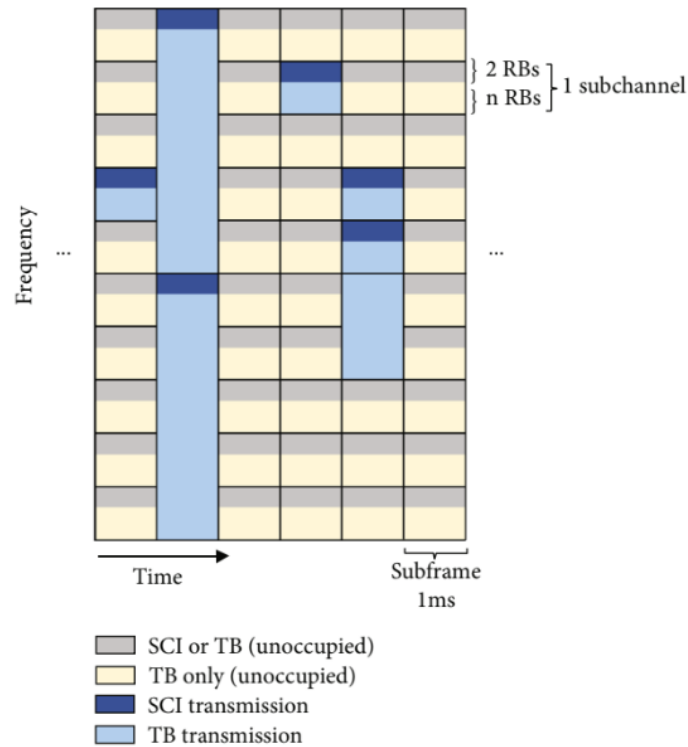


Figure 7: Structure of the LTE-PC5 resource grid with 10 subchannels per subframe.

- **Channelization:** Frequency channel bandwidth of 10 MHz or 20 MHz.
- **Channel division:** Each channel is divided into subframes (time), and subchannels (frequency). And these subchannels are also divided into resource blocks (RBs).
- **Subframes:** One subframe is 1 ms long and contains 14 symbols.
- **Resource block:** It is defined as the smallest unit of frequency resources that can be allocated to a user. One RB occupies 180 kHz and contains 12 subcarriers which implies a subcarrier spacing of 15 kHz. With a channelization of 10 MHz, the channel can contain a total of 50 RBs, but with 20 MHz, we achieve 100 RBs.
- **Subchannel:** A subchannel is defined as a certain group of RBs in the same subframe. Therefore, a channel can be defined as a certain number of subchannels with a given number of RBs/subchannel. The subchannel is used to transmit data and control information.
- **UE properties:** UE properties: The maximum transmission power of a vehicle is 23 dBm, and the typical sensitivity of the receiver is around -92 dBm.
- **Modulation and coding schemes:** The modulation used depends on the type of data to be sent. For data transmission can be used QPSK and 16-QAM, but, for control information transmission only can be used QPSK modulation. Also, are used some coding schemes like turbo coding and normal cyclic prefix.

Once defined the resource grid that will be used, we need to specify which physical channels



are implemented in the LTE-V protocol. All the data is transmitted in Transport Blocks (TBs), which contain the information of a full packet and are transmitted over the Physical Sidelink Shared Channel (PSSCH). The other type of data is the control information which is known as the Sidelink Control Information (SCI) and is transmitted through the Physical Sidelink Control Channel (PSCCH).

In the standard, there are defined different transmission rates such as 10 packets per second (pps), 20 pps, or 50 pps. In this thesis, we use the default value of 10 pps, which transmits a data packet every 100 subframes (100 ms).

The SCI is used as a scheduling assignment since it includes information about the modulation and the coding schemes used to transmit the TBs, the number of RBs used, and the Resource Reservation Interval (RRI) for the Semi Persistent Scheduling (SPS). Due to this correlation between the control information and the data sent, the transmission is known as a pair SCI+TB, where both blocks must be transmitted in the same subframe. To implement the transmission of this SCI+TB pair, LTE-V defines two subchannelization schemes, which are shown in Figure 8:

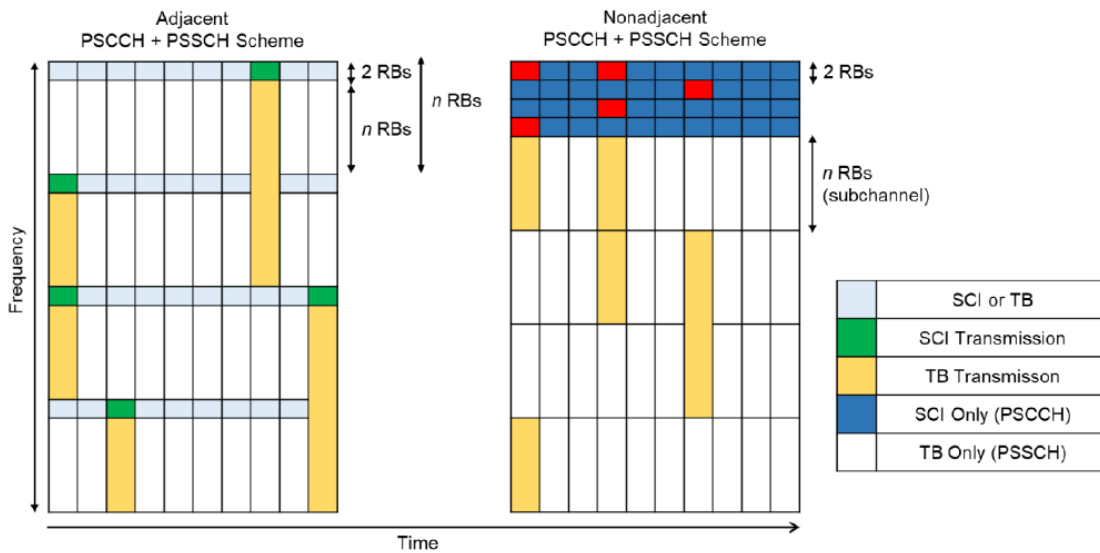


Figure 8: LTE subchannelization schemes.

- **Adjacent PSCCH + PSSCH:** In this scheme, the SCI+TB pair is transmitted in adjacent RBs. The SCI occupies always the first two RBs of the first selected subchannel by the vehicle, and the TB is transmitted after the RBs of the SCI. Depending on its size, the TB can occupy more than one subchannel. If this is the case, it will be able to occupy the first two RBs of any of the extra subchannels that it needs.
- **Non-Adjacent PSCCH + PSSCH:** In this scheme, the SCI+TB pair is transmitted in non-adjacent RBs. The frequency-time scheme is divided into two pools. The first one is only committed to transmitting the SCIs (always occupying only two RBs), and the second pool is divided into subchannels that are used to transmit only TBs.

At the PHY and MAC layers, C-V2X relies on the same single carrier frequency division multiple access (SC-FDMA), used by the LTE uplink. Also, the resource allocation is normally associated with the SPS and the DCC mechanisms which improve the SL communications. The sensing-based SPS is performed in the MAC layer and is used to support the vehicle UE to autonomously select appropriate radio resources, and the UE can also reuse these resources for several consecutive periods. The number of consecutive transmitted packets with the same selected resource is defined as Resource Counter (RC), which, by default, selects a random value from the interval [5, 15]. This counter is reduced after each transmission, and once the RC goes to zero, we maintain the same resource with a keep probability of  $P_k$ , or else, we trigger the selection of a new resource with a probability  $1 - P_k$ . In addition, the vehicle cannot retransmit a packet if it has been lost.

The SPS scheme can be seen in figure 9, and its procedure works as follows:

- When a vehicle wants to select a new subchannel in time  $T_1$ , it has a time window from  $T_1$  and  $T_2$ , where  $T_2$  adds a latency constraint to  $T_1$  (the latency should be lower or equal to 100 ms). This time is called Selection Window and is used to identify the Candidate Single-Subframe Resources (CSRs), which consist of a set of contiguous subchannels in a single subframe. The number of consecutive subchannels depends on the size of the message.
- Then, the vehicle analyzes all the information received in the last 1000 subframes known as Sensing Window. This information is used to create 2 lists  $L_1$  and  $L_2$ , where  $L_1$  contains all the CSRs in the Selected Window, and  $L_2$  is declared empty.
- If a SCI is correctly received and specifies that the CSR is used at that same time, the corresponding CSR will be removed from list  $L_1$ .
- Also, the vehicle measures the average Reference Signal Received Power (RSRP) and compares it to a threshold, so if a CSR has a RSRP measurement higher than the threshold, that one is also excluded from the list  $L_1$ .
- These two steps are repeated until the CSRs represent less than 20% of the initial resources, and in each iteration, the threshold is decreased by 3 dB.
- Finally, the CSRs with lower average Received Signal Strength Indicator (RSSI) from list  $L_1$ , will be added to list  $L_2$ , and that list is reported to higher layers. Then on the MAC layer is selected a random candidate resource is.

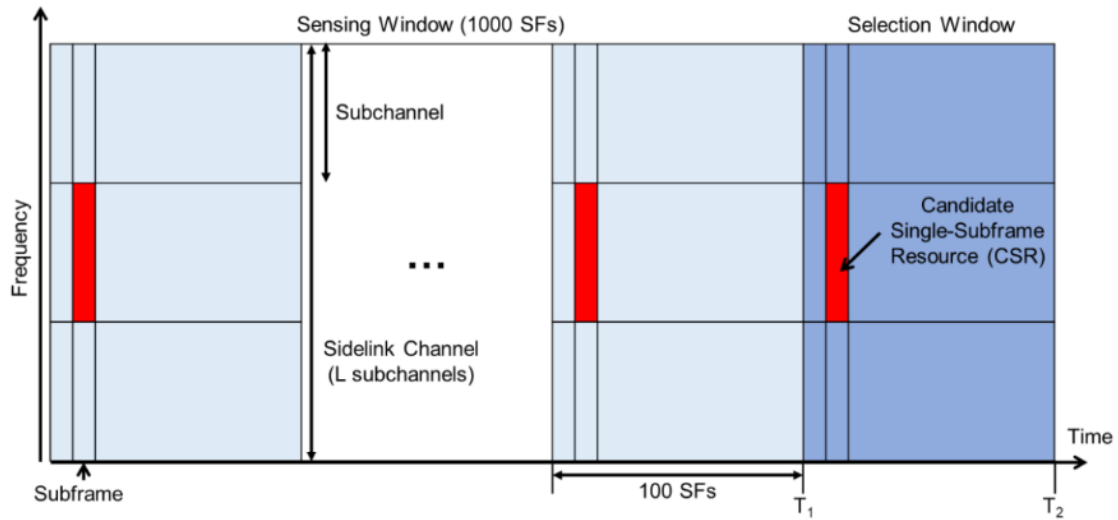


Figure 9: Semi-Persistent Scheduling in C-V2X. Sensing and Selection Windows.

And regarding the DCC protocol, it is not used in the framework of this thesis, since this one is focused on analyzing the performance of different technologies in the MAC layer, and we pretend to put the channel resources at their limit by congesting the network with a high number of vehicles.

## 3 Deployment of simulation environments and results

This chapter is devoted to presenting all the simulators used to perform this thesis, as well as studying their capabilities. The main simulators used are SUMO, OMNeT++, and CARLA. SUMO is a simulator focused on creating network scenarios and implementing their mobility models, so it is used to create all the roads and vehicles of the simulation. Then, OMNeT++ is used to simulate the network and transmit all the V2X messages between vehicles. It can implement different technologies like IEEE 802.11p and LTE-PC5, and is really useful to extract statistics from those communications. And last, CARLA is an open-source simulator used for autonomous driving research, which provides a 3D representation of the scenarios. Also, CARLA can introduce and control vehicles inside the simulation, making emphasis on collecting information about their sensors [6], such as cameras, GPS, line detectors, and RADARs among many others. However, despite the correlations that this simulator may have with the orders, it is not used in this project. As it is explained in this section, CARLA is taken into account in the decision of scenarios, and for future related works, but it is not needed in the framework of this project.

Moreover, this section presents the main results obtained from each simulator, and how are they involved in the development of the FEM-IOT, INTEGRA, and CPSoSAAware projects.

### 3.1 Simulation environments

#### 3.1.1 SUMO

The SUMO, “Simulation of Urban Mobility” is an open-source, portable simulator that handles vehicular traffic simulations and their characterizations. It allows the experimentation of several mobility models over an infinite possibility of road topologies, which can be created and modified on the simulator.

To start a new simulation we need three essential files: a network file, a routing file, and a configuration file.

- **Network file:** Contains the description of the physical topology of the scenario. This may include all roads and intersections of the map, as well as other elements like the traffic logic, and roundabouts. Using the sumo naming convention, the roads or streets are referred to as edges and the intersections as junctions. Two edges are connected by junctions.
- **Routes file:** Species the types of vehicles that will be added to the simulation and all the possible routes that these vehicles can follow. In this file, we can predefine different types of vehicles (like cars, motorcycles, vans, buses, etc. . . ), and the physical properties of the vehicle (such as the shape, the color, the maximum and minimum values of speed, and acceleration, and the minimum gap distance with the vehicles ahead). Also, different routes are identified by their road id, and each of them specifies all the edges and junctions that define that route.

- Configuration file:** A general configuration file that indicates which network and routes file is used. It can contain other types of files to add more elements to the simulation like inserting buildings into the scenario or controlling the traffic lights, and some global configuration parameters such as the duration of the simulation and the step length, which defines the interval of time between consecutive simulation steps (its minimum value is 1 ms).

The SUMO simulator has implemented many features that allow us to create and customize a large number of different scenarios. The main tool for creating these scenarios is Netedit, which is a visual editor that allows us to modify or create traffic networks from scratch, an example of the interface is shown in figure 10.

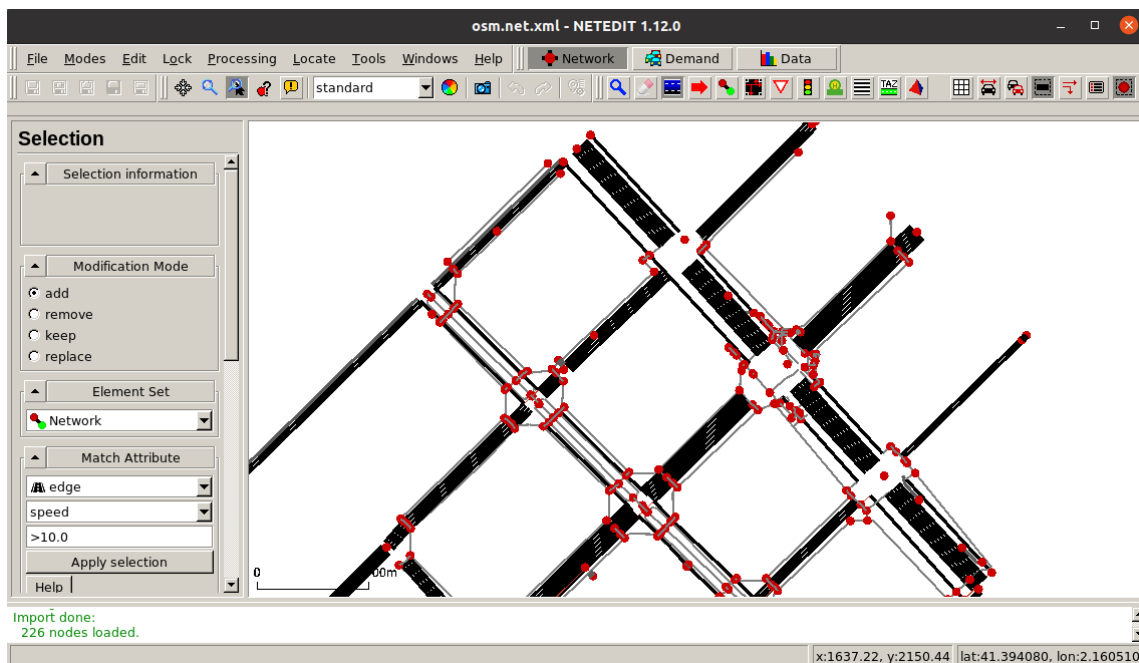


Figure 10: Netedit's visual interface.

Here we can see that there are many different options that allow us to personalize a scenario: modify the networks, create highways or intersections, add traffic lights, or modify the direction of the vehicles on each road. All these elements are the ones recorded in the “network file”. In addition, we can customize the traffic of this simulation by adding “demand elements”, where we can select what type of vehicle we want (car, motorcycle, bicycle, truck, bus...), and predefine specific routes for each vehicle in the simulation. The same goes for pedestrians, who can walk on sidewalks and pedestrian crossings. These parameters are stored on the “routes file”. Therefore, one of the best advantages of using Netedit is that it generates all the files needed to run the simulation: the network file with a “.net.xml” extension, the routes file with a “.rou.xml” extension, and also the configuration file with a “.sumocfg” extension.

With all these elements we can create some accurate simulations, but we are only using a few functionalities of SUMO. Now we only can create and personalize a scenario, but we can't extract information from the simulation, nether modify it. In these circumstances

is when we introduce the TraCI “Traffic Control Interface”, which is a SUMO interface that allows us to retrieve the value of different parameters in an ongoing simulation and manipulate them in real-time. It does not have an interface like Netedit, but it has a large number of commands to modify all aspects of a simulation.

The general structure of Traci is organized according to the following actions: ”Retrieve values” and ”Change state”, both related to the following topics:

- **Traffic Objects:** Including different types of pedestrians and vehicles, and their respective routes.
- **Detectors and Outputs:** Such as induction loops, area detectors, multiple input-output detectors, and calibrators.
- **Network elements:** related to edges, junctions, and lanes.
- **Infrastructure:** related to traffic lights, bus stops, parking areas, overhead wires, and the rerouter.
- **Simulation elements:** related to obtaining values, displaying information, or analyzing certain points of interest in the simulation.

These TraCI commands can be executed from a Python script and allowed us to customize the scenarios and extract a large number of parameters from the simulations. As will be seen in the following sections, TraCI is used to modify the vehicles in the simulation, create routes, produce collisions, and obtain some simulation results.

Taking into account all of that, SUMO is our main simulator to create and define new scenarios for our simulations, and using TraCI we also manage to integrate these maps with other simulations like OMNeT++ and CARLA. Once we integrate a SUMO scenario with OMNeT++ we can simulate V2X messages between all the vehicles, as will be explained in section 3.1.2. And since CARLA can provide us a 3D representation of the same map, we can do a co-simulation with SUMO and CARLA, where both are using the same scenario at the same time, as can be seen in figure 11.





Figure 11: CARLA and SUMO co-simulation.

### 3.1.2 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) is an extensible and modular simulation library and framework for the research and development of complex distributed systems. Countless simulation models and model frameworks have been written on top of OMNeT++ by researchers in diverse areas, and vehicular networks are one of them. OMNeT++ works by assembling individual components/models into larger ones, which makes the models easier to be reused and incorporated into different applications. Moreover, OMNeT++ is also considered a network simulator platform since it has a large number of users, and allows us to use diverse model frameworks to implement more specific functionalities. An example of the framework used to implement the IEEE 802.11p technology is shown in figure 12.

As it can be seen, the first important framework is INET, which is an open-source library containing various models to simulate communication networks, and was created particularly for the OMNeT++ environment. Some of its main features are:

- Models for the Internet stack like IPv4, IPv6, Transmission Control Protocol (TCP), and User Datagram Protocol (UDP).
- Wired and wireless interfaces such as Ethernet or IEEE 802.11.
- Support for physical environment modeling, like the propagation model or the presence of obstacles.
- Can be used as a base to create other simulation frameworks.

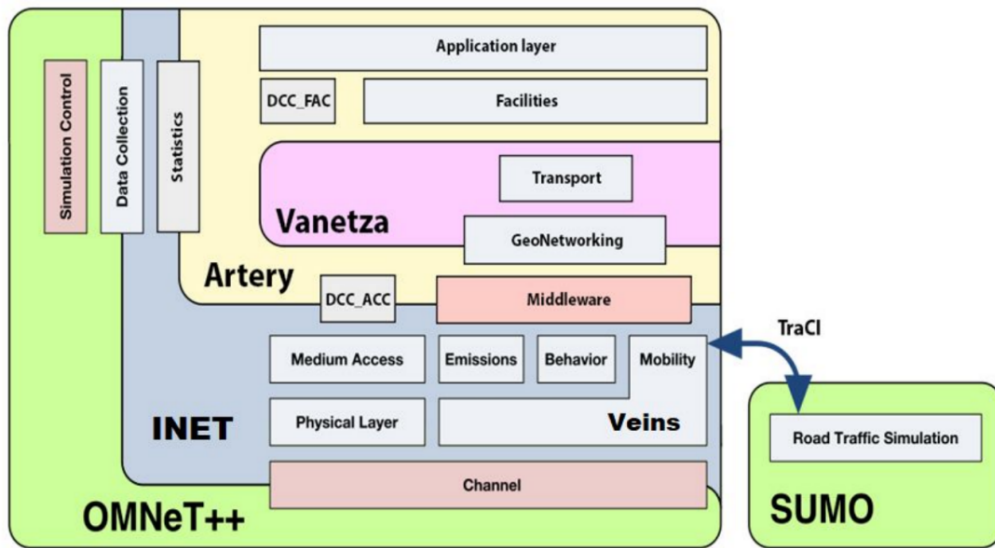


Figure 12: Simulation framework overview.

Another useful framework is Artery, which was originally developed as an extension of Veins, yet now it can be used independently. Artery corresponds to the application and facilities layers, which enable the generation of V2X messages like CAMs and DENMs. Moreover, Artery’s middleware provides common facilities for multiple ITS-G5 services running on individual vehicles. Its architecture is depicted in figure 13.

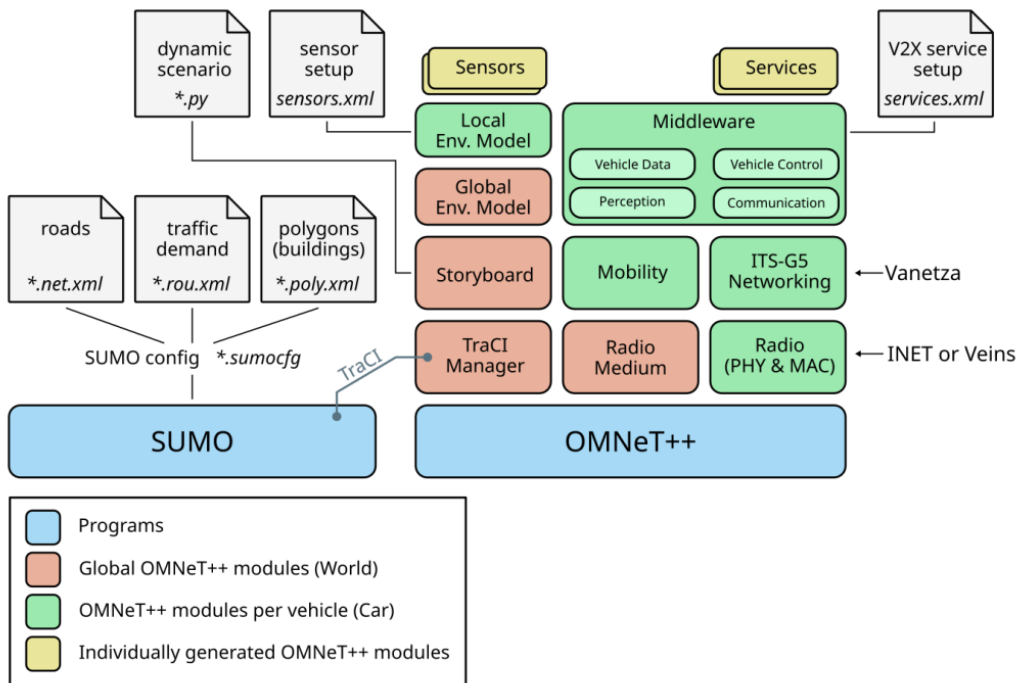


Figure 13: Artery simulation framework.

In this case, the SUMO and OMNeT++ simulators are running simultaneously. On the



left side, there is the traffic simulator SUMO that provides information about the position of the vehicles, and on the right side, we have the OMNeT++ runtime environment, which simulates and evaluates the cooperative awareness basic service (implementing vehicular communication and computing the propagation of the packets). The interaction of these simulators is made possible using a TCP socket and a standardized protocol known as the TraCI. This connection allows us to identify the movements of the vehicles on SUMO and replicate it over the corresponding nodes on OMNeT++.

With all these modules we managed to carry out the necessary simulations with IEEE 802.11p technology, nonetheless, to analyze other technologies we need other frameworks. The module OpenCV2X is an open-source implementation of the 3GPP standard C-V2X (Rel 14) Mode 4 [7]. It is based on an extended version of the SimuLTE framework which enables LTE network simulations. It also can be integrated with Artery to provide full ITS-G5 standardization across the entire communication stack.

## ROS

With this description of the OMNeT++ simulator we manage to obtain a lot of results, but we still need an efficient way to extract those results or even send them to other applications like CARLA simulator or a database to store the results. For this purpose, we integrate ROS in the OMNeT++ simulator.

The Robot Operating System (ROS) is a set of software libraries and tools that help create robot applications, providing a huge variety of services such as hardware abstraction, low-level device control, inter-process message passing, and packet management.

ROS processes are represented as nodes in a graph structure, and the edges that connect the processes are called topics. These nodes can have several functionalities like sending messages through a specific topic, calling services from other nodes, or providing services to a subscribed node. All this is orchestrated by a process called ROS Master, which registers the connected nodes, establishes node-to-node communications for different topics, and updates the parameter server. Messages and services do not pass through the ROS Master, it only establishes peer-to-peer communication between all node processes after they have registered.

All ROS nodes must have a unique name to distinguish the running processes on the ROS network, and those can be used to create clients to interact with other nodes. Also, nodes can be used to advertise services, representing the actions that can be performed. To complete this interaction, we need to use ROS topics, which are information buses that send and receive messages between nodes. These topics also must have a unique name and can have different contents depending on the type of service to be implemented, like sensor data, motor control commands, or status information among other possibilities.

Finally, implementing ROS into our simulations allows us to share the messages generated with external applications, for example: to store the messages on a database, or to compute and predict the trajectories of the vehicles in order to avoid a collision. Also, another reason to choose ROS over other tools is its compatibility with the CARLA simulator, which has a “ROS Bridge” module. This module sends external information to the vehicles and this one can be interpreted as an added sensor to the vehicle.

## 3.2 Creation of scenarios

Once introduced the two main simulators, we need to create the scenarios that will be used, and therefore define the network and the routes to be used by each vehicle. Also, to make the most of SUMO we created two different scenarios with different methods and for different purposes.

### 3.2.1 Collisions scenario

The first scenario we created is a realistic scenario located on the Eixample of Barcelona, in particular, the intersection between “Passeig de Gràcia” and “Carrer Aragó”, and some surrounding streets, as it can be seen on figure 14. The main objective of this scenario will be to simulate vehicle collisions in a realistic environment and extract information about them, with the objective of generating a dataset that could be used to study the trajectories of the vehicles to predict or detect collisions.

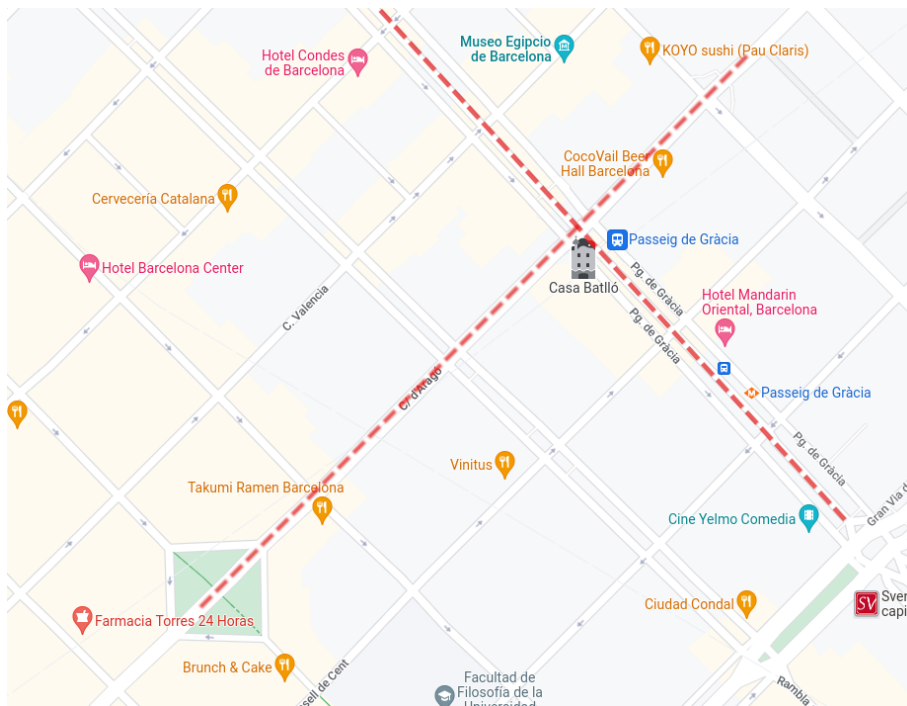


Figure 14: Map view of Aragó with Passeig de Gràcia streets.

As we introduced before, we can create the whole scenario manually on Netedit, but since we know the location of those routes, SUMO has a tool named OpenStreetMaps (OSM) that helps us to extract the network structure of a map. OSM is a free editable map of the whole world that creates and provides free geographic data such as street maps. This tool provides us with all the network data, but to use it on the simulation we have to use the OSMWebWizard, a tool implemented by SUMO. Once we run OSMWebWizard, it opens a web browser where we can select a specific location in the world and it generates the corresponding network file of that spot. Its interface is shown in figure 15. Also, with this tool, we can select what kind of vehicles we want in the simulation and its duration,

but since we want to personalize those parameters, we only used the network file as a base to create our scenario.

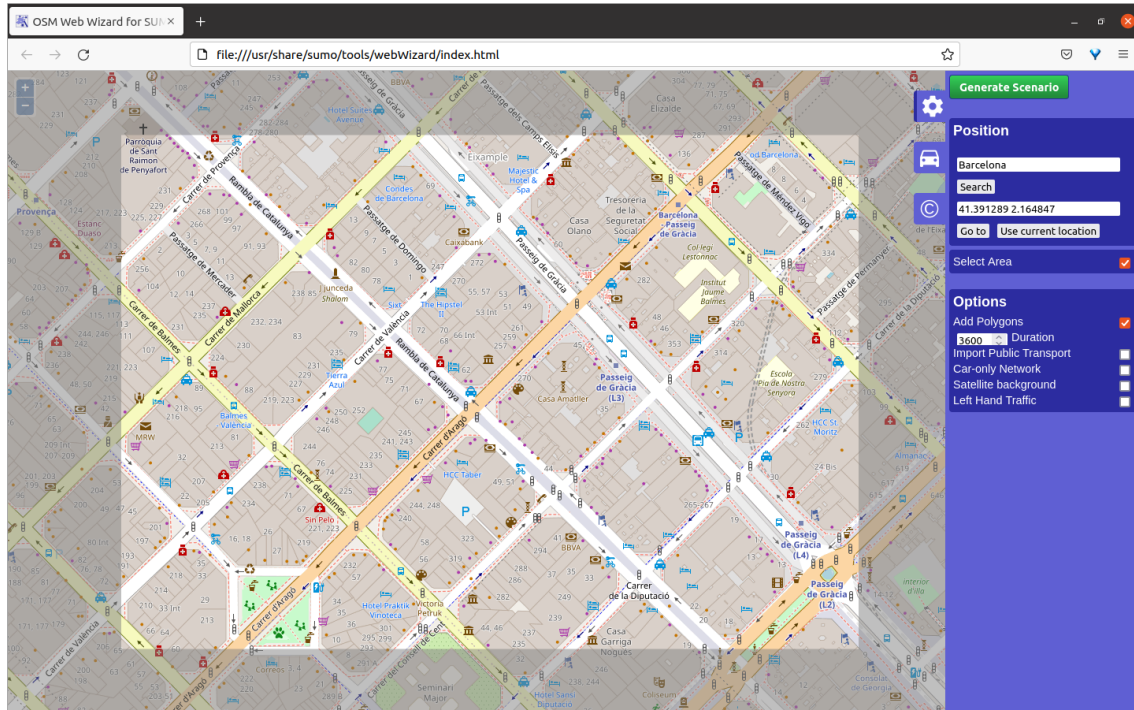


Figure 15: OpenStreetMap Web Wizard tool.

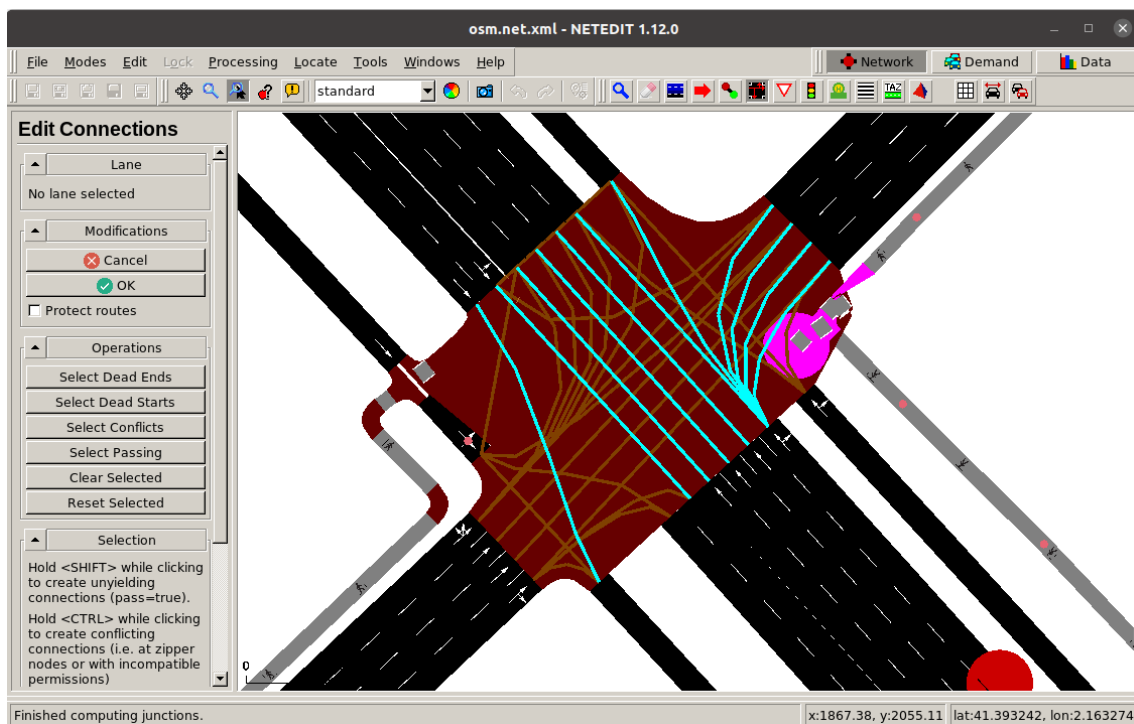


Figure 16: Junction with prohibited lane changes.

Next, we need to modify and correct some errors from the loaded network file. OSM is really useful since it creates all the roads on the map, which are also geolocalized with their latitude and longitude coordinates. However, we still need to correct some elements like connections between edges and junctions, changes of lanes, or the traffic lights of an intersection. A closer look at the changes made in an intersection can be seen in figure 16.

Once we had checked and corrected all network elements of the map, we can start adding on-demand elements, like adding vehicles and defining roads on the map.

Since our main objective was being able to recreate collisions on the simulation, we decided to personalize the main parameters of the vehicles created, so we generated 100 different types of vehicles with their main parameters randomized by the following distributions:

- **acceleration:** the acceleration ability of the vehicle, in  $\text{m/s}^2$ . Follows a gaussian distribution  $N(3.3, 0.5)$ .
- **deceleration:** the deceleration ability of the vehicle, in  $\text{m/s}^2$ . Follows a gaussian distribution  $N(5.5, 0.5)$ .
- **length and width:** the size of the car in the simulation. 5.5m x 1.5m.
- **maximum speed:** Maximum velocity of the vehicle in  $\text{m/s}$ . follows a gaussian distribution  $N(8.3, 4)$ , and only accepts values larger than 7.
- **sigma:** The driver imperfection (0 denotes perfect driving). Follows a gaussian distribution  $N(0.5, 0.3)$ , but only accepts values from 0 to 1
- **tau:** The driver's desired (minimum) time headway, based on the net space between the leader's back and the follower's front. Follows a gaussian distribution  $N(1, 0.25)$ , but only accepts values from 0 to 1.
- **apparent deceleration:** the apparent deceleration of the vehicle in  $\text{m/s}^2$ . The follower uses this value as the expected maximal deceleration of the leader. Follows a gaussian distribution with a deviation of 0.5 regards the original declaration of the leader,  $N(\text{decel}, 0.5)$ .

With these changes, we manage to add diversity to the simulation in order to recreate a realistic environment. And on top of that, we also introduced different types of vehicles to the simulation by adding buses, vans, and motorcycles. In table 3 we can see a list of the types of vehicles selected and their main parameters.






Vehicle Class	Shape	Dimensions [m]	mingap [m]	accel [m/s <sup>2</sup> ]	decel [m/s <sup>2</sup> ]	emergency decel [m/s <sup>2</sup> ]	Max speed	Speed Deviation
passeger		5 x 1.8 x 1.5	2.5	2.6	4.5	9	200	0.1
delivery		6 x 2.16 x 2.86	2.5	2.6	4.5	9	200	0.05
bus		12 x 2.5 x 3.4	2.5	1.2	4	7	85	0
motorcyce		2.2 x 0.9 x 1.5	2.5	6	10	10	200	0.1
moped		2.2 x 0.9 x 1.5	2.5	1.1	7	10	45	0.1

Table 3: Types of vehicles used in the SUMO scenario.

Now, we need to create the routes for all these vehicles, and we decided to use a tool of SUMO called RandomTrips. This tool is based on TraCI and also can be executed through a python script. RandomTrips is used to create vehicles with random routes over the whole scenario, but since we have already defined the type of vehicles we want, we only use it to obtain the generated routes. To run this script, we introduced the following parameters:

- **-n:** to indicate the input network file, with all the roads and intersections of the scenario. (extension .net.xml).
- **-r:** to indicate the output file, with all the randomly generated routes in the scenario. The length of the routes is also random. (extension .rou.xml).
- **-e:** to specify the execution time of the simulation. If we simulate a longer time, we generate more random routes. Value set to 7200.
- **-p:** time in seconds between the generation of consecutive vehicles. Value set to 0'8.
- **--intermediate:** to specify the number of intermediate points that will be added to the route. So we can create longer or shorter routes for the vehicles. Value set to 8.
- **--fringe-factor:** parameter from 0 to 10, to increase the possibility that the routes start/end at the frontiers of the simulated network. Value set to 10.

With this tool, we manage to create around 5000 different routes which will be selected randomly for each generated vehicle.

Finally, with all the types of vehicles and the routes defined, we can use TraCI to insert the vehicles into the simulation. In this case, we start adding vehicles to the simulation until reach a constant value of 150 vehicles. All the vehicles introduced are selected randomly with a different type of vehicle and a different route, so when TraCI needs to add a new



vehicle it selects one of the 100 types of vehicles predefined, and one of the 5000 routes also predefined.

## Types of Collisions

Now, we have all the basic elements for the simulation (the network, the vehicles, and the routes) we need to study how to make collisions happen on running time. To simplify the possible types of collisions we identify only frontal collisions and lateral collisions.

**Frontal Collisions** : Normally, frontal collisions appear on straight roads near some intersection or crossing, and occur because the front vehicle stops suddenly and/or the back vehicle can not break in time. This kind of collision is already implemented due to the parameters of the vehicles we used. Since there are cars with random acceleration and deceleration values, different maximum speeds, and different values of temerity (sigma and tau), all that together generate possibilities for these collisions to happen. Therefore, these collisions are easily found in the simulation.

**Lateral Collisions** : Now, generating lateral collisions is more complex, because these could only happen at an intersection, and all the intersections are regulated by traffic lights. In this case, we need to go one step further into the TraCI and modify more elements related to the vehicles.

Our objective is that some cars could ignore a red traffic light and go into an intersection generating a lateral collision. To achieve it, we need to change the state of the vehicle by modifying the command “speed mode”. This command is a bitset that forms an integer and each bit corresponds to a different check:

- **bit0**: Regard safe speed.
- **bit1**: Regard maximum acceleration.
- **bit2**: Regard maximum deceleration.
- **bit3**: Regard right of way at intersections (apply to approaching foe vehicles outside the intersection).
- **bit4**: Brake hard to avoid passing a red light.
- **bit5**: Disregard right of way within intersections.

With this into account, some example modes could be:

- **Speed mode: 31**, [0 1 1 1 1 1]. Default value, all checks on.
- **Speed mode: 32**, [1 0 0 0 0 0]. All checks off.
- **Speed mode: 55**, [1 1 0 1 1 1]. Disable right-of-way check.
- **Speed mode: 39**, [1 0 0 1 1 1]. Run a red light even if the intersection is occupied.

For our use case, we could select speed mode 39, which would be enough to cause some lateral collisions at intersections. But, to generate some extra collisions, we decided to ignore all the checks and use mode 32. Therefore, once we add a new

vehicle into the simulation we choose one of the predefined types already created and set its speed mode to 32. And to visualize these changes in the simulation, we also changed the color of the vehicle to red, so it can be easily identified.

By making these changes we achieved to recreate different types of collisions in this scenario. Now, we should decide how the system reacts when a collision occurs, and this response can also be defined through TraCI. In this case, we decided that when two vehicles collide they will remain in the collision's position for 1 second, and afterward, both vehicles will be removed from the simulation to avoid interfering with the rest of the vehicles.

Finally, with all the elements of the simulation implemented, we can execute the complete SUMO simulation and observe how the cars move through the network, and how the collisions occur. The final simulation is shown in figure 17, where we can see different types of vehicles and some red cars which drive more recklessly to create some collisions



Figure 17: SUMO simulation of the collision scenario.

### 3.2.2 OMNeT++ scenario

#### Create Scenario

For this scenario, we want to focus on studying the CAM messages sent between the vehicles of the simulation. So we need a network capable of producing collisions between the messages sent, also the map should contain buildings to study path loss effects.

To create this scenario we used a previously defined map from CARLA database, Town05. This map is defined for CARLA, but it also contains all the network files needed to start a

SUMO simulation: “Town05.net.xml”, “Town05.rou.xml” and “Town05.sumocfg”. Having those files allows us to make a co-simulation with CARLA and SUMO, where after every simulation step made by SUMO, it sends actualized information to CARLA, so both simulations work simultaneously representing the same vehicles.

The network file for this scenario can be seen in figure 18, and we choose this scenario because it has a little urban area in the center and an external highway that goes around them which can be used to introduce more vehicles to saturate the network and create more interferences among the messages sent.

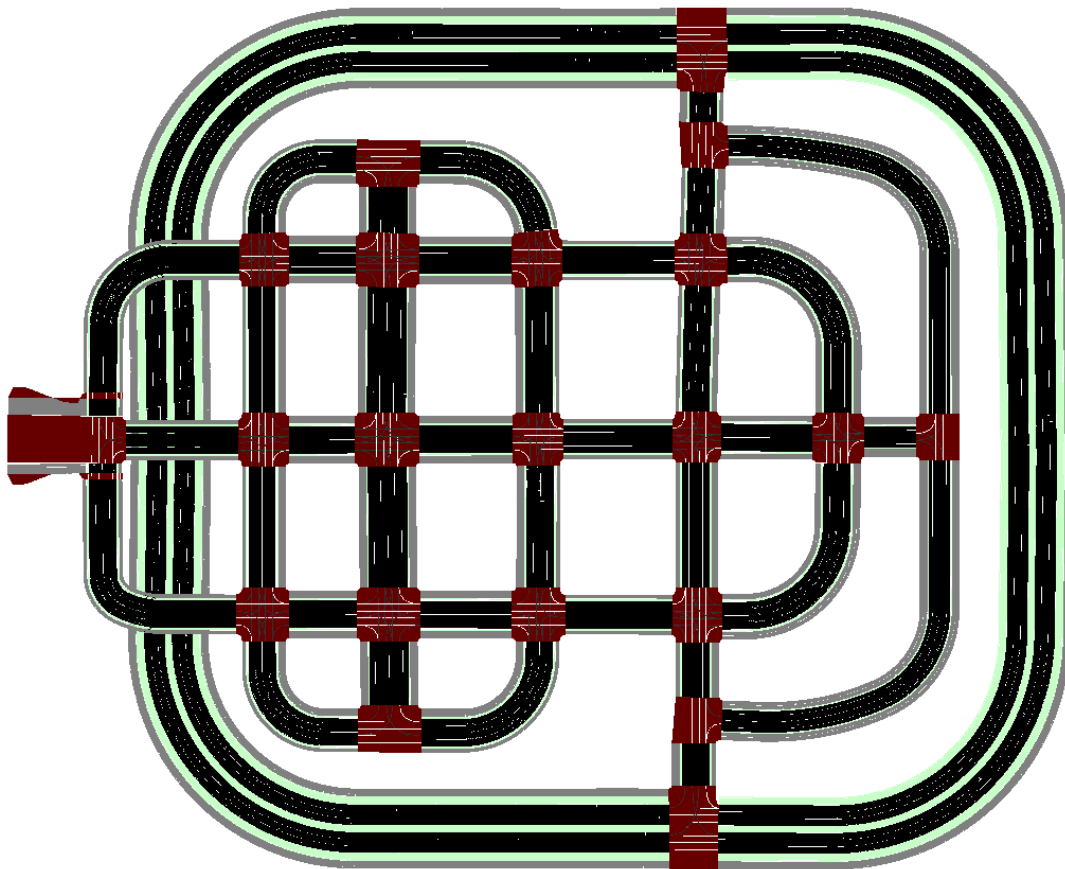


Figure 18: Netedit view of Town05 scenario.

Since we already have the network files for this scenario, we only need to introduce the vehicles and define their routes. In this case, we redid the routes from scratch on Netedit so we can distribute the vehicles along the scenario.

First, to create new routes, we need to introduce “demand elements” with the option “Demand<sub>i</sub>Routes”, which allows us to select consecutive roads on the map to define the final path, as can be seen in figure 19. Also, we have a menu on the left to modify some parameters related to the route, such as changing the color, the number of repetitions on that route, and if the roads need to be consecutive or not. Now, with the roads created, we can add vehicles manually to a specific route with the option “Demand<sub>i</sub>Vehicles”, which allows us to select the type of vehicle that we want.



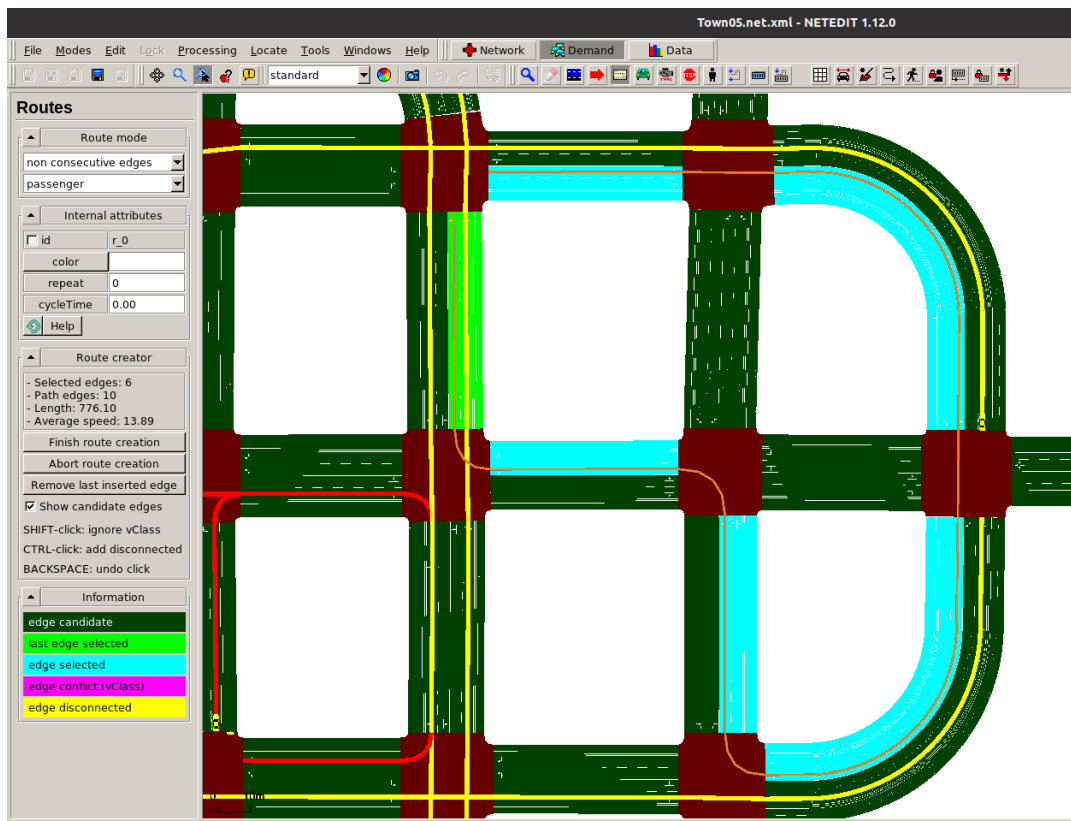


Figure 19: Netedit interface to define new routes.

Next, we can add obstacles into the scenario which will act as buildings that will interfere with vehicular communications. To do so, we used the CARLA map of Town05 to know the exact location of the buildings and then we added the obstacles manually. To place the obstacles on the map we used the option “Shapes” to define some polygonal shapes where the building should be. All these shapes are stored on an XML file that must be added to the Netedit configuration file, and after that, the type of material is selected on the OMNeT++ simulator. With that, we finished creating the scenario, and an example of the final simulation running can be seen in figure 20:

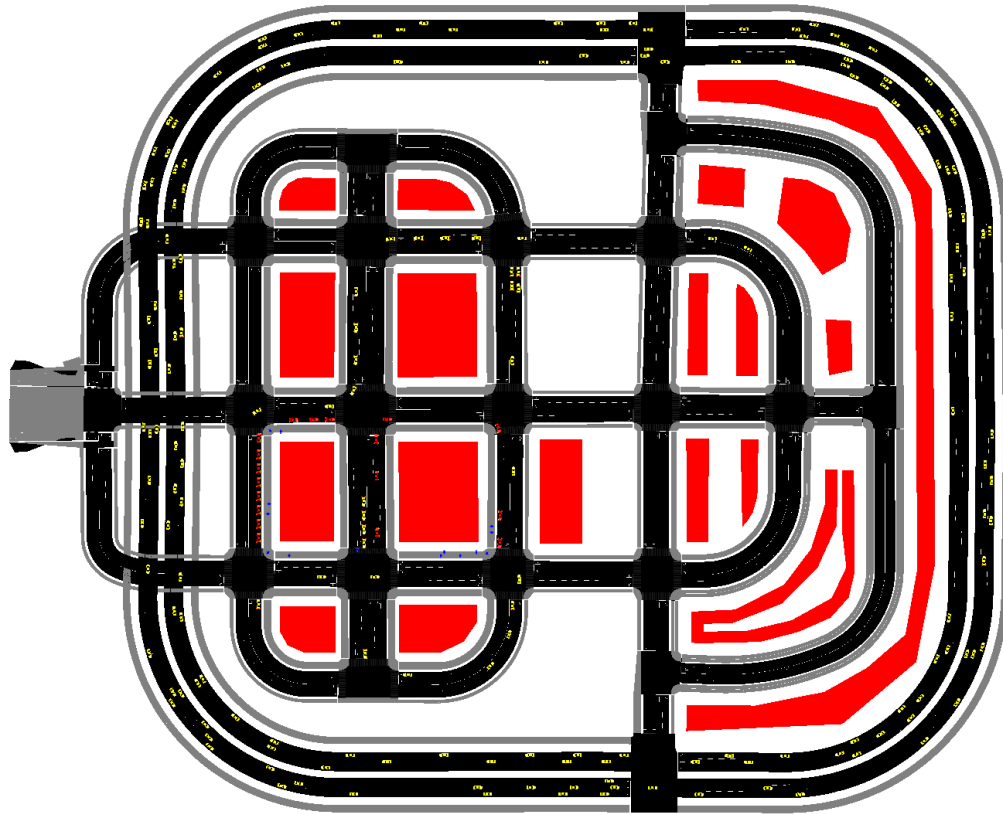


Figure 20: Scenario Town05 running.

## OMNeT++

Now that we have a scenario for the simulation, we need to configure OMNeT++ to define all the elements regarding the communication between vehicles. The specifications of the general configuration parameters are stored in the `omnetpp.ini` file, and some of the main parameters are shown in the following table:

Category	Parameter	Value
Node	Operation mode	802.11
	Carrier Frequency	5.9 GHz
	Bandwidth	10 MHz
	Channel number	180
	Modulation	QPSK
	Bitrate	6 Mbps
	Transmitter power	200 mW
	Receiver sensitivity	-85 dBm
	Energy detection	-85 dBm
	SNIR threshold	4 dB
Medium	Obstacle's loss type	DielectricObstacleLoss
	Path loss type	Rayleigh Fading
	Path loss factor	3
	Background noise type	IsotropicScalarbackground Noise
	Background noise power	-110 dBm
Scenario	CAM message period	100 ms
	Simulation time limit	300 s
	Density of vehicles	25

Table 4: IEEE 802.11p, OMNeT++ simulation parameters.

The parameter values, used to model the individual nodes, were selected based on those specified in the standards. A control channel (CCH) was used, with 10 MHz of bandwidth centered at 5.9 GHz, a channel number of 180, and a default data rate of 6 Mbps. The nodes were configured to transmit with a power of 200mW or 23 dBm, which was below the 33 dBm power limit. The receiver sensitivity was set to -85 dBm, based on the reference from [8].

The radio medium was modeled using the Rayleigh fading profile, which allowed the simulation of highly dense urban environments without direct line-of-sight between the communicating nodes. The corresponding path loss factor value for urban areas, ranged from 2.5 to 3.5, so we choose an arbitrary value of 3. On the physical environment, we define the type of material of the buildings as a dielectric to compute the obstacle loss.

## ROS

To use the ROS in the OMNeT++ simulator, we need to implement the necessary C++ tools, which will be responsible to create the ROS nodes and the topics corresponding

to the messages sent. First of all, we need to start a ROS Master before running the simulation, and once it starts OMNeT++ will create a single ROS node named “Artery” which is our connection point to the OMNeT++/Artery simulation. This node publishes all the information from the transmitted CAM messages and sends them through different ROS topics.

The modifications made to the OMNeT++ simulator are characterized by the addition of a new ROS module, which generates a new service in the simulation. This service is called “ProxyService” and is in charge of creating a personalized topic for each vehicle, and all the CAM messages received on that vehicle will be also sent through this topic.

Therefore when the simulation starts, we have only 1 node corresponding to the simulator, and it publishes all the messages through topics (specific for each vehicle). To better understand this structure, we can see figure 21, where we have a small example with 2 vehicles (vehicle\_0 and vehicle\_1) that publish messages in their respective topics (called “/artery/vehicle\_0/cam\_rx” and “/artery/vehicle\_1/cam\_rx”). And for visualization purposes, to close the communication, we have added two external nodes that have subscribed to the specific topic they want to observe. Therefore each of these nodes can receive the messages sent by the selected vehicle.

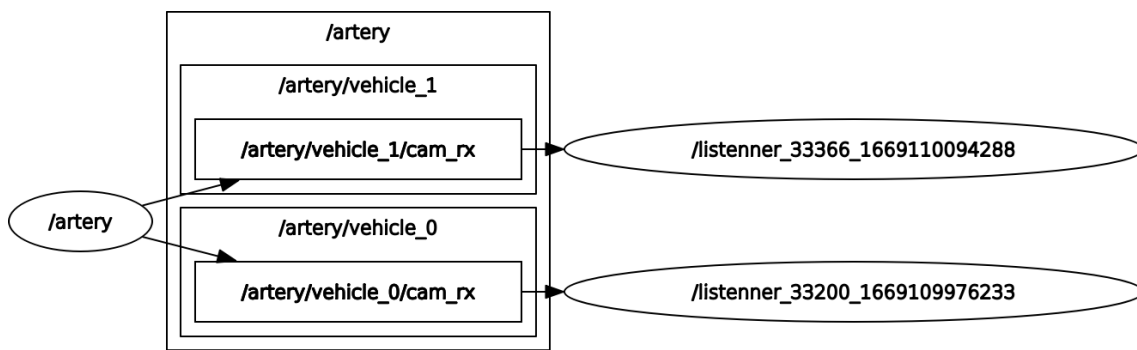


Figure 21: ROS scheme of the nodes and topics involved in the transmission of CAM messages. Example with 2 vehicles.

## ROS standard for CAM messages

CAM messages are standardized according to ETSI ITS with the ASN1 protocol. These messages are divided into four main sections: the header, with generic information about the packet; the basic container, with relevant information about the vehicle position; the high-frequency container, with information about parameters that change rapidly over time (such as vehicle speed and direction); and finally the low-frequency container, with additional vehicle information about less frequent changes (such as activating the exterior lights). In each of these sections, we can find several parameters, which are explained in detail in the standard [9].

These messages are created using the “ProxyService”, mentioned in the previous section, and all containers and parameters are defined in a JSON format and then sent through the corresponding ROS topic. An example of the messages that are sent and their values would be the following:

```

## CAM header
header:
  seq: 81
  stamp:
    secs: 1669110150
    nsecs: 116143003
  frame_id: ''
its_header:
  protocol_version: 2
  message_id: 0
  station_id: 1251505443
  generation_delta_time: 1448

## Basic Container
station_type:
  value: 5
reference_position:
  latitude: 25770
  longitude: 9180
  position_confidence:
    semi_major_confidence: 4095
    semi_minor_confidence: 4095
    semi_major_orientation:
3601
  altitude:
    value: 800001
    confidence: 15

## High Frequency Container
high_frequency_container:
  heading:
    value: 901
    confidence: 10
  speed:
    value: 249
    confidence: 3
  drive_direction:
    value: 0
  vehicle_length:
    value: 1023
    confidence_indication: 0
  vehicle_width:
    value: 62
  longitudinal_acceleration:
    value: 2
    confidence: 102
  curvature:
    value: 0
    confidence: 7
  curvature_calculation_mode:
    value: 0
  yaw_rate:
    value: 0
    confidence: 0
  has_acceleration_control: False
  acceleration_control:
    value: 0

## Low Frequency Container
has_low_frequency_container:
False
low_frequency_container:
  vehicle_role:
    value: 0
  exterior_lights:
    value: 0
  path_history:
    points: []

```

Figure 22: Example of CAM message in JSON format.

## 3.3 Results

### 3.3.1 Collisions scenario

First of all, before extracting results from the simulator, we need to specify how the simulator will deal with these collisions. To do it we must change some parameters of TraCI which are related to the security and the management of collisions. These are some of the parameters that we modified to obtain our results:

- `--collision.action = remove`

Defines which actions are to be taken as a consequence of a collision. In this case, both vehicles involved in the collision are removed so as not to affect the rest of the

simulation.

- `--collision.stoptime = 1`  
Waiting time after a collision. If two vehicles collide, both will remain stationary for 1 simulation step before being removed.
- `--collision-output =scenario/collisions.xml`  
Specifies the name of the file that records all the collisions produced in the simulation.
- `--collision.check-junctions`  
Tag that enables lateral collision detection produced at an intersection.
- `--collision.mingap-factor = 0`  
Minimum distance to detect a collision. Being equal to 0, only physical collisions are registered.

With all these parameters set, we can start to retrieve results, which are stored in the file “collisions.xml”, and contains the following parameters:

<b>time</b>	Simulation time when the collision happens.
<b>type</b>	Type of collisions: “collision” referring to frontal collisions, and “junction” referring to lateral collisions
<b>lane</b>	SUMO lane ID where the collision happens
<b>pos</b>	position inside the lane where the position happens
<b>collider</b>	ID of the vehicle that produces the collision
<b>victim</b>	ID of the vehicle that suffers the collision
<b>colliderType</b>	Type of vehicle that produces the collision
<b>victimType</b>	Type of vehicle that suffers the collision
<b>colliderSpeed</b>	Velocity of the vehicle that produces the collision
<b>victimSpeed</b>	Velocity of the vehicle that suffers the collision

Table 5: List of parameters in ”collisions.xml”.

However, some of these values are not practical for the interpretation of the results, such as the ID of the roads and lanes, or the internal coordinates of the vehicle inside the simulation. Consequently, we have decided to implement another kind of results file. So we decided to record the positions of all vehicles in a single file called ”positions.csv”, which stores the positions of all vehicles at each step of the simulation (every 0.1 seconds), and in case of any collision, this one will also be recorded. In the following table we can see a list of all the recorded parameters:

<b>time</b>	Simulation time when the collision happens
<b>vehicle_id</b>	ID of the vehicle that produces the collision
<b>victim_id</b>	ID of the vehicle that suffers the collision. By default value if there are no collisions: -1
<b>latitude</b>	Position of the vehicle in geographic coordinates: latitude
<b>longitude</b>	Position of the vehicle in geographic coordinates: longitude
<b>speed</b>	Velocity of the vehicle that produces the collision
<b>heading</b>	Direction of the vehicle that produces the collision
<b>collision</b>	Collision indicator. Value equals to "1" if there is a collision, or "0" otherwise

Table 6: List of parameters in "positions.csv".

Finally, with both files, we have enough information to create a database capable of studying the movements and trajectories of the vehicles in order to predict or avoid collisions. And if some collisions occur, we also have information about both vehicles involved in order to have a detailed report of the scene.

These results have been used in the framework of the FEM-IoT project, where they were used as a database to study urban mobility scenarios. In this case, we sent the results to the CTTC (Centre Tecnològic de Telecomunicacions de Catalunya) research center, which used this data to train Machine Learning algorithms to predict the movement of the vehicles and detect possible collisions. An example of the files used can be seen in Figure 23, where there are the collisions.xml file and the positions.csv file respectively.

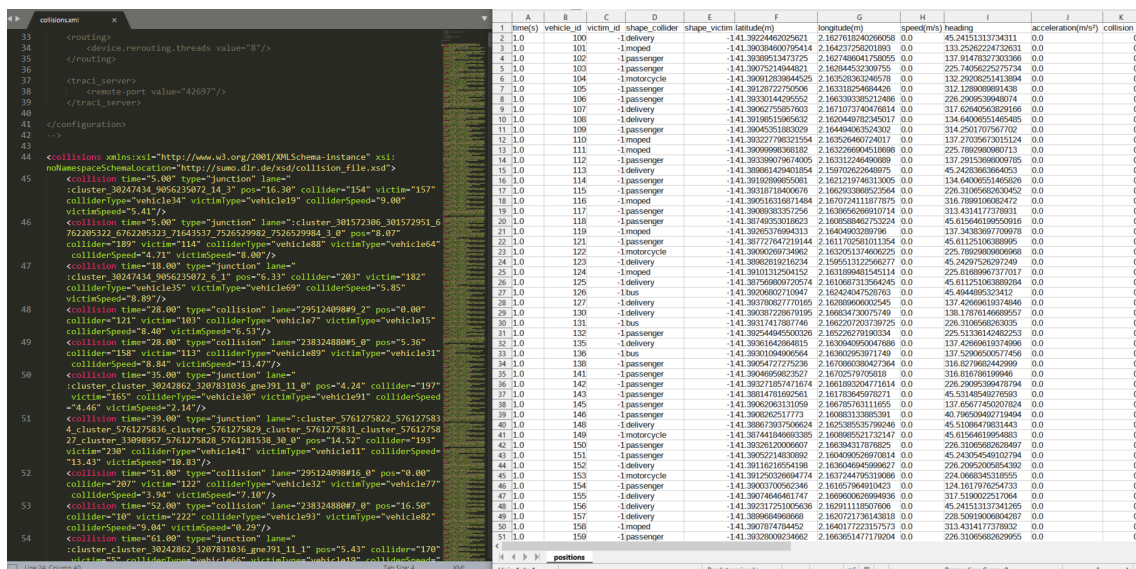


Figure 23: Simulation results. Example of the collisions.xml and the positions.csv files.



### 3.3.2 OMNeT++ scenario

As we have seen previously in section 3.2.2, we are able to extract CAM messages through ROS topics. These messages can be listened to by other applications while OMNeT++ is running, which enables it to establish a real-time synchronization with other simulators, such as CARLA. In addition, ROS messages can be stored in "ROS bags". A ROS bag is a ROS file format used to store ROS message data. This tool allows you to save, process, analyze and visualize the stored messages, which are recorded at the same time as the simulation is running, and can be recreated after the simulation ends. So if we execute the simulation faster than in real-time, the messages are also stored according to that velocity. The ROS bag only records the topics sent by the simulation (which contain the transmitted CAM messages), so it follows the speed the simulator generates those messages.

On the other hand, we are also able to extract specific parameters and save them in CSV format once the simulation is finished. This allows us to extract additional parameters other than the CAM ASN1 format, which facilitates the integration with other simulators. For example, the position in the CAM messages is stored as geographic coordinates (latitude and longitude), but in case of working on a SUMO scenario, it is more useful to know the internal coordinates of the simulation, which can be used in other simulators that use the same scenario (such as CARLA).

As a compilation, these are some parameters that we can extract from the simulations in OMNeT++:

**Vehicle ID:** Vehicle identifier. We can distinguish between the ID used in the SUMO and the internal ID of the OMNeT++.

**Simulation time:** Simulation instants related to the vehicle messages sent. We can identify when messages are generated, when they are sent, and when they are received by each vehicle. This simulation time represents the real-time in which the messages would be sent, which is different from the time the simulation takes to run (a fast simulation may take 1 minute, but represent 5 minutes of real simulation).

**Position:** We also have several formats to locate the vehicle within the scenario. With CAM messages we can obtain the positions in geographical coordinates (latitude and longitude). But with OMNeT++ we can also obtain the cartesian coordinates (X-axis and Y-axis) corresponding to the SUMO scenario.

**Speed and Acceleration:** As we have already seen, with CAM messages we can obtain more specific information about the vehicle sending the message. One of these parameters is the speed and acceleration of the vehicle, which can be used to predict the positions of other neighboring vehicles. In this case, the values can be given in cartesian coordinates (by decomposing the velocities and accelerations into their X-axis and Y-axis components), or give the same information based on the modulus of the vehicle's velocity and direction.

With all these parameters obtained from the simulator, we can create a lot of personalized datasets, and an example can be seen in table 7, where we made a personalized Local



Dynamic Map (LDM). An LDM is a local database stored on each vehicle that collects the information received from other vehicles. In our case, the LDM contains a list of all the CAM messages that a vehicle receives, so it has information about all the vehicles nearby without the need to have a direct line of sight with them. The OMNeT++ simulator already creates an LDM for each vehicle, but it only stores the last CAM messages received on a specified time window. However, we combined the information obtained from the CAM messages and the OMNeT++ simulator to create these LDMs, which contain the main parameters of the messages and their references to the SUMO scenario (like the identifiers or the positions of the vehicles).

<b>LDM of the vehicle</b>	SUMO ID of the vehicle that receives the messages
<b>LDM Simulation Time</b>	Time when the LDM of the vehicle is actualized. Matching the time of the last CAM message received. [Unidades: s]
<b>Vehicle Identifier</b>	SUMO ID of the vehicle that sends the messages.
<b>Generated time</b>	Time when the CAM message is generated
<b>Received time</b>	Time when the CAM message is received
<b>Position X</b>	Position of the vehicle that sends the CAM: SUMO Cartesian coordinates : X Axis [Units: m]
<b>Position Y</b>	Position of the vehicle that sends the CAM: SUMO Cartesian coordinates : Y Axis [Units: m]
<b>Speed X</b>	Velocity of the vehicle that sends the CAM: X Axis [Units: m/s]
<b>Speed Y</b>	Velocity of the vehicle that sends the CAM: Y Axis [Units: m/s]
<b>Acceleration X</b>	Acceleration of the vehicle that sends the CAM: X Axis [Units: m/s <sup>2</sup> ]
<b>Acceleration Y</b>	Acceleration of the vehicle that sends the CAM: Y Axis [Units: m/s <sup>2</sup> ]

Table 7: Personalized parameters of the LDM.

These results are stored in a CSV file that contains the LDMs of all the vehicles of the simulation. So, if we want to extract the LDM of a specific vehicle we just need to filtrate the results by the column “LDM of the vehicle”.

The resulting LDM is used in the CPSoSAAware project, in collaboration with a Greece research center called ISI (Industrial Systems Institute). This center is working on a use case to prevent and detect GPS spoofing attacks, where a drone could send false GPS coordinates to an autonomous vehicle, in order to progressively change its trajectory. Their approach to preventing it is to get information about the position of the neighbor

vehicles to double-check the position of the ego-vehicle (vehicle under the GPS attack). To implement this knowledge of the surrounding vehicles, V2X messages are an improvement compared to previously implemented systems like cameras or radars. In this case, we used CAM messages to broadcast the position of each vehicle, so the ego-vehicle can collect them and create an LDM of their surroundings. For this reason, we simulated and collect all the CAM messages sent to create a personalized LDM for this use case. An example of the resulting file can be seen in figure 24, where there is shown all the parameters explained in table 7.

	A	B	C	D	E	F	G	H	I	J	K
1	LDM of vehicle	LDM sim time	Vehicle ID	Sent_time	Received Time	Position X	Position Y	Speed X	Speed Y	Acceleration X	Acceleration Y
2	vehicle_16	1.106408538398	vehicle_0	1.105928446164	1.106408538398	86.193096	115.895822	0.0	0.0	0.0	0.0
3	vehicle_0	1.108952609607	vehicle_16	1.108472517373	1.108952609607	99.69	140.029082	-0.0	-0.0	-0.0	-0.0
4	vehicle_16	1.206136538398	vehicle_0	1.205928446164	1.206136538398	86.193096	115.895822	0.0	0.0	0.0	0.0
5	vehicle_0	1.208680609607	vehicle_16	1.208472517373	1.208680609607	99.69	140.029082	-0.0	-0.0	-0.0	-0.0
6	vehicle_16	1.306136538398	vehicle_0	1.305928446164	1.306136538398	86.193096	115.895822	0.0	0.0	0.0	0.0
7	vehicle_0	1.308680609607	vehicle_16	1.308472517373	1.308680609607	99.69	140.029082	-0.0	-0.0	-0.0	-0.0
8	vehicle_16	1.406136538398	vehicle_0	1.405928446164	1.406136538398	86.193096	115.895822	0.0	0.0	0.0	0.0
9	vehicle_0	1.408680609607	vehicle_16	1.408472517373	1.408680609607	99.69	140.029082	-0.0	-0.0	-0.0	-0.0
10	vehicle_16	1.506136538398	vehicle_0	1.505928446164	1.506136538398	86.193096	115.895822	0.0	0.0	0.0	0.0
11	vehicle_0	1.508680609607	vehicle_16	1.508472517373	1.508680609607	99.69	140.029082	-0.0	-0.0	-0.0	-0.0
12	vehicle_16	1.606408538398	vehicle_0	1.605928446164	1.606408538398	86.193096	115.895822	0.0	0.0	0.0	0.0
13	vehicle_0	1.608952609607	vehicle_16	1.608472517373	1.608952609607	99.69	140.029082	-0.0	-0.0	-0.0	-0.0
14	vehicle_16	1.706136538398	vehicle_0	1.705928446164	1.706136538398	86.193096	115.895822	0.0	0.0	0.0	0.0
15	vehicle_16	1.806136538398	vehicle_0	1.805928446164	1.806136538398	86.193096	115.895822	0.0	0.0	0.0	0.0
16	vehicle_0	1.808680609607	vehicle_16	1.808472517373	1.808680609607	99.69	140.029082	-0.0	-0.0	-0.0	-0.0
17	vehicle_16	1.906136538398	vehicle_0	1.905928446164	1.906136538398	86.193096	115.895822	0.0	0.0	0.0	0.0
18	vehicle_0	1.908680609607	vehicle_16	1.908472517373	1.908680609607	99.69	140.029082	-0.0	-0.0	-0.0	-0.0
19	vehicle_16	2.006136548521	vehicle_0	2.005928446164	2.006136548521	86.19134	113.485062	2.340034	0.002433	2.340034	0.002433
20	vehicle_11	2.008650667588	vehicle_10	2.008442657441	2.008650667588	323.52687	78.65512	0.00268	-1.189297	0.00268	-1.189297
21	vehicle_0	2.00868061973	vehicle_16	2.008472517373	2.00868061973	101.581703	140.032452	-0.003369	-1.8917	-0.003369	-1.8917
22	vehicle_14	2.008787467269	vehicle_13	2.008579456198	2.008787467269	150.72783	354.407122	0.672845	0.006081	0.672845	0.006081
23	vehicle_10	2.102344592333	vehicle_11	2.101864582186	2.102344592333	322.215296814	0.0	-0.0	0.0	0.0	-0.0
24	vehicle_16	2.106408548521	vehicle_0	2.105928446164	2.106408548521	86.19134	113.485062	2.340034	0.002433	2.340034	0.002433
25	vehicle_13	2.108555813721	vehicle_14	2.10807580265	2.108555813721	153.983911	355.05	0.0	0.0	0.0	0.0
26	vehicle_11	2.108922667588	vehicle_10	2.108442657441	2.108922667588	323.52687	78.65512	0.00268	-1.189297	0.00268	-1.189297
27	vehicle_0	2.10895261973	vehicle_16	2.108472517373	2.10895261973	101.581703	140.032452	-0.003369	-1.8917	-0.003369	-1.8917
28	vehicle_14	2.109145824792	vehicle_13	2.108579456198	2.109145824792	150.72783	354.407122	0.672845	0.006081	0.672845	0.006081
29	vehicle_10	2.202072592333	vehicle_11	2.201864582186	2.202072592333	322.215296814	0.0	-0.0	0.0	0.0	-0.0
30	vehicle_16	2.206136548521	vehicle_0	2.205928446164	2.206136548521	86.19134	113.485062	2.340034	0.002433	2.340034	0.002433
31	vehicle_13	2.208283813721	vehicle_14	2.20807580265	2.208283813721	153.983911	355.05	0.0	0.0	0.0	0.0
32	vehicle_11	2.208650667588	vehicle_10	2.208442657441	2.208650667588	323.52687	78.65512	0.00268	-1.189297	0.00268	-1.189297
33	vehicle_0	2.20868061973	vehicle_16	2.208472517373	2.20868061973	101.581703	140.032452	-0.003369	-1.8917	-0.003369	-1.8917
34	vehicle_14	2.208787467269	vehicle_13	2.208579456198	2.208787467269	150.72783	354.407122	0.672845	0.006081	0.672845	0.006081
35	vehicle_10	2.302072592333	vehicle_11	2.301864582186	2.302072592333	322.215296814	0.0	-0.0	0.0	0.0	-0.0
36	vehicle_16	2.306136548521	vehicle_0	2.305928446164	2.306136548521	86.19134	113.485062	2.340034	0.002433	2.340034	0.002433
37	vehicle_13	2.308283813721	vehicle_14	2.30807580265	2.308283813721	153.983911	355.05	0.0	0.0	0.0	0.0
38	vehicle_11	2.308650667588	vehicle_10	2.308442657441	2.308650667588	323.52687	78.65512	0.00268	-1.189297	0.00268	-1.189297

Figure 24: Simulation results. A csv file with the actualized LDM from all the vehicles on the simulation.

## 4 Performance Evaluation. 802.11p and LTE-PC5

This chapter is devoted to comparing the 802.11p and the LTE-PC5 technologies. First, we create a new scenario where we can easily congest the network by increasing its density of vehicles. Then, we modify some main parameters of these technologies so we can compare them under similar conditions. In this case, we focus on comparing the performance of the MAC layer, so we changed some parameters of the PHY layer to achieve a similar coverage on both technologies. Finally, we obtain results and compare the strengths and weaknesses of these technologies.

### 4.1 Highway scenario

To compare these technologies we decided to use the map of a highway, since it allows us to easily increase the number of vehicles in the simulation, and all the vehicles will be driving in the same direction (along the X-axis), which will be more precise to compare the range coverage between technologies.

In this scenario, we defined a bidirectional highway with a length of 5000 m and a total of four lanes, two lanes in each direction. The lanes have a width of 3.2 m, so the whole highway covers 12.8 m. A representation of the scenario can be seen in figure 25, which contains a reduced version of the highway built in the SUMO simulator. And for statistical purposes, we will only analyze the communication done in the central region of the highway, covering only the 3 km in the middle (the vehicles in the extremes of the highway would suffer lower interferences since they are not completely surrounded by other vehicles).

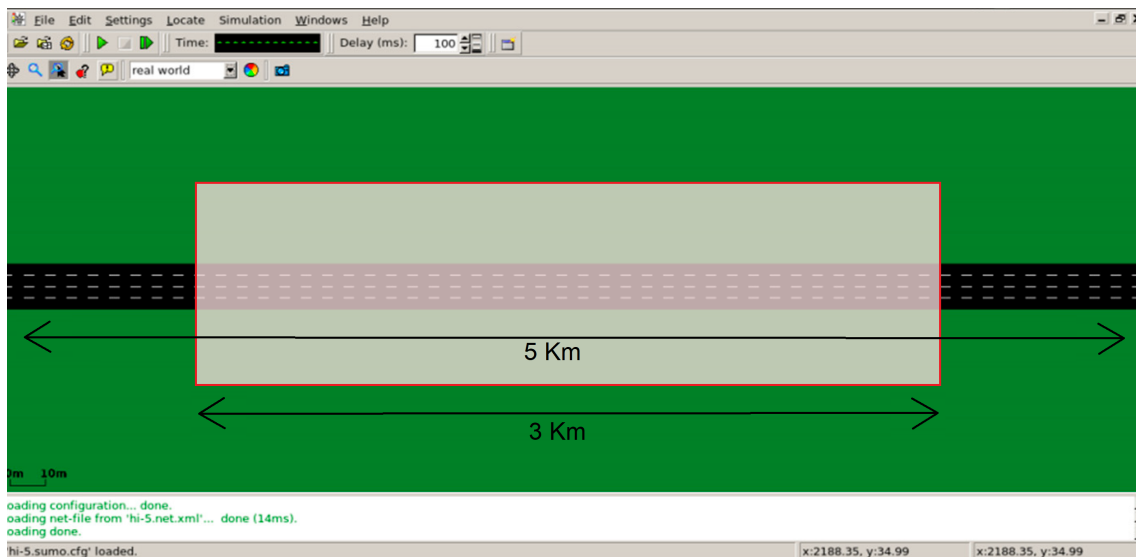


Figure 25: 5 km highway scenario.

And last by not least, to add vehicles to the scenario, we defined some simple routes following the highway. But, in order to distribute the vehicles along the highway, we defined four different spawn points, two in the extremes, and two in the middle of the

highway. Doing that makes it easier to add a high density of vehicles in the simulation, since the cars are added faster, and it produces a more realistic environment.

## 4.2 Simulation Parameters

Once defined the scenario used in the simulations, we can discuss the configuration parameters used for both technologies: IEEE 802.11p, and LTE PC5 Mode 4.

### 4.2.1 Standard Configuration

The first approach taken is to compare the performance of each technology using the configuration parameters defined in the standard. And, considering that our objective is to compare the performance of the MAC layer protocol, the first simulation will be done with only two vehicles. These simulations are done to analyze the capabilities of the PHY layer in a controlled environment without interferences.

### IEEE 802.11p

The main configuration parameters used for 802.11p are the same ones used in Chapter 3, since all the OMNeT++ simulations were done with this technology.

Category	Parameter	Value
Node	Operation mode	802.11p
	Carrier Frequency	5.9 GHz
	Bandwidth	10 MHz
	Channel number	180
	Modulation	QPSK
	Bitrate	6 Mbps
	Transmitter power	23 dBm
	Receiver sensitivity	-85 dBm
	Energy detection	-85 dBm
	SNIR threshold	4 dB
Medium	DCC restriction	false
	Path loss type	Rayleigh Fading
	Path loss factor	3
	Background noise type	IsotropicScalarbackground Noise
Scenario	Background noise power	-110 dBm
	CAM message period	100 ms
	Simulation time limit	1000 s
	Number of vehicles	2

Table 8: OMNeT++ configuration parameters for IEEE 802.11p standard.

All the parameters related to PHY and the MAC layer are defined by the standards and were explained in section 3.2.2. The only significant changes are related to the new scenario, which only has two vehicles. Also, to obtain more values from the simulation we also increased the simulation time, so the vehicles interact more with each other.

## LTE-PC5

On the other hand, implementing the C-V2X technology into the OMNeT++ simulator was more challenging. First of all, we added a new module to the simulator called OpenCV2X [7], which specifically implements the 3GPP standard C-V2X (Rel. 14) Mode 4. This module is an extended version of the SimuLTE OMNeT++ simulator which enables LTE network simulations. Then, once added the OpenCV2X module we replicated the same code as before to extract results from the simulation. The main parameters to be extracted are a collection of all the CAM messages sent and received, which will be used to compute several Key Performance Indicators (KPIs) and compare the technologies used.

In table 9 we gathered the main configuration parameters related to the 3GPP C-V2X Mode 4 standard, which will be used as the basic configuration for all the LTE simulations of this chapter.

The main parameters related to the highway scenario have the same values as the IEEE 802.11p technology. The duration of the simulation is set to 1000 seconds in order to increase the number of interactions with the only two vehicles of the simulation, which will be transmitting CAM messages periodically every 100ms (10 pps). The channel is configured to work at a frequency of 5.91 GHz with a bandwidth of 10 MHz. Moreover, the grid resource is divided into 3 subchannels with 16 RBs per subchannel, resulting in a total of 48 RBs per channel. The packet size of the CAM messages transmitted is 190 B, which is a typical size for short messages in V2X communications. Also, despite the standard allowing transmissions with more than one channel, with these short messages, we decided to limit the number of channels to just one. The transmission power of the vehicle is 23 dBm (as defined in the standard for D2D communications), and the RSSI and RSRP thresholds are set to -92 dBm and -128 dBm respectively.

Now, regarding the modulation used, the standard defines a Transport Block Size (TBS) Index of 7, which corresponds to a modulation of order 2 like QPSK. The value of the TBS Index will limit the number of bits that fit inside a single TB. Since we transmit packets of 190 B (i.e. 1520 bits), we need a minimum of 13 RBs for each TB, which, with a TBS Index of 7, provides us 1608 bits (see Table 1 in [10]), enough to cover the size of the packets. Taking these values into account, the subchannels used have a total of 16 RBs, where 2 of them are used to transmit the SCI, and the TB is composed with 13 RBs more, which only leaves 1 RB to spare. The subchannelization scheme used is adjacent PSCCH + PSSCH, explained in section 2.4.

Category	Parameter	Value
LTE Node	Operation mode	LTE PC5 Mode 4
	Carrier Frequency	5.91 GHz
	Bandwidth	10 MHz
	packetSize	190 Bytes
	subchannelSize	16
	numSubchannels	3
	TBS Index	7
	Modulation	QPSK
	adjacencyPSCCHPSSCH	true
	probResourceKeep	0.8
	RC	[5, 15]
	useCBR	false
	packetDropping	false
	DCC	false
	Transmitter power	23 dBm
	RSRP Threshold	-128 dBm
RSSI Threshold	-92 dBm	
Medium	Path loss type	Rayleigh Fading
	Path loss factor	3
	Background noise type	IsotropicScalarbackground Noise
	Background noise power	-110 dBm
Scenario	CAM message period	100 ms
	Simulation time limit	1000 s
	Number of vehicles	2

Table 9: OMNeT++ configuration parameters for C-V2X LTE Mode 4 standard.

The resource selection is done following the SPS scheme. The Resource Counter is chosen randomly with the standard values between 5 and 15, and the probability to keep the same resource is set to 0.8. Regarding the congestion control mechanisms, like the DCC protocol and the Channel Busy Ratio (CBR), by default they are not used in order to increase the congestion of the network.

### 4.3 Results

This section presents the results obtained from the simulations done with both IEEE 802.11p and LTE-PC5 technologies. All the configuration parameters are explained in



the previous section, and the aim of this study is to analyze the performance of these technologies while varying only the density of vehicles in the simulation. In this case, we computed several simulations with the following number of vehicles in the simulation: 2, 100, 200, and 400. Also, in the cases of 100, 200, and 400 vehicles, we reduced the duration of the simulation, since we already transmitted more messages due to the additional vehicles. Therefore, we defined a setup time of 200 s, which is enough time for the vehicles to get distributed along the whole map, and then run the simulation for 50 s more.

To compare the results, a number of performance metrics had been designed and employed, including the computation of the Packet Error Rate, the distance errors produced between the actual vehicles and the positions sent by CAM messages, and the Neighbor Awareness Ratio.

#### 4.3.1 Packet Error Rate

The Packet Error Rate (PER) is an ITS access layer metric conventionally used to evaluate the link reliability of a network. The resulting PER is computed as the total number of lost packets divided by the total sent packets that have arrived at the receiver with a signal value above the Signal to Noise Ratio (SNR) threshold, so it is computed as follows.

$$PER = \frac{\text{Not received packets}}{\text{Total received packets above the SNR threshold}} \quad (1)$$

The CAM messages sent in the simulator will be correctly received if they reach the MAC layer without error, so they must have surpassed the receiver sensitivity, and energy detection thresholds established in the PHY layer. Otherwise, if any of these thresholds are not achieved, we consider the packet lost.

### Additional Modifications

The first simulations done to analyze the 802.11p and the LTE-PC5 technologies only had 2 vehicles and used the configuration parameters defined by their standards. However, once we analyzed the initial results, we observed a huge difference regards their coverage range. Despite transmitting the same power, with IEEE 802.11p we only achieve a coverage slightly bigger than 100 m, however, with C-V2X we reach a maximum coverage of 1100 m. Also, this difference is further enhanced due to the sensitivity values defined by the standards, since 802.11p sensitivity is 8 dB lower than LTE-PC5. These results can be observed in figure 26, where we computed the PER for both technologies.

As it can be seen, with 802.11p, at around 70 m we already have lost 50 % of the messages, but with LTE-PC5, to lose the same amount of packets we reach a distance larger than 700 m.

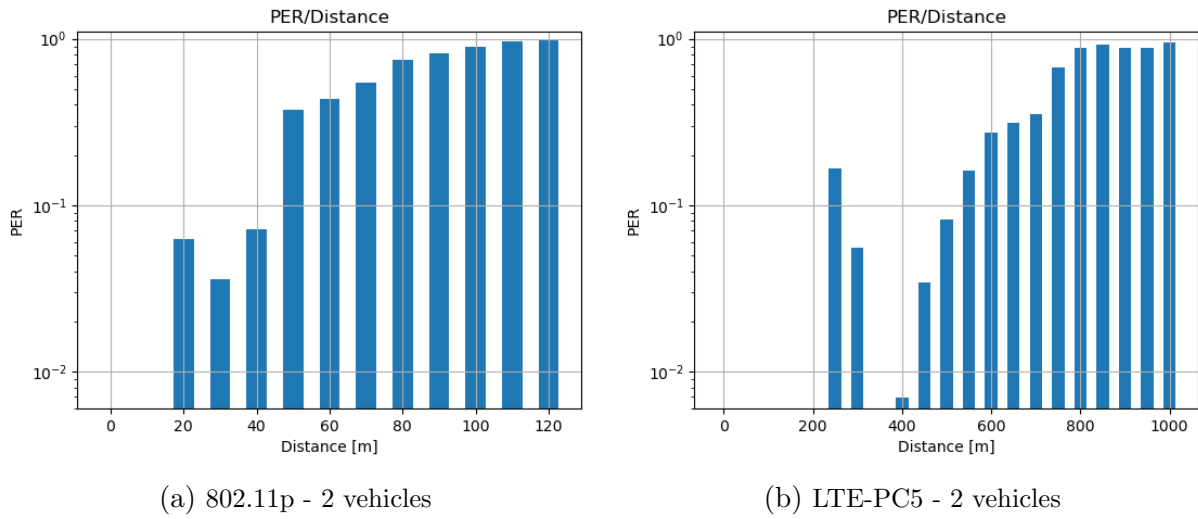


Figure 26: PER results with standard configurations and 2 vehicles.

Seeing these results, we observed that the implementation of the PHY layer of C-V2X is remarkably better than with 802.11p. However, we decided to modify some of the parameters in order to achieve a similar coverage range for both technologies. By doing so, we would remove some limitations of the PHY layer, allowing us to focus on the protocols of the MAC layer. Therefore, to increase the coverage of IEEE 802.11p we increase the transmitted power from 23 dBm to 30 dBm and changed the value of the receiver sensitivity to the same one used in C-V2X. And last, to reduce the coverage of the LTE protocol we decreased its transmitted power from 23 dBm to 13 dBm. Finally, with these modifications, we manage to achieve a similar coverage of 550 m for both protocols.

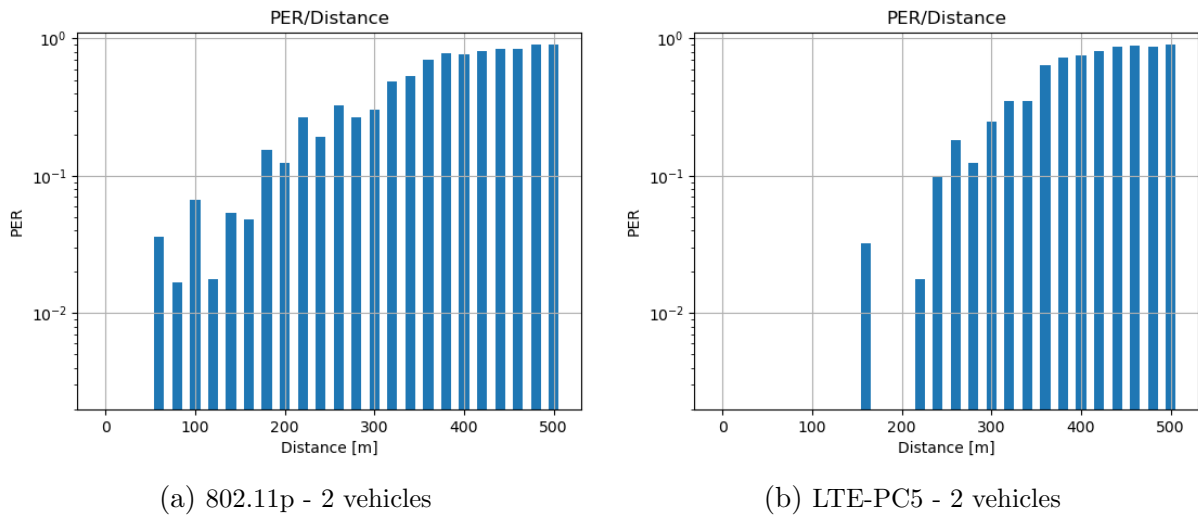


Figure 27: PER results with modified configurations and 2 vehicles.

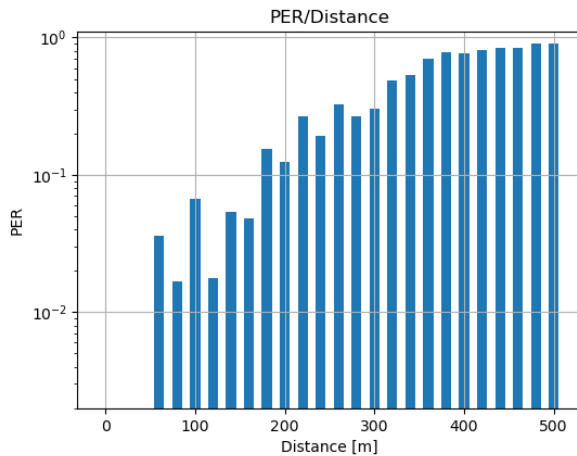


Parameters	IEEE 802.11p		LTE PC5 Mode 4	
	Standard (old)	New	Standard (old)	New
Transmitted Power	23 dBm	30 dBm	23 dBm	13 dBm
Receiver Sensitivity	-85 dBm	-92 dBm	-92 dBm	-92 dBm
Max Coverage	110 m	550 m	1100 m	550 m

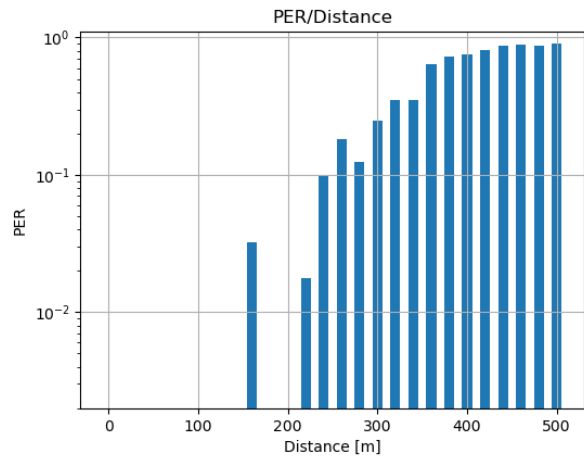
Table 10: Coverage modifications to compare 802.11p and LTE PC5 technologies.

After these changes, we found an optimal configuration of parameters that allows us to analyze the MAC layer of both technologies, avoiding the limitations of the PHY layer. As shown in figure 27, if we use the OMNeT++ configuration of table 10, we manage to reach a really similar coverage range with both 802.11p and LTE-PC5.

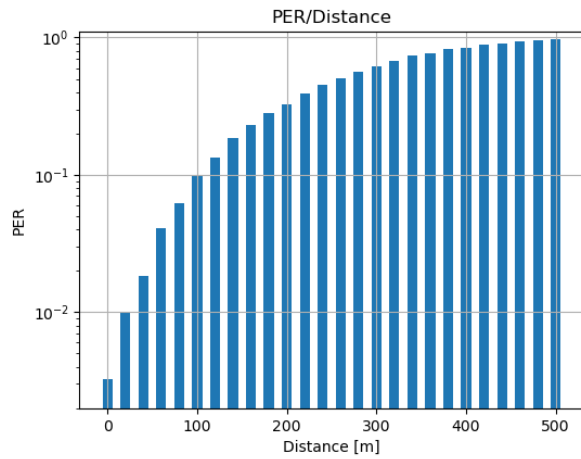
Once modified these parameters, the only value that needs to be changed is the number of vehicles in the simulation, which will be increased until we manage to congest the network. These are the obtained results comparing both technologies with different densities of vehicles.



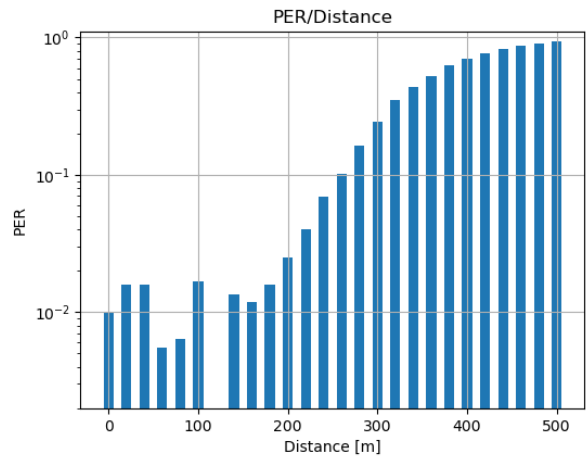
(a) 802.11p - 2 vehicles



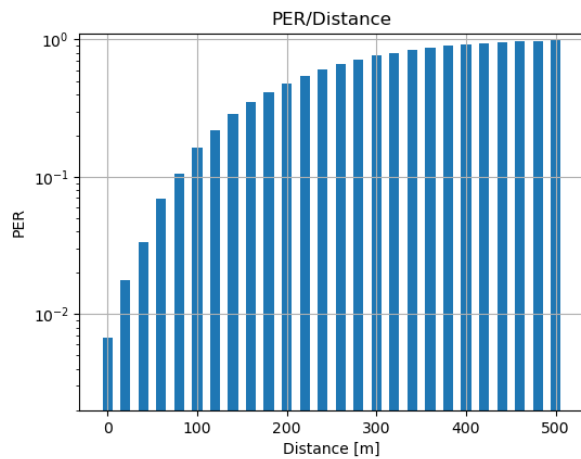
(b) LTE-PC5 - 2 vehicles



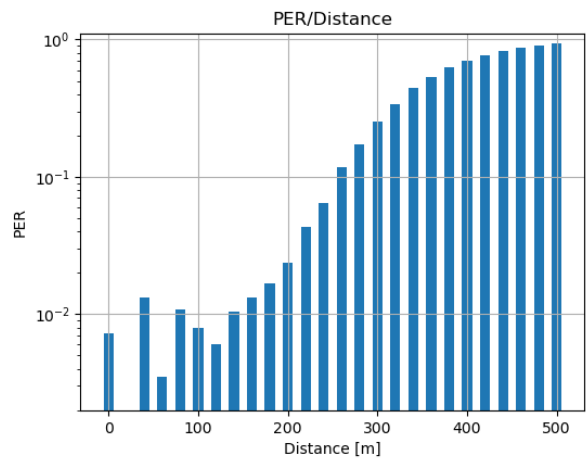
(c) 802.11p - 100 vehicles



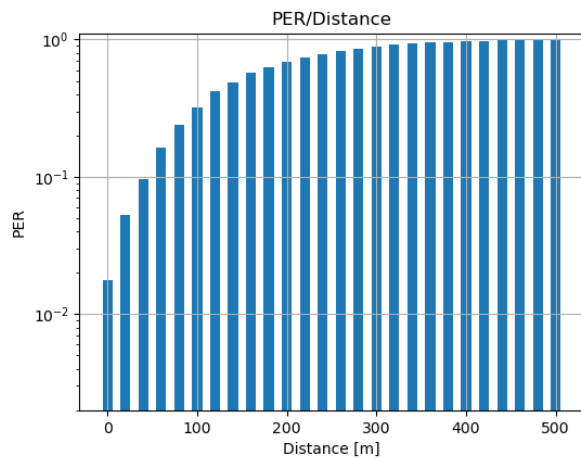
(d) LTE-PC5 - 100 vehicles



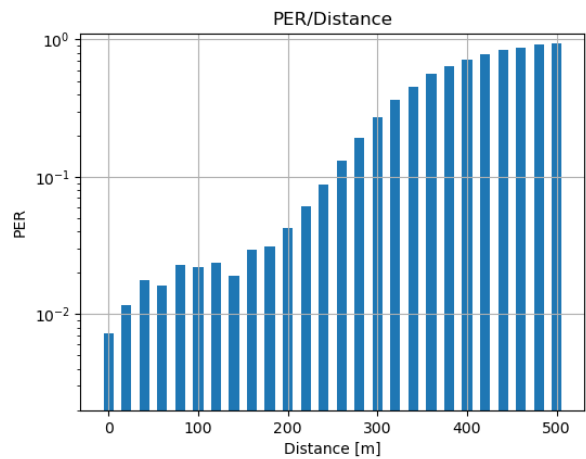
(e) 802.11p - 200 vehicles



(f) LTE-PC5 - 200 vehicles



(g) 802.11p - 400 vehicles



(h) LTE-PC5 - 400 vehicles

Figure 28: PER results.

As can be observed in the generated results, the simulations done with only 2 vehicles are

more irregular than the others, which is due to a lack of samples in those scenarios (there are only 2 vehicles on a highway of 5 km length). This same effect can also be perceived in the lower distances of the LTE-PC5 simulations, specifically in the range below 100 m. In this case, since the vehicles travel through the highway at high velocity, the amount of messages received at that short distance range is notably lower compared to longer distances, where are received the majority of the messages. Also, this irregularity is not shown in the case of LTE-PC5 and 400 vehicles, since we managed to obtain more samples for those shorter distances.

With a high number of vehicles, we observe that the LTE-PC5 performs better than the 802.11p, meaning that the MAC layer manages better the congestion of the network. However, despite the overall number of lost packets is lower with LTE-PC5, it is also more irregular than 802.11p, in particular for lower distances. This effect is mainly explained by the different implementations of the MAC layers on both technologies. 802.11p is more stable in shorter distances since it listens to the channel and only transmits when it is free. On the other hand, LTE-PC5 selects a specific channel resource and uses it on several consecutive transmissions. If two vehicles have randomly chosen the same subchannel, the interference produced will last longer and more packets will be lost.

And as we had expected, if we increase the number of vehicles in the simulation, the congestion of the network increases, and therefore, the resulting PER has slightly lower values, which have decreased uniformly in the whole range of distances. With 802.11p we observe a progressive increase in the PER values. With only 2 vehicles, we lose 10 % of the packets at 200m, but if we congest the network this distance is reduced to 100 m, 80 m, and 60 m, respectively for 100, 200, and 400 vehicles. Nonetheless, with LTE-PC5 that effect is not that clear. With distances larger than 200 m, the PER remains nearly the same, and with shorter distances, there is a little increase in the errors produced, but it is hard to compare due to the irregularity in figures 28d and 28f.

From these results, we could conclude that 400 vehicles are enough to congest the 802.11p network. But the LTE-PC5 technology does better management of the channel resources allowing a higher density of vehicles in the same scenario. As we have seen, there would be needed even more vehicles to completely saturate the LTE-PC5 network.

### 4.3.2 Distance errors

For this KPI we collect the positions errors made between the perceived position of a vehicle (the position sent on the CAM messages), and the actual position of that vehicle. This metric is a significant value to study in V2X communications since it defines the maximum error that a vehicle could have while recording the position of its surrounding vehicles.

Whenever a correct CAM message is received, the vehicle stores the position of the corresponding transmitting vehicle. Because of this, in the time interval between consecutive messages (100 ms), the record of that position won't be updated until the next message is also correctly received. Therefore, the difference between the position sent in the first CAM, and the actual position of the vehicle (right before sending the next CAM), is what we have defined as a "distance error", which is also represented in figure 29. So, usually,

that distance will be the displacement of the transmitter vehicles during the 100 ms taken to transmit another CAM, which defines a strong correlation between this metric and the velocity of the vehicles. Moreover, that distance error could also be increased if some of the following CAM messages were lost, increasing the error each time a CAM message is lost. As an example, if two consecutive messages are lost, but the third is correctly received, the total distance error is the difference between the actual position of the vehicle (before sending the third CAM), and its position sent 3 messages earlier, 300 ms ago. And if that effect is repeated, we could reach quite a large distance error, potentially affecting some safety applications.

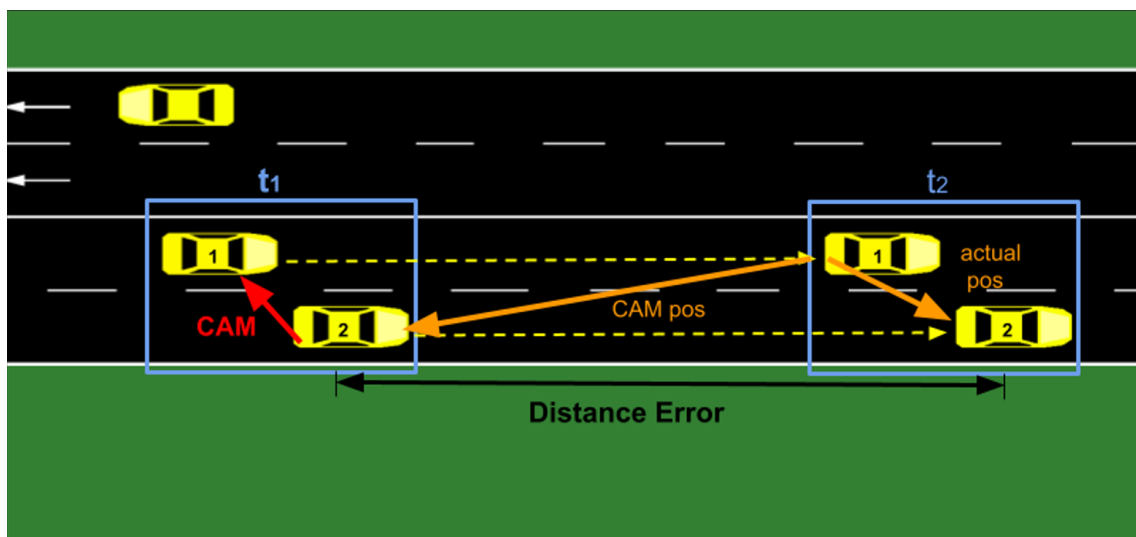
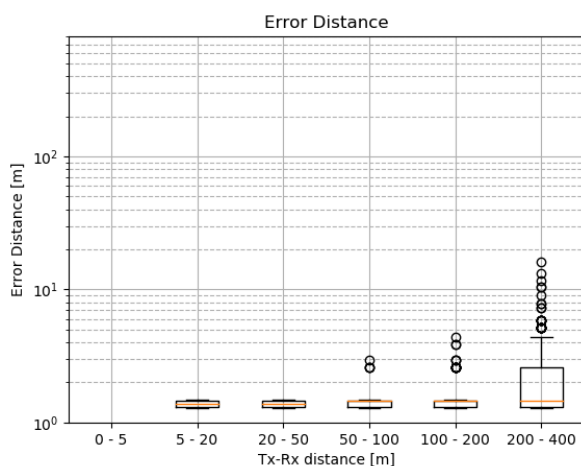
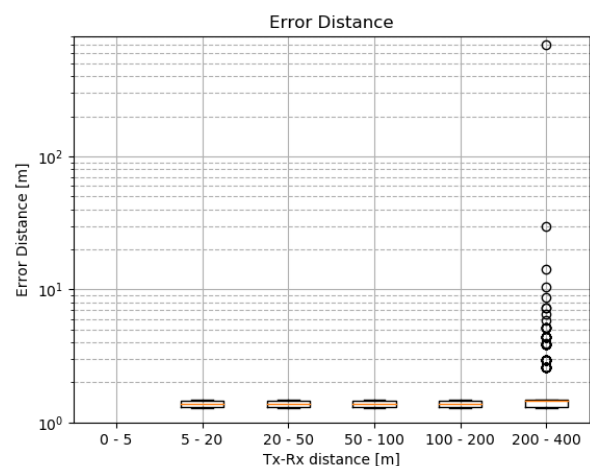


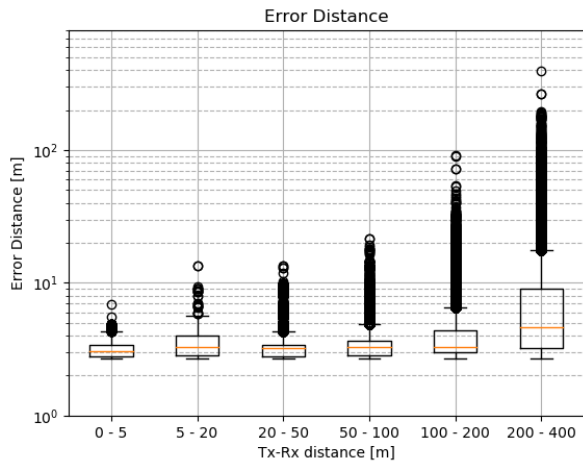
Figure 29: Error distance perceived by vehicle 1 before receiving the 2n CAM from vehicle 2.



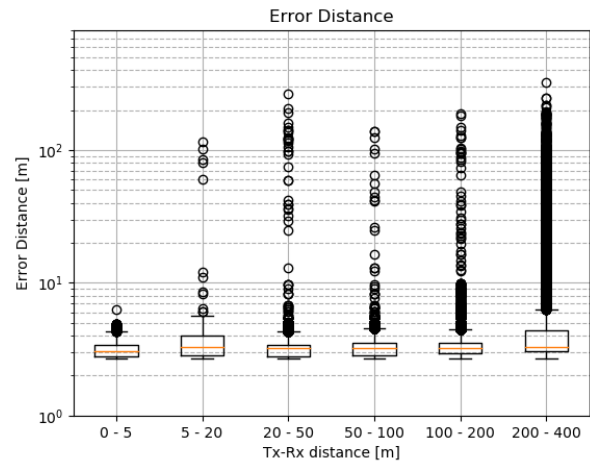
(a) 802.11p - 2 vehicles



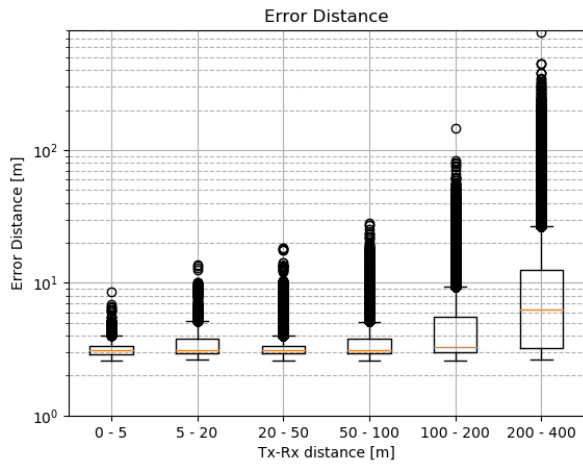
(b) PC5 - 2 vehicles



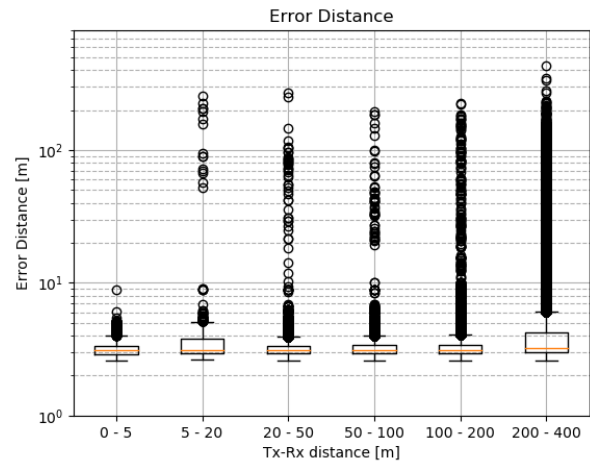
(c) 802.11p - 100 vehicles



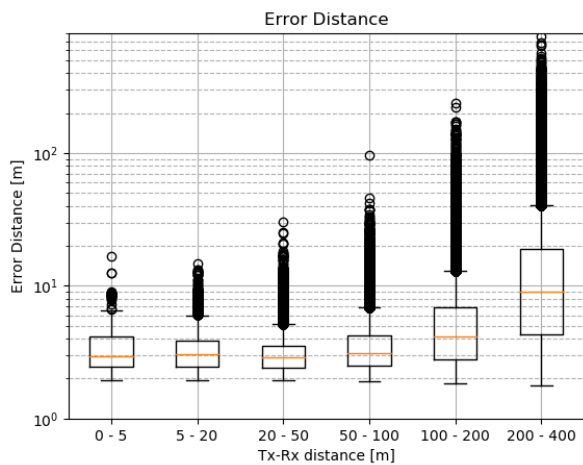
(d) PC5 - 100 vehicles



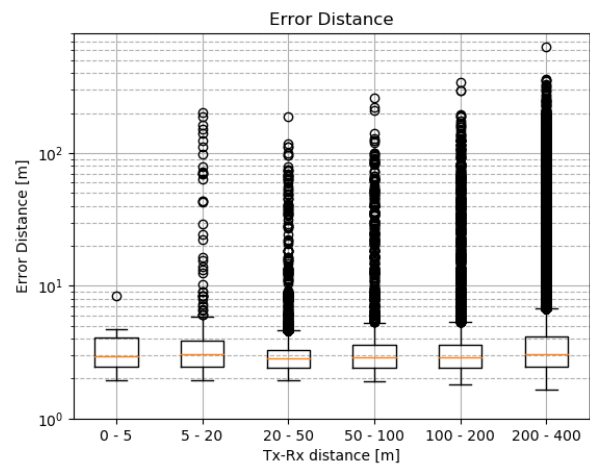
(e) 802.11p - 200 vehicles



(f) PC5 - 200 vehicles



(g) 802.11p - 400 vehicles



(h) PC5 - 400 vehicles

Figure 30: Distance error results.

Analyzing these results, we observe that with only 2 vehicles in the simulation, their distance errors are extremely low, corresponding to the minimum displacement of the vehicles in a 100 ms interval (frequency transmission of CAM messages). And regarding the higher outliers (the values with a higher distance error shown above the boxes), we observe that they only appear when the vehicles are too far apart, at a distance greater than 200 m. In this case, the values with 802.11p have a larger variance and smaller outlier errors. However, with LTE-PC5 and large distances, we maintain a lower variance with the majority of the values, but their outliers are much larger.

To analyze the results of the distance errors, it is crucial to consider the velocity of the vehicles, since larger velocities will translate into greater distance errors. In this case, the scenario used is a straight-line highway which allows the vehicle to drive at elevated velocities. In the simulation, the exact velocity of the vehicles can vary depending on their surroundings. If a vehicle pretends to overtake another one, it will speed up, but if it is enclosed among other vehicles, it will decelerate. Despite that, the average velocity of the vehicles is 105 km/h (29.4 m/s). Consequently, if CAM messages are sent every 100 ms, the typical minimum distance error achieved would be 2.94 m, corresponding to the majority of samples taken in our simulations.

Now, if we analyze the simulations done with a large number of vehicles, we observe that the 802.11p is more stable, and the distance error increases progressively with the distance between vehicles. Also, as it happens in the case of two vehicles, the variance of the errors is bigger for longer distances, but now, some outliers appear for shorter distances. On the contrary, with LTE-PC5, we achieve a great reduction of the variance of the errors (all the mean values are close to 3 m, matching with the minimum distance error computed for this scenario). Nevertheless, we also observe an increase in all the outliers independently of the distance. With higher distances, there are significantly more outliers than with shorter distances, but all of them reach really high values. With LTE-PC5 we can find a maximum distance error of 200 or 300 m on really short distances, however, with 802.11p those values are only achieved for long distances. It is really important to analyze these effects, since a large distance error between vehicles that are really close to one another, is critical for any safety application. Consequently, despite having fewer outliers with LTE-PC5, their higher values could be a critical problem if they happened at shorter distances.

This particular behavior is the result of the different implementations of the MAC layer in both technologies. 802.11p uses CSMA/CA, which tries to avoid collisions by sensing the channel and transmitting only when it is idle. Therefore, the transmitter needs to check the status of the channel after any transmission, and if the channel is highly congested, it is easier to keep increasing the errors, due to longer distances or a huge amount of vehicles. On the other side, LTE-PC5 uses the sensing-based SPS scheme, which first senses the channel for a long time (a Sensing Window of 1000 subframes), and then it assigns a channel resource that will be used on several consecutive transmissions (defined by the RC). This approach makes a more efficient selection of the resources used, which helps to reduce the congestion and the errors produced. Even so, a drawback of that system is the rare cases when two vehicles choose to transmit in the same subchannel, which produces interference that could last several transmissions. These worst cases are the outlier values

that can be seen in the short distances of the LTE-PC5 simulations, which are also more critical for safety applications.

To sum up, this metric is really useful to analyze the effects of the MAC layer in both technologies. As we have seen, 802.11p provides us with better results while transmitting at close distances, however, LTE-PC5 manages to reduce the overall errors on longer distances.

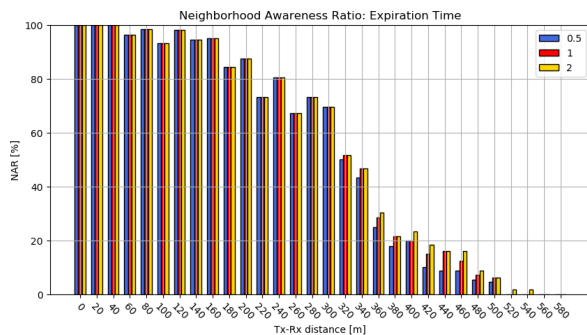
### 4.3.3 Neighborhood Awareness Ratio

The Neighbor Awareness Ratio (NAR) is a measure of the cooperative awareness achieved by the vehicles in a scenario. This statistic is taken from all the vehicles in the simulation every 100 ms (the transmission frequency of the CAM messages), and is computed as follows.

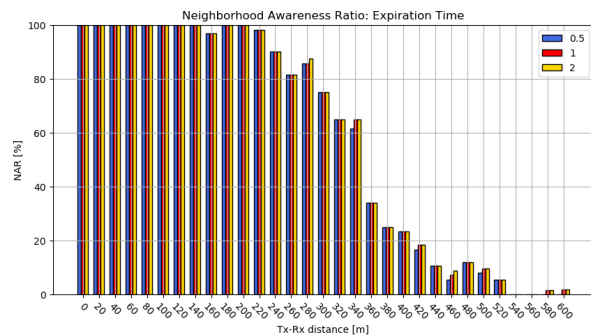
$$NAR = \frac{\text{Perceived number of vehicles}}{\text{Actual number of vehicles}} \times 100 \tag{2}$$

Once a vehicle correctly receives a CAM message, it stores the position of the transmitter vehicle in a database and adds that vehicle to a list of neighbors which are in range. While receiving messages from that vehicle, it will remain in the list of neighbors, but if a defined expiry time is passed without receiving new messages, it will be removed from that list. For instance, if the expiry time is defined as 2 s, the received would have to miss 20 CAMs in order to remove the vehicle from the list. Therefore, the NAR is computed by dividing the perceived number of vehicles (number of elements in the list), by the actual number of vehicles, defined as all the vehicles in the sensitivity range (close enough to surpass the receiver sensitivity threshold).

For the analysis of this metric, we decided to compare the performance of several expiry values, so we used the following time intervals: 0.5 s, 1 s, and 2 s.

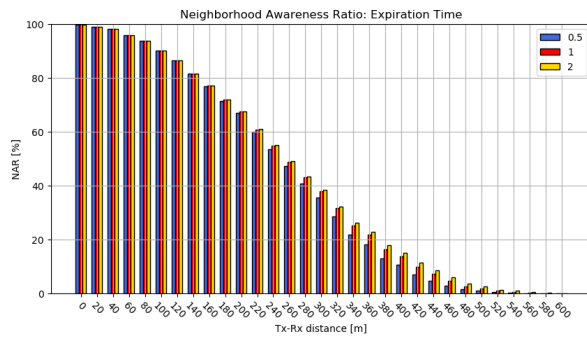


(a) 802.11p - 2 vehicles

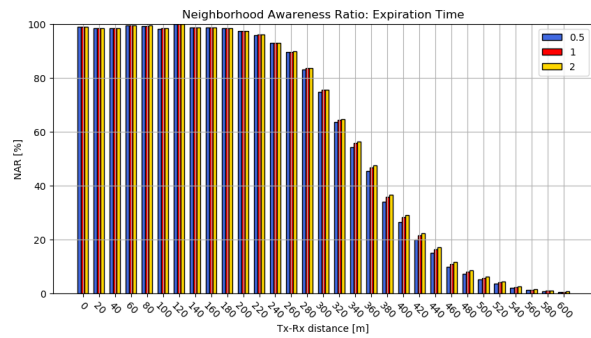


(b) PC5 - 2 vehicles

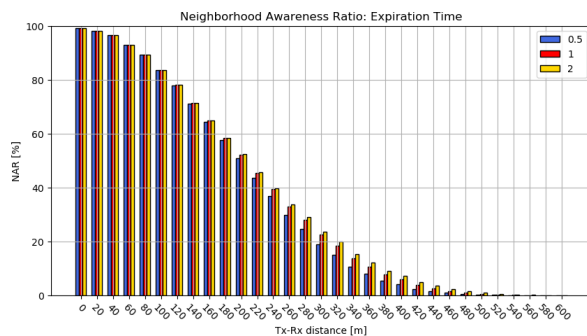




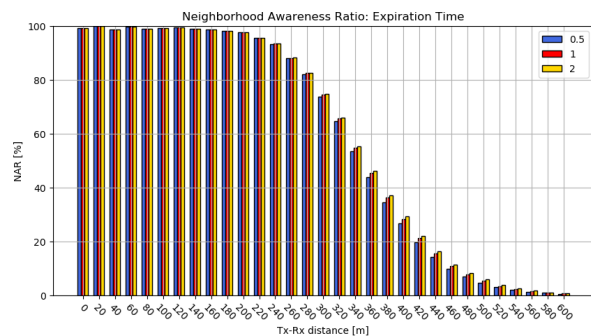
(c) 802.11p - 100 vehicles



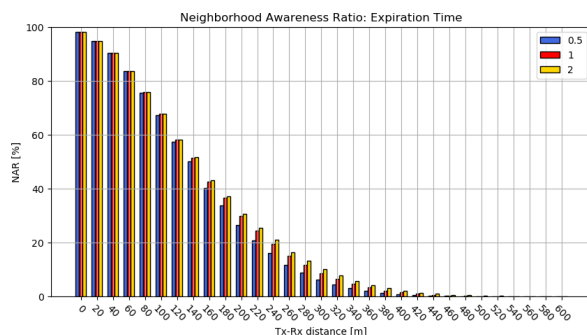
(d) PC5 - 100 vehicles



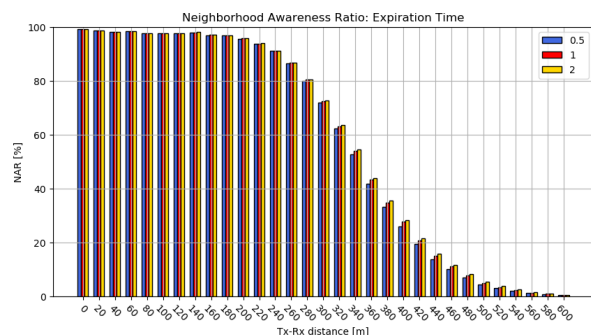
(e) 802.11p - 200 vehicles



(f) PC5 - 200 vehicles



(g) 802.11p - 400 vehicles



(h) PC5 - 400 vehicles

Figure 31: NAR results.

These results are similar to the PER results. With the PER we focused on studying the lost packets inside a sensitivity range, while the NAR is centered in collect the correctly received messages inside that sensitivity range (with the addition of the expiry time to define the neighbor awareness).

As seen in the previous sections, the results with only 2 vehicles tend to have higher values (caused by a low congested network), but there are more irregular due to a lack of samples. Also, we can not see any significant variation regards the different expiry values used. Since nearly all the transmissions done with lower distances are always correctly received, the changes in the expiry time don't have any effect.

Now, analyzing the results of a high density of vehicles, we clearly observe how the con-

gestion of the network increases with the number of vehicles, since the graphs tend to have a lower slope. Also, this reduction of the NAR is more relevant on the 802.11p, where the values tend to decrease more rapidly. If the number of vehicles is increased from 100 to 400, the required distance to achieve a NAR over 80 % is reduced from 150 m to 60 m. However, as seen in the PER analysis, the NAR with LTE-PC5 remains essentially the same independent of the number of vehicles. It maintains a NAR over 80 % for distances up to 280 m, in all simulations. Consequently, we come to the same interpretation as section 4.3.1, the MAC protocol from LTE-PC5 does better management of the channel resources when the network is congested, allowing it to support a scenario with more vehicles.

Despite obtaining a narrow increase in the NAR at longer time intervals (especially seen in middle and longer distances), we don't observe any relevant effect. As we expected, if we allow a longer time for the vehicles to remain in the list of neighbors, it is easier to receive a correct packet after several lost ones, which increases the NAR value. However, these cases are extremely unusual, so when we start losing packets, it will be rare to correctly receive a new one after a short time. Therefore, using expiry times between 0.5 and 2 s, does not really make a difference in this scenario.

## 5 Conclusions and future development:

After an elaborate study of several simulators related to V2X technologies, we managed to comprehend the process needed to create a personalized scenario in SUMO, and all the network elements involved in an OMNeT++ simulation. Also, we managed to introduce two different technologies inside OMNeT++: IEEE 802.11p and C-V2X LTE-PC5, which have been the basis for this thesis.

The focus of this project is to analyze and compare the performance of the MAC layer of two principal V2X technologies. One of the main advantages of the C-V2X compared to the 802.11p are its improvements in the PHY layer, which manages to transmit packets nearly ten times further than 802.11p. For that matter, we decided to modify some configuration parameters to ignore the limitations of the PHY layer and define several KPIs to analyze its performance in the MAC layer, where both technologies had the same coverage conditions.

With the results obtained from the PER and the NAR performance metrics, we conclude that LTE-PC5 is highly effective in selecting the channel resources of a congested network, minimizing the possible interferences between vehicles. The errors produced in 802.11p increase gradually while increasing the density of vehicles. However, LTE-PC5 succeeds in balancing the network load and reducing the errors produced on shorter distance ranges (under 200 m). That allows LTE-PC5 technologies to support scenarios with higher densities of vehicles. Also, the NAR results offer a visual representation of the awareness achieved by each technology. In this case, both technologies have a similar performance with a low density of vehicles, but when the network is congested, 802.11p worsens rapidly.

With these initial metrics, LTE-PC5 outperforms 802.11p in nearly all features. Nevertheless, we obtained some interesting results from the distance error metric. In that case, 802.11p manages to maintain a really low error on close distances (up to 20 or 50 m), which then increases progressively in larger distances. On the other hand, LTE-PC5 has a really low mean error, but there are some unusual cases when the errors reach extremely large values, both at shorter and longer distances. The main drawback is that the distance errors produced on shorter distances are critical values that could endanger the safety of the vehicles. Therefore, in the framework of safety applications, 802.11p could be a better solution, despite their limitation in the PHY layer. This effect can be explained due to the different MAC layer mechanisms used in both technologies. Since 802.11p uses CS-MA/CA algorithm to access the channel resources, it is easier to correctly transmit a packet at close distances, which will be transmitted once the channel is sensed idle. However, LTE-PC5 uses a sensing-based SPS scheme that first senses the channel for a long time (a Sensing Window of 1000 subframes), and then it assigns a channel resource that will be used on several consecutive transmissions. For this reason, in the rare cases when two vehicles randomly decided to transmit at the same sub-channel, that communication will suffer continuous interferences until one of the vehicles finished its RC, and reselects a new resource with probability  $1 - P_k$ . So, during that time, several consecutive CAM messages will be lost, producing those high values of distance errors.

After seeing these results, there are a lot of recommendations for continuing with this research study. To obtain optimal results from the KPIs used, the simulation time could

be increased in order to reduce some of the irregularities shown in the shorter distances of the LTE-PC5, specifically in the PER results. Another option would be to increase even more the density of vehicles in the simulation in order to find the maximum load that could be supported with LTE-PC5 technology (since 400 vehicles did not seem enough to fully saturate the channel). In addition, despite these few changes, it would be interesting to analyze these same metrics in a different scenario. A Manhattan grid or some urban scenario could be a good option to test the presence of buildings, which would add extra attenuation to the messages.

And finally, a last idea that could be approached would be the addition of other technologies into the analysis. For instance, the later releases of the 3GPP are developing the standard 5G New Radio, which aims to use 5G technologies focused on mobility enhancement applications. This topic is highly related to vehicular communications, and could be the starting point to add 5G technologies into the OMNeT++ simulator.

## References

- [1] Daniel Jiang and Luca Delgrossi. Ieee 802.11p: Towards an international standard for wireless access in vehicular environments. In *VTC Spring 2008 - IEEE Vehicular Technology Conference*, pages 2036–2040, 2008.
- [2] Release description; release 14 (v14.0.0, release 14). Standard, 3rd Generation Partnership Project (3GPP), June 2018.
- [3] Intelligent transport systems (its); part1: Functional requirements. Standard, European Telecommunications Standards Institute (ETSI), 2010.
- [4] Intelligent transport systems (its); geonetworking; part 1: Requirements. Standard, European Telecommunications Standards Institute (ETSI), 2014.
- [5] Elena Lopez-Aguilera, Eduard Garcia-Villegas, and Jordi Casademont. Evaluation of ieee 802.11 coexistence in wlan deployments. *Wireless Networks*, 25(87), 2019.
- [6] CARLA main web page. [https://carla.readthedocs.io/en/latest/ref\\_sensors/](https://carla.readthedocs.io/en/latest/ref_sensors/). Accessed: 2022-10-15.
- [7] Open Cellular Vehicle To Everything - Mode 4 main web page. <http://www.cs.ucc.ie/cv2x/>. Accessed: 2022-11-18.
- [8] Alessandro Bazzi, Giammarco Cecchini, Michele Menarini, Barbara M. Masini, and Alberto Zanella. Survey and perspectives of vehicular wi-fi versus sidelink cellular-v2x in the 5g era. *Future Internet*, 11(6), 2019.
- [9] Intelligent transport systems (its); vehicular communications, basic set of applications, part 2: Specification of cooperative awareness basic service. Standard, European Telecommunications Standards Institute (ETSI), 2019.
- [10] Leandro Miguel Lopez, Charmae Franchesca Mendoza, Jordi Casademont, and Daniel Camps-Mur. Understanding the impact of the pc5 resource grid design on the capacity and efficiency of lte-v2x in vehicular networks. *Wireless Communications and Mobile Computing*, 2020.