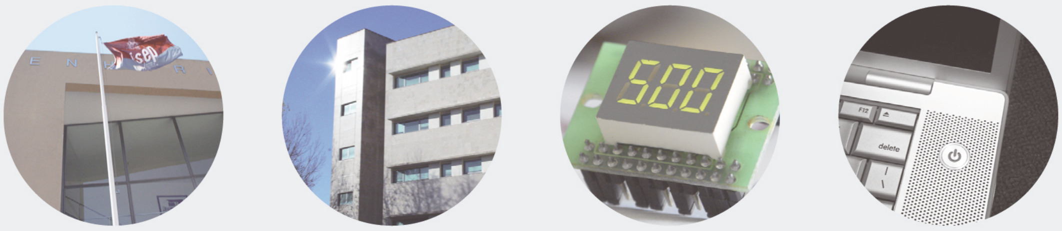


A simulation approach for increased safety in advanced C-ITS scenarios

BRUNO FILIPE BARROS VIEIRA

novembro de 2019

ISEP

Instituto Superior de Engenharia do Porto

A simulation approach for increased safety in advanced C-ITS scenarios

Master Thesis

To obtain the degree of master at the
Instituto Superior de Engenharia do Porto,
public defend on November by

Bruno Filipe Barros Vieira
1140464

Degree in Electronics and Computer Science - Automation and Systems
Porto, Portugal.

Supervisor:

Prof. Dr. Ricardo Severino

ISBN ++-++++-+++-+

Author email: 1140464@isep.ipp.pt

Acknowledgments

Not disregarding at all any of my personal commitment and work ethics, the last five academic years of my life wouldn't be possible to go so well if it wasn't for some really important people's roles in them. This thesis, finishes perfectly this cycle and hopefully starts a new one, with many more challenges ahead. From all the people that participated one way or another in my life path during this cycle, I can't let away this opportunity without naming some of the more impactful ones.

First of all, I have to acknowledge my supervisor, Dr. Ricardo Severino, for inviting me onto CISTER and have the chance to be part of amazing research projects, for all the roles he had to fulfill during this time, as well as, to all the knowledge and insight provided.

Would like to thank all the people from CISTER, for their support and compassion with one another, and for clearly showing that really great quality research gets carried out in this Research Center.

A special sign of gratitude to Nuno Guedes and Daniel Almeida, two amazing friends met during this last five years that were always a great support during all this time, and particularly, during the last months when our work was all carried out at CISTER, concluding in, hopefully, three excelent Master's thesis.

I'm not able to finish this without giving the most important acknowledgment to my family, but specially, to my parents. Words, sometimes, fail me to properly say how important their roles were and still are in my life, however, one thing I'm sure, if it wasn't for their incredible and almost heroic effort through all their life, nothing I am and can become would be possible. Obrigado!

Resumo

Com os recentes desenvolvimentos em diferentes áreas de conhecimento, como redes de comunicação sem fio e sensores, bem como a evolução recente em vários tópicos na área da Computação, os Sistemas Inteligentes e Cooperativos de Transporte (C-ITSs) tornaram-se um tema muito importante, e espera-se que comecem a ser cada vez mais implementados num futuro próximo.

Nesta tese, é feita uma análise sobre estes sistemas e diferentes possiveis cenários focando no cenário de Platooning, assim como sobre comunicações Veículo-a-Tudo (V2X) com foco no ETSI ITS-G5, o standard europeu mais amplamente aceite na indústria automóvel para este tipo de comunicações.

O desenvolvimento de duas ferramentas de co-simulação para análise de cenários C-ITS usando comunicações veículo para veículo (V2V), foi feito no contexto desta tese. COPADRIVe é uma ferramenta de co-simulação que junta um simulador de rede e um simulador robótico. A outra ferramenta de co-simulação, é uma ferramenta hardware-in-the-loop que úne um simulador robótico com comunicações feitas através de hardware real, On-Board units (OBUs). Ambas as ferramentas foram desenvolvidas e usadas como forma de análise e teste de situações de Platooning e componentes de software para implementação neste tipo de cenários. Este desenvolvimento teve origem na necessidade de existência deste tipo de ferramentas para suporte dos desenvolvimentos feitos no contexto dos Projetos europeus de I&D SafeCOP e ENABLE-S3, onde o CISTER participava ativamente.

Keywords— Sistemas Inteligentes e Cooperativos de Transporte , Comunicações inter-veiculares, Pelotões de veículos , Simulação de Rede, Simulaçao Robótica, ROS

Abstract

With the developments in different areas like Wireless Communication Networks and sensors, as well as, the recent evolution on various topics on Computing, Cooperative Intelligent Transportation Systems(C-ITSs) became a hot topic for research, and are expected to be increasingly deployed in the future.

From the different possible scenarios, in this thesis, we focus in analyzing Cooperative Platooning and particularly, in enabling a set of simulation tools capable of encompassing the supporting Vehicle-to-Everything(V2X) communications guaranteed by the ETSI ITS-G5, the most widely accepted European standard on the automotive industry for these kind of communications.

Therefore this thesis presents the development of two co-simulation tools for analysis of C-ITS scenarios using Vehicle-to-Vehicle(V2V) communications.

First, COPADRIVe is a co-simulation tool joining together a network simulator and a robotic simulator. The other co-simulation tool, uses a a hardware-in-the-loop approach one bridging a robotic simulator with real communications via On-Board-Units (OBUs). Both tools were developed and used as the means to test and analyze Platooning scenarios and software components relevant in such applications, importantly. These tools' were developed in line with the R&D European Projects SafeCOP and ENABLE-S3, where CISTER was and active participant.

Keywords— Cooperative Intelligent Transportation Systems, V2V Communications, Vehicular Platooning, Network Simulation, Robotic Simulation, ROS

Contents

List of Figures

List of Tables

Acronyms

API	Application programming interface
ASN.1	Abstract Syntax Notation One
AU	Aplication Unit
BSA	Basic Service Application
BSP	Basic Service Profile
C-ITS	Cooperative Intelligent Transport System
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CCH	Control Channel
CLW	Control Loss Warning
CPS	Cyber-Physical Systems
DCC	Decentralized Congestion Control
DENM	Decentralized Environmental Notification Message
DSRC	Dedicated short-range communications
ETSI	European Telecommunications Standards Institute
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
HIL	Hardware-in-the-loop
HMI	Human Machine Interface
IEEE	Institute of Electrical and Electronics Engineers
ITS	Intelligent Transportation System
LDM	Local Dynamic Map
LiDAR	Light Detection And Ranging
LOA	Loss of Acceleration

LOB	Loss of Brakes
LOD	Loss of direction
LTE	Long Term Evoolution
MAC	Medium access control
MANET	Mobile ad hoc network
NMEA	National Marine Electronics Association
OBU	On-board Unit
OS	Operating System
ROS	Robotic Operating System
RSU	Road-Side Unit
RTM	Runtime Monitor
RT	Runtime
SSH	Secure Shell
SUMO	Simulation of Urban Mobility
UDP	User Datagram Protocol
UI	User interface
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
VANET	Vehicle ad hoc Network
VDP	Vehicle Data Provider
VP	Vehicular Platooning
WAVE	wireless access in vehicular environments

1 Overview

1.1 Introduction

Modern embedded systems, coupled with the advancements on digital communication technologies, have been enabling a new generation of systems, tightly interacting with the physical environment via sensing and actuating actions: Cyber Physical Systems (CPS). These systems, characterized by an unprecedented levels of ubiquity, have been increasingly relying upon wireless communication technologies to provide seamless services via flexible cooperation. As these Cooperating CPS (Co-CPS) reach into safety-critical application domains (e.g. automated vehicles platooning in the automotive domain), safety becomes a crucial topic that must be carefully analyzed.

As in some areas of application of CPS's, such as in aviation or automotive, the initial phase of technology development poses a significant toll in terms of cost and risk, there is an increasing interest in depending on simulation tools.

This is clearly the case of cooperative ITS's such as cooperative platooning, in which, automated vehicles cooperate to follow each other during a certain path, with different advantages, according to [1] vehicles on a single line can reduce fuel consumption by up to 20% for non-leading vehicles. Besides, the use of a leader can greatly reduce the need for V2I information exchange for trajectory tracking since only the leader would need to receive the data regarding the route while the others would only receive the follow-up information from the leader.

As autonomous vehicles start to become much more relevant, and regarded as the future of transportation, it is important to notice the complexity of it, as well as, all the implications of these kind of future implementations. The deployment of human-free driving vehicles requires not only for their intra-vehicle systems to be well-tested and safety assured but, also, to have a whole lot of systems deployed all around cities and highways, in order to make sure that these vehicles are not able to cause any kind of harm to neighbour vehicles or even pedestrians.

The study of Platooning advances in several areas, such as control models, V2V and V2I communication [2], energy consumption, safety, interaction with other vehicles and platoons, among others. In addition, their modeling demands knowledge of real conditions so that systems can be simulated accurately before being implemented in the real world. Thus advanced simulation tools that can mimic real sensors and actuators, control models and networks, must be deployed to better evaluate performance and detect the safety limits of such cooperative scenarios.

To address these challenges it is important to carry out a tough analysis of these systems regarding different aspects such as control and communications, using simulation is the most efficient way to go for these tests since automotive scenarios that are very critical in terms of safety are very cost-heavy.

In this thesis, we address these issues by developing two co-simulation frameworks, bridging the capabilities of robotic simulators for control and network simulators for vehicular communications support. Creating a toolset for a more comprehensive and inclusive analysis of these systems.

1.2 Research Context

This thesis' proposal emerged from the SafeCOP european project, where CISTER/ISEP was a participant, under the use case 3, control loss warning. The aim of this work was to develop new tools and technologies for increased safety that could be applied to future cooperative autonomous functions in vehicular scenarios. COPADRIVe[3][4] appears as a simulation tool that is able to evaluate communications impact on platooning scenarios, while the HiL simulation tool was a cooperative work between CISTER and GMV Skysoft SA in an attempt to provide a simulation framework that was able to test out safety software modules for a cooperative platooning scenario depending on real communication hardware.

1.3 Research Objectives

The main objective of this thesis is to provide technologies that can support the analysis of the impact of communications on cooperative Platooning scenarios, as well as, validate safety mechanisms (e.g Control Loss Warning). Implementation and test results will be shown as well, as a way, to demonstrate this tools' capabilities and possible future deployments.

1.4 Research Contributions

In this thesis the main contributions are:

- Overview of different Cooperative Intelligent Transportation System (C-ITS) scenarios, focusing on the way vehicular communications can take part in them.
- Implementation and adaptation of a Cooperative Platooning control model fully-based on Vehicle-to-vehicle (V2V) communications.
- Overview of the main ITS communications standard for Europe (ETSI ITS-G5) and its adequancy for supporting Cooperative Platooning scenarios.
- Development of a Co-simulation tool(COPADRIVe), integrating a robotic (Gazebo) and a network simulator (OMNeT++) over ROS[3][4].
- Development of a Hardware-in-the-loop simulation framework that brings together a robotic simulator (Gazebo) and a ITS communications On-Board Unit (Cohda's MK5 OBU).
- Performance analysis of different ETSI ITS-G5 communications stack settings and its impact upon Cooperative Platooning control using COPADRIVe[3][4].
- Validation of a Control Loss Warning (CLW) mechanism for increased safety on Platooning scenarios, using the Hardware-in-the-loop simulation tool.
- Writing and publishing of two, already accepted, conference articles [3][4], appendixes A and B, respectively.

1.5 Thesis Structure

In the remaining of this thesis, Chapter 2, will focus on different C-ITS scenarios and in particular on cooperative platooning by looking into its advantages and challenges.

Vehicular communications will be overviewed in Chapter 3 in regards to V2X communications, ETSI ITS-G5 and IEEE 802.11p will be focused, it is the current European baseline for enabling ITS communications.

A state-of-the-art analysis regarding different kinds of simulators and frameworks is done in Chapter 4, as a way to establish the background for the COPADRIVe simulation tool.

Under Chapter 5, the development of the COPADRIVe[3][4] simulation tool will be presented and analyzed.

In a similar fashion, Chapter 6 will provide a clear overview on the development of the Hardware-in-the-loop simulation framework.

Simulation Results regarding the two developed tools, previously mentioned, will be shown at Chapter 7, along with some simulation conclusions.

Finally, this Thesis closes with Chapter 8 where conclusions are withdrawn , as well as, some thoughts about future work and developments related to these projects.

2 Cooperative Intelligent Transportation Systems Scenarios - Cooperative Platooning

Cooperative Intelligent Transport Systems (C-ITS) must rely heavily on communications and data exchange between different peers, in order to, guarantee safety mechanisms during all the different and complex scenarios where they can be involved.

With this in mind, in this section, some examples of possible C-ITS scenarios will be shown. Only C-ITS scenarios that can or have already been using vehicular communications in their safety assurance models, will be presented. Most of this next subsection will be focused on [5], considering its insights about many different scenarios already over viewed by automotive experts, as scenarios that can and will require vehicular communications.

Then the Platooning scenario will be focused as the implemented and evaluated scenarios on both of the developed simulation frameworks. In this subsection, a particular V2V communications fully reliant platoon following control model will be presented and detailed.

2.1 Scenario Examples

Following [5], and as previously mentioned, C-ITS scenarios/applications may be split all over three major groups:

- Co-operative road safety.
- Traffic efficiency.
- Other applications/scenarios.

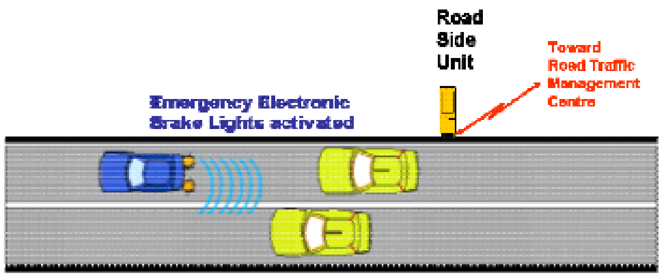
On this section, some examples, that feature the each of the above groups, will be presented with a short description of each other and a graphical representation, if relevant.

2.1.1 Co-operative road safety scenarios

Co-operative road safety scenarios implement mechanisms that provide help to maintain and guarantee safety to both pedestrians and vehicles' passengers. Three examples are described:

Emergency electronic brake lights - Supports signaling from any vehicle of its hard breaking to its local followers. In such case, the hard braking triggers the switch of emergency electronic brake lights.(Figure 1)

Figure 1: *Emergency electronic brake lights use case scenario [5]*


Road Side Unit
Toward Road Traffic Management Centre
Emergency Electronic Brake Lights activated

Safety function out of normal condition warning - Supports signaling of steering, braking, etc... abnormal safety condition to others. (at 6.2, a similar application is tested out and analyzed)

Overtaking vehicle warning - An overtaking (passing) vehicle signals its action to other local vehicles to secure the overtaking situation.

2.1.2 Traffic efficiency scenarios

Traffic efficiency scenarios are use-cases where C-ITSs help to improve traffic efficiency by inter exchanging information in between ITS participants and inducing some kind of behavioural conduct in them. Following, some examples:

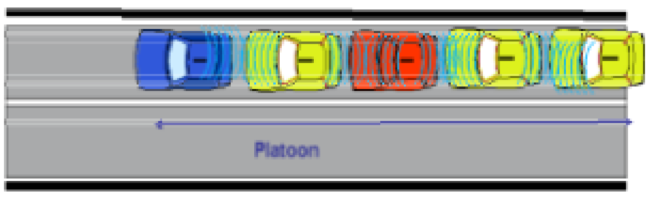
Regulatory/contextual speed limits - Support for a capable Road Side Unit to broadcast at a given frequency the current local speed limits (regulatory and contextual).

Intersection management - Consider all the V2V and V2I co-operation possibilities to improve the traffic efficiency at a road intersection for example through traffic lights synchronization.

Co-operative flexible lane change - Support of flexible allocation of a dedicated lane (e.g. reserved to public transport) to some vehicles which get a permanent or temporary access right under specific conditions (e.g. if no bus is present).

Co-operative vehicle-highway automation system (Platoon) - This use case is based on the use of V2X CAM messages and unicast exchanges for vehicles to operate safely as a platoon on a highway or specific lane.(Figure 2)

Figure 2: *Platoon use case scenario [5]*



As already mentioned, this is the use case applied as a baseline on both developed simulation tools. On the next section, an overview on Platooning and on the implemented algorithm will be shown.

2.1.3 Other scenarios

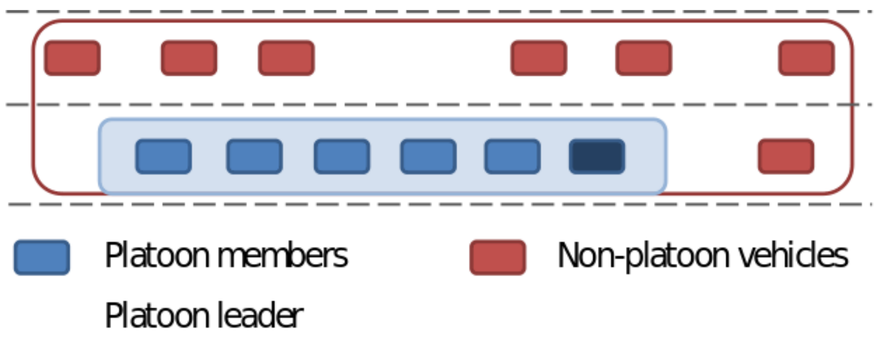
There are other scenarios proposed, however they are not so interesting to be analysed in the context of this thesis, either way, some more examples can be seen at [5] Annex C Section 3. Some of these examples, have already some real deployments working on some urban areas, some already running for some years on. It's relevant to see that [5] was published 10 years ago (June 2009), so it's actually easy to see why some of these scenarios are already deployed. However, the use-case scenarios relative to co-operative road safety and traffic efficiency have a really low percentage of deployment, since most of them require some level of autonomy on vehicles, and even new car models coming out recently don't have these capabilities implemented.

2.2 Cooperative Platooning

Vehicular Platooning (VP) as previously stated, is one of the most interesting scenarios to develop within ITS because of the many advantages it can bring to transportation, both in safety measures and cost-efficiency. VP can potentiate several benefits, such as increasing road capacity and fuel efficiency and even reducing accidents [6], by having vehicle groups traveling close together and following similar paths. However, VP presents several safety challenges, considering it can heavily depend on wireless communications to exchange safety-critical information, and upon a set of sensors that can be affected by noise. For instance, quite often in VP, wireless exchanged messages contribute to maintain the inter-vehicle safety distance, or to relay safety alarms to the following vehicles. Message losses or delays may lead to serious crashes among the vehicles in the platoon with dramatic consequences to them and to the remaining road members [2].

To implement a platooning application, technologies need to be able to follow the preceding vehicle in the platoon while keeping a safe and small inter-vehicle gap. The platooning benefits is proportional to the distance between platoon members. The smaller the distance, more vehicles we can have on the road (better traffic efficiency), and more aerodynamic drag is reduced (better fuel efficiency). However, reducing the distance clearly causes safety issues. Therefore, a trade off is required between inter-vehicle distances and safety to provide fuel efficiency, while preserving safety. In order to implement a platoon model that can handle all these requirements two general approaches can be considered. The first one, completely dependent on V2V communications, where in a platoon of N vehicles, all but the global leader are simultaneously local leaders and followers in what their role on the platoon is regarded. For this approach, every vehicle should have a proper Control algorithm based only on inputs from their local leader, such as, current speed, steering angle, heading, GPS coordinates. In this first approach, and as it is fully dependent on V2V communications, these ones must be assured to be properly working and safe, as any minor issue regarding information reception can be dangerous for a proper platoon maintenance. As this can have serious implications for passenger's safety, the following platooning control approach should be considered. This complementary approach, besides V2V communications must encompass a form of control performed mainly using intra-vehicle sensors, like a LiDAR or a Camera, to be able to properly follow its local leader (car in-front). Obviously, for this approach to be viable and cleanly implemented, the hardware components involved must be very precise and calibrated i.e LiDAR. Adding to this, the software developed must be tested and guarantee some level of reliability so we can depend completely on this approach to maintain the platoon running in the event of communications shortage. These two approaches are important to co-exist in this context, in order to have a flexible and stable way of assuring the proper running of a platoon.

Figure 3: *Platooning Overview*

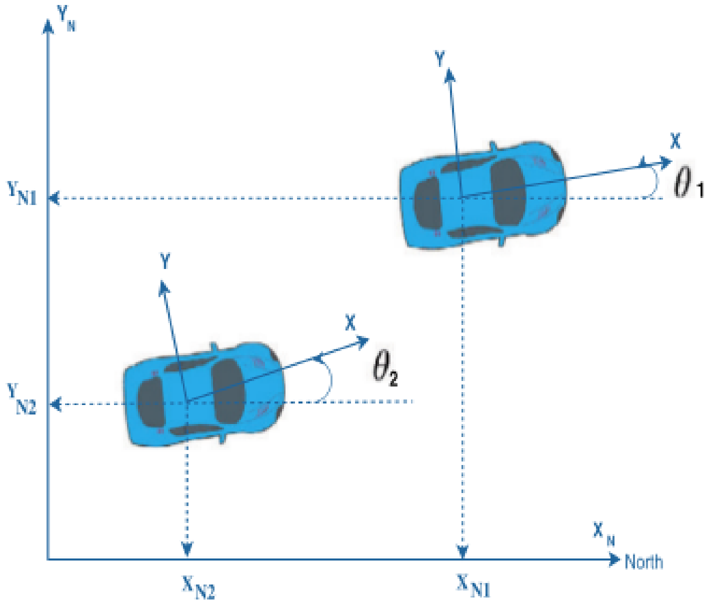


Vehicular Platoon Model Implemented The model used for platoon following in this project was based in the one proposed at [7]. This model, belongs to the group of fully V2V communication dependent models previously stated.

This platoon model supports a global decentralized approach. In a decentralized approach, the control is distributed among vehicles and is not located in one of them. The considered platoon is composed of N identical autonomous vehicles and one different vehicle called the global leader: it is driven manually by a professional driver and it is responsible for defining the platoon properties such as the inter-distance, the speed and the followed trajectory. Each vehicle is defined by its position in a global referential (X,Y,Z). A vehicle Vi is considered as a follower of the vehicle Vi-1 and a local leader of its preceding vehicle Vi+1 . The decentralized approach may increase the dependency in the platoon since each vehicle makes its own decisions and transfers data to the next car that may be affected by the decision error of preceding local leaders.

Following Figure 4, let us suppose a data message $m_{i,i+1}$ sent by a local leader V_i to its follower V_{i+1} . Each $m_{i,i+1}$ contains the following elements: the local leader orientation θ_i and the local leader GPS coordinates (Xi, Yi). Once the vehicle V_{i+1} receives $m_{i,i+1}$, it performs the control process to accomplish the tracking goal. The follower V_{i+1} should gather the following information: the vehicle orientation (its direction to the north) θ_{i+1} , the inter vehicle distance between two vehicles D_{i+1} and $m_{i,i+1}$ data.

Figure 4: *Platoon model. [7]*



Regarding the following actuating control done on the followers, it's important to understand at first, what kind of information does the follower receive from its local leader, the data received through V2V communications is: *GPS coordinates*, *Heading value* (relative to the GPS North axis) and *Current speed.* Of course, some more data could be exchanged and used by the control algorithm for it to get optimized.

The properly applied control for both lateral(steering) and longitudinal(throttle/brake) control was done using PID algorithms for both controls. With inputs being the ones gathered from V2V communication messages and data gathered from the self-vehicle(follower).

For the longitudinal control, the comparison is simple, the PID error is calculated using the GPS distance between both follower and leader vehicles and comparing it to a previously specified set point, i.e if the cars are 10 meters apart and the defined set point is 8 meters, the follower should proceed to accelerate, in order to, reduce this distance. Algorithm 1 was the one implemented following this approach.

For the lateral control, GPS positions, and Heading from both vehicles are used as inputs for the PID algorithm. In this case, the heading of both vehicles is compared to retrieve the PID error, the leader's current heading is used as the set point value for this. Worth to take a look into a developed contribution to this control, that refers

Algorithm 1 Algorithm for longitudinal distance control

```
Input: KP, KI, KD, distance, SAFETY_DISTANCE, speed1 (speed at k),
    speed2(speed at k+1), currentSpeed(Cruise control current speed value), old-
    Value, integral,dt
Output: result
    function PIDsetSafeDistance()
1: if $speed1 = speed2$ then
2:     $Vfollowed$ ←speed1
3: else
4:     $Vfollowed$ ← currentSpeed
5: end if
6: $error$ ← distance-SAFETY_DISTANCE
7: $diff$ ← (error-oldValue)/ dt
8: $result$ ← ((KP × err) + Vfollowed) + (KI × integral) + (KD × diff)
9: Return result
```

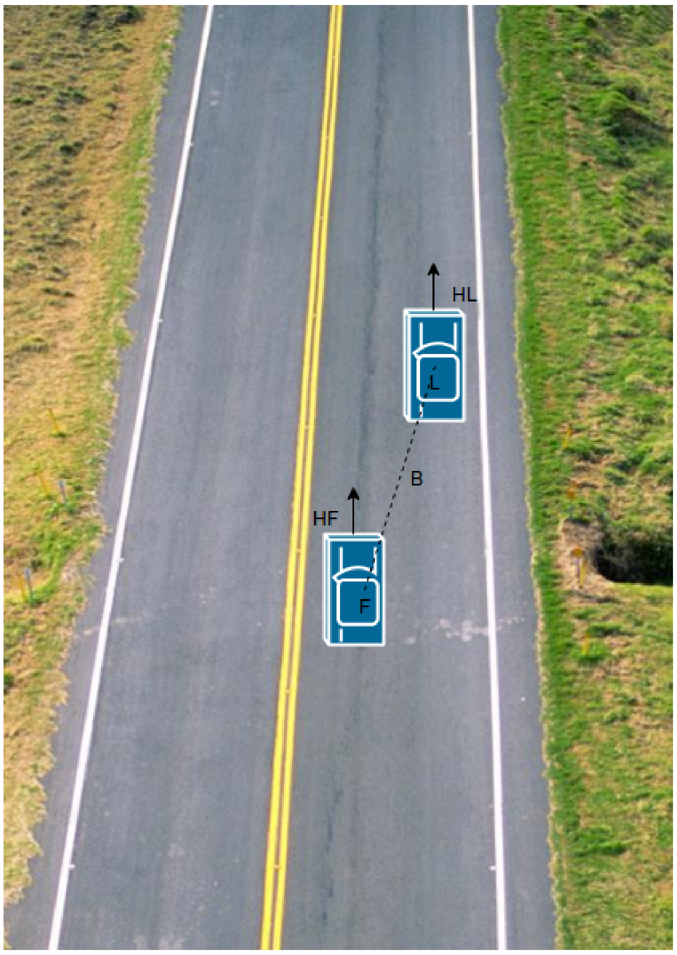
to the bearing angle value being used as a input for this PID. These bearing and heading angles are demonstrated at Figure 5, Hl and Hf being the heading values of both leader and follower vehicles and B being the Bearing angle (calculated through the GPS coordinates) between both vehicles. This bearing component is added to the PID error computation using the existent bearing difference and trying to reach a set point of 0°which means that both vehicles are perfectly aligned in relation to their current heading directions. For instance, at Figure 5 the currently follower, even though their heading is aligned, has to steer to its right in order to follow its leader. Algorithm 2 shows the algorithm used for this control on the developed model.

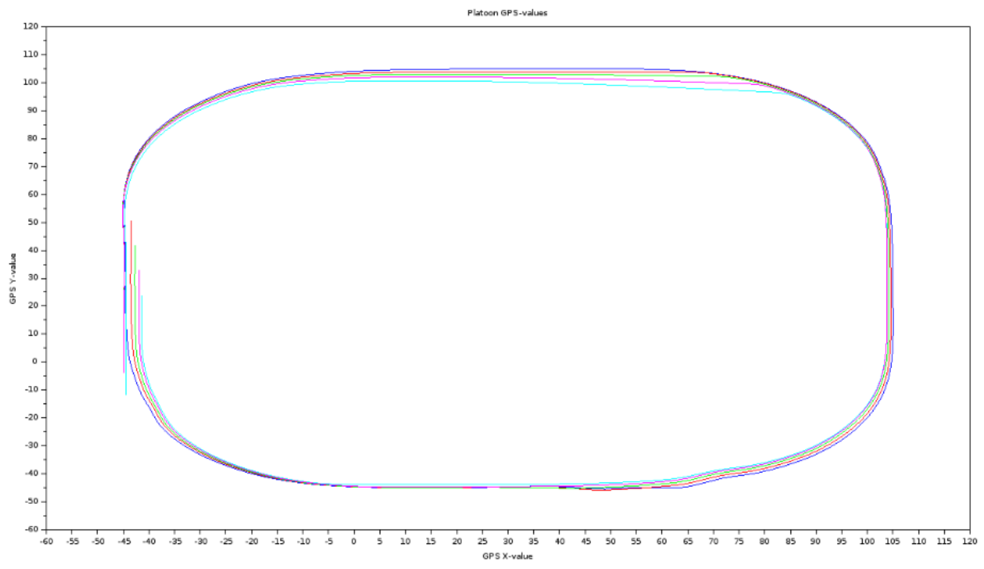
On some tests and analysis done earlier we were able to evaluate this new bearing component impact on the platoon execution. In Figure 6(a) we can see the trajectory of a 5-car platoon going through a simulation of this lateral control without the Bearing component taken into account. At figure 6(b) we can see the same trajectory result of this Bearing addiction on the same platoon run. We can, easily, acknowledge the difference result of this component in regards to lateral differences between vehicles. Each line drawn represents each car of the platoon.

2.3 Conclusions

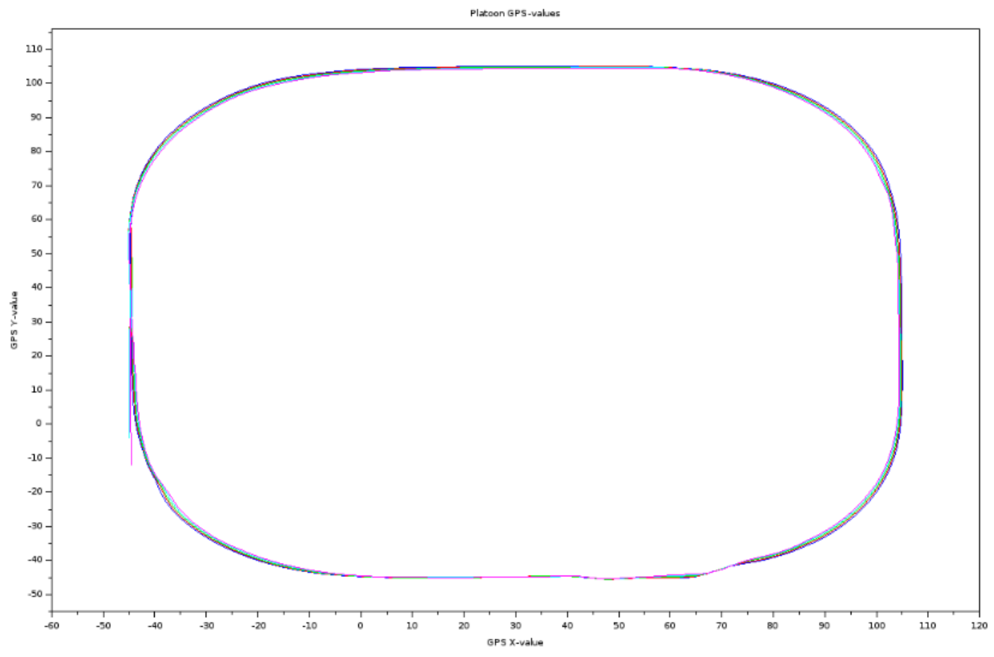
Throughout this section, many different C-ITS scenarios were presented as result or improvements that can be guaranteed with the addiction of V2X communications to these systems.

Figure 5: *Bearing and Heading*





(a)



(b)

Figure 6: *(a) Platoon without bearing component (b)Platoon with bearing component.*

Algorithm 2 Algorithm for steering control

```
Input: KP, KI, KD, bearing _threshold, leader_heading, follower_heading, oldValue, integral, dt, bearing
Output: result
    function PIDsetSteer()
 1: $leader\_heading, follower\_heading, bearing \leftarrow$ convert values from $-\pi$ to $\pi$(rad) into $0°$ to $360°$
 2: $headingfix \leftarrow$ calculate the difference between leader's and follower's heading values
 3: $bearingfix \leftarrow$ similar to headingfix, calculate the difference between follower's heading value and the bearing angle value
 4: if $bearingfix < bearing\_threshold$ then
 5:     $bearingfix \leftarrow 0$
 6: end if
 7: $error \leftarrow$ headingfix+bearingfix
 8: $diff \leftarrow$ (error-oldValue) / dt
 9: result $\leftarrow (KP \times err) + (KI \times integral) + (KD \times diff)$
10: Return result
```

The platoon model used in this Thesis relies solely on V2V communications. Although posing a safety risk in case of communications shortage, it is important as the means to represent a worst-case scenario for the evaluation of he communications impact on platooning.

3 Vehicular Communication Technologies

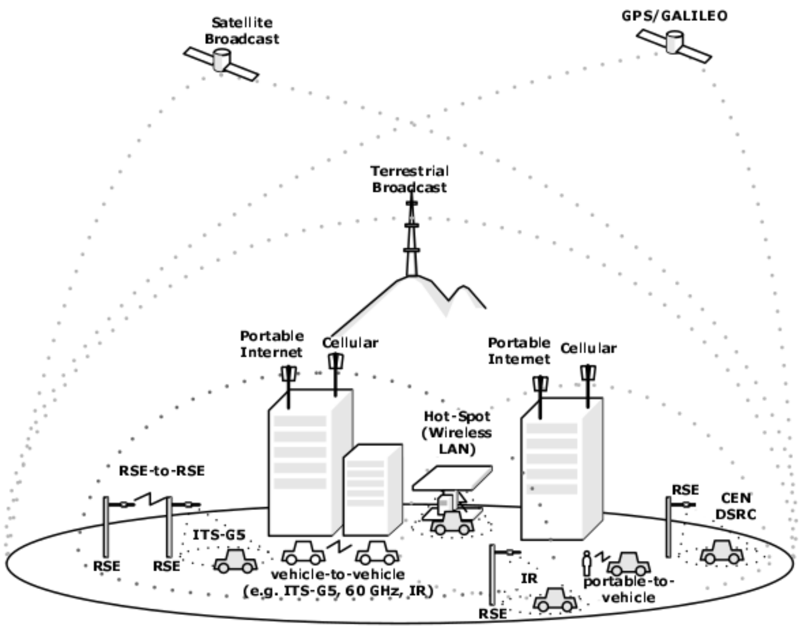
This section presents the current status regarding vehicular communications with a focus on how they can support different C-ITS scenarios where vehicles have some kind of level of autonomy. ETSI ITS-G5, most probably, the most ready to be deployed standard in Europe for this kind of networks and application, will be over viewed, by particularly focusing on the most relevant modules, regarding the specific use-case scenario analyzed within this thesis.

3.1 General C-ITS Architecture

In many respects, today's vehicles are already connected devices. However, in the very near future they will also interact directly with each other and with the road infrastructure. This interaction is the domain of Cooperative Intelligent Transport Systems (C-ITS), which will allow road users and traffic managers to share information and use it to coordinate and co-decide on their actions. The connection and data exchange referred previously is often called and referenced as a Vehicular ad hoc Network (VANET).

A VANET is a sub-form of Mobile Ad-Hoc Network (MANET), since their network nodes (mostly, vehicles) are usually mobile and capable of taking decisions autonomously regarding their positioning or even in their role inside this kind of network. For VANETs, their most relevant basic components are Road-side Units (RSU), On-board units (OBU) and Application-units (AU). Usually, a vehicle is

Figure 7: *ITS-G5 Architecture [8]*



equipped with the last two, AUs and OBUs usually communicate using Intra-car wireless networks, that often include other participants like sensors and, usually, have an interface to other intra-car networks(e.g CAN bus). OBUs are responsible to deal and manage external communications, from these ones Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) are the most relevant. RSUs are, typically, fixed communication units, that provide relevant data (like traffic conditions) to vehicles in their communication range and are connected to the Internet, in order to, provide remote access to infrastructure/road managers.

Figure 7, a full communication architecture for C-ITSs defined by ETSI[8] is shown. In this figure, every possible wireless communication link and participants in a city traffic scenario are presented. Emphasis should be on V2V and V2I provided by ETSI ITS-G5, RSUs previously mentioned are represented as RSE's (Road Side Equipment) in this architecture. Adding to this, satellite links are, as well, shown, providing different services, focus on GPS which is a technology already very mature and deployed. Of course, Internet connectivity is worth to be mentioned since it is almost obligatory to every intelligent system nowadays to be Internet connected as a consequence of all the possible services that can be provided through it. Internet connection, can be achieved through two major forms, cellular networks, 4G/5G, or by getting authenticated access to WLANs through Wi-Fi.

These can be two inter-dependent architectures that, soon, we should be able to see deployed with the majority of the vehicles being driven across urban areas, mainly on big cities. With these kind of communication systems properly deployed and fulfilling certain security and safety requirements, the possibilities of intelligent cooperative services to be developed and deployed seems to be virtually limitless.

3.2 ETSI ITS-G5

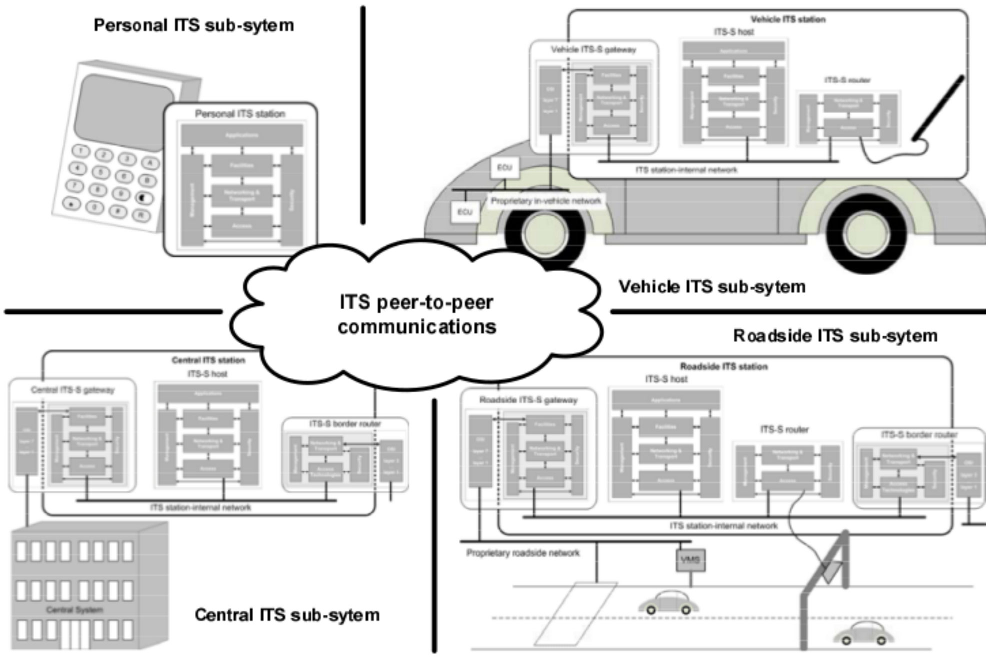
Regarding the type of vehicular communications that are the focus on the tools developed within this thesis' scope, V2V, as previously stated, ETSI ITS-G5 is currently the most widely accepted standard within the biggest European industry players in the automotive area, similarly, in the US, IEEE 1609 WAVE is the region equivalent accepted standard, it is very close in technical terms to what ETSI ITS-G5 specifies, and they both share IEEE 802.11p, as underlying communications layer.

However, there's currently a discussion going on regarding the possibility of usage the LTE technology (LTE-V2X) [9], with 5G in mind, and its advantages comparing to ETSI ITS-G5 usage of IEEE 802.11p, this comparison will not be analyzed within the scope of this project, since the goal is to develop tools, in order to, analyze and evaluate ITS-G5 implementations on different automotive scenarios using the IEEE 802.11p lower layers. Moreover, a recent(04/2019) update on the standard previews the transmission of CA Messages over LTE-V2X channels[10].

As previously stated, ETSI ITS-G5 appears, as the standard with biggest chances of getting implemented on the future of the automotive industry, at least, in Europe. Because of all the complexity existing on all the ITS scenarios, communications and information exchanges are done and should be possible in between different ITS subsystems, that can have a wide variety of topologies, since the most obvious ones, vehicles, to RSUs that can provide useful information regarding the current status of its surroundings, or even, road walk pedestrians carrying a smart phone that can be implied in whatever scenario exists at the moment. Some of these sub-systems are illustrated at Figure 8, as well as, their relative stack architecture.

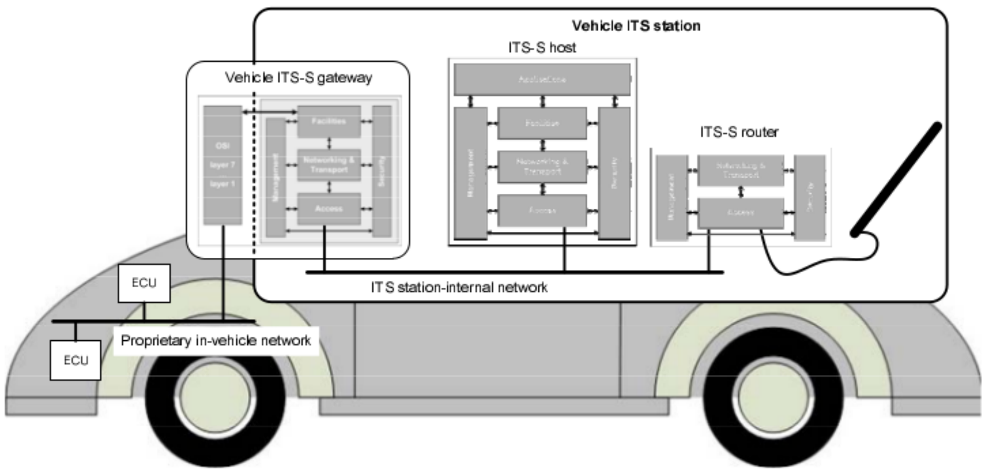
Since only V2V communications are focused in this thesis, a more in-depth overview of the stack architecture expected in a vehicle is shown under Figure 9, where 3 major components can be seen, The ITS-S host, which is the main focus for this project, and represents the component responsible for the majority of the implementation required by ETSI ITS-G5 going since the application layer, until the lower layers that are, as well, implemented on the ITS-S router component. The Vehicle ITS-S gateway is the component responsible for the interface of both pre-

Figure 8: *Illustration of ITS sub-systems [8]*



viously referenced pieces with the Proprietary in-vehicle Network that should vary within different manufacturers, in most cases, this gateway will connect the ITS-S components to a CAN Network, in order to, get access and exchange information with the car's ECUs responsible for the vehicle actuators and to gather some useful data as well.

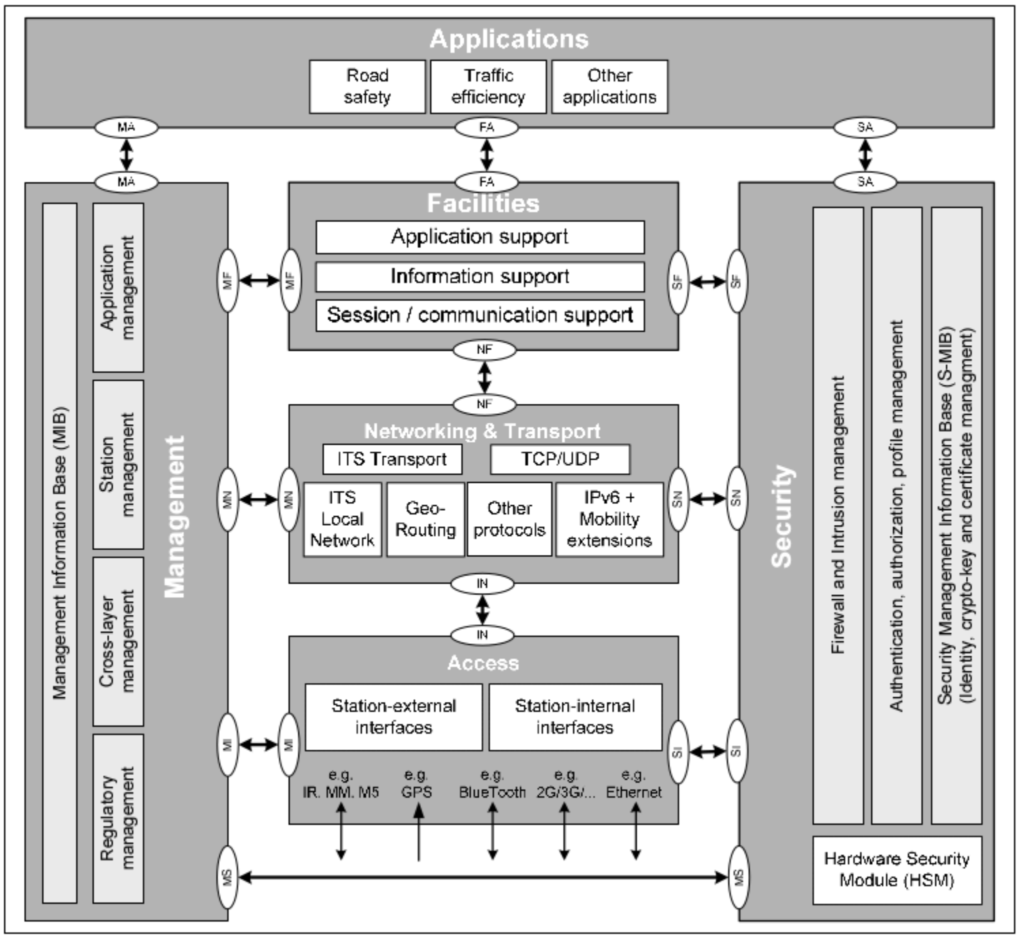
Figure 9: *Vehicle ITS station in a vehicle sub-system [8]*



At Figure 10, the stack architecture that is present in a typical Vehicular ITS-S Host is represented, with some example elements possible to be implemented within each layer. According to the usually used as reference OSI model [11],the OSI's Application layer can be incorporated within both the Applications and Facilities layers specified by ETSI, just like the Access layer defined by ETSI, that brings together the Physical and data link layers from the OSI model.

With this in mind, this section and following work will mostly be focused on the analysis of the Application and Facilities layers and the way they interchange information, in order to, verify possible co-influences under different implementations.

Figure 10: *Examples of possible elements in the ITS station reference architecture [8]*



Starting by providing an overview on these different layers within the ITS-G5 stack:

Application

Applications on ITS scenarios can vary in a wide spectrum of types, however, ITS-G5 divides them in three major groups:

- Road Safety.
- Traffic efficiency.
- Other Applications.

Platooning, the focused ITS scenario on this thesis, belongs to the second referred group of applications, however, can be stated, as well, as a Road Safety application, since the platoon members should implement safety measures in order to guarantee that they don't cause any harm on the normal traffic work flow.

For most of these ITS applications, certain requirements should be guaranteed by their supportive communication services. ETSI ITS-G5 defines the following requirements as the ones to be more or less strictly imposed:

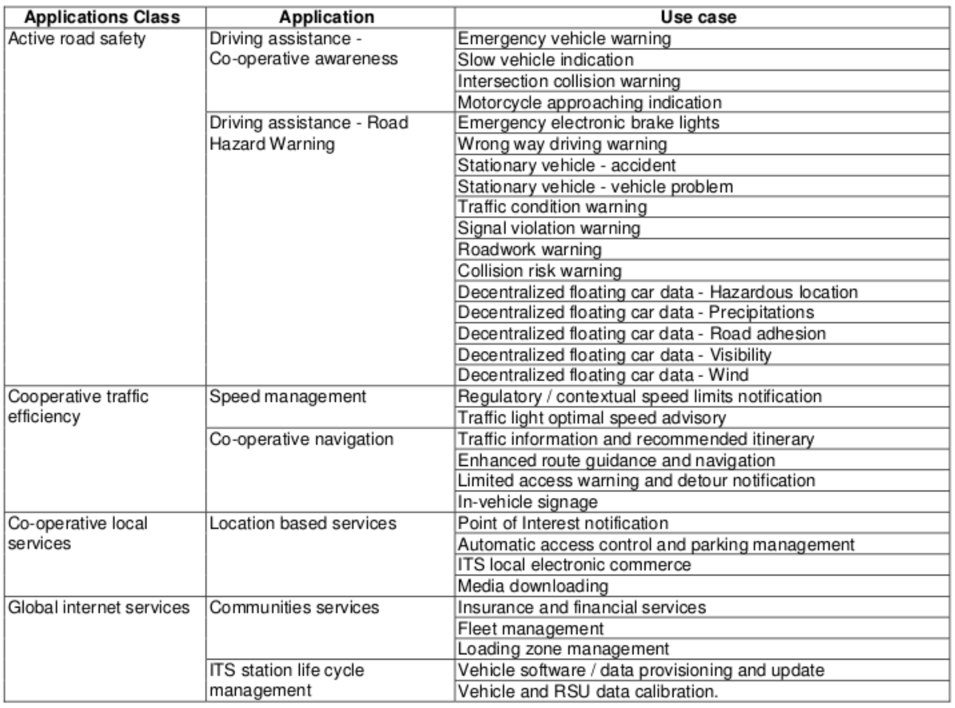
- Reliability.
- Security.
- Latency.
- Other performance parameters.

With all of this in mind, ETSI ITS-G5 defines a set by the name of Basic Set of Applications, that aggregates most of the most relevant ITS applications deployed or to be deployed on vehicles. This set list was created by ETSI with help from a wide variety of users and stakeholders of the automotive industry, taking into account different criteria. In Figure 11 it is possible to see a table of different use cases grouped in various Application group and classes.

A detailed description of all these requirements, different use cases and assessments done to define this Basic Set of Applications, can be found on the annexes of [5]. This document will be, once again, referred later in this thesis when different ITS scenarios are mentioned.

CAM Messages According to ETSI, Cooperative Awareness Messages (CAMs) are sent periodically between ITS stations to all the neighbours stations within communication range(Single-hop and Broadcast).

Figure 11: *Basic Set of Applications definitions [5]*



Applications Class	Application	Use case
Active road safety	Driving assistance - Co-operative awareness	Emergency vehicle warning
		Slow vehicle indication
		Intersection collision warning
		Motorcycle approaching indication
	Driving assistance - Road Hazard Warning	Emergency electronic brake lights
		Wrong way driving warning
		Stationary vehicle - accident
		Stationary vehicle - vehicle problem
		Traffic condition warning
		Signal violation warning
		Roadwork warning
		Collision risk warning
		Decentralized floating car data - Hazardous location
		Decentralized floating car data - Precipitations
		Decentralized floating car data - Road adhesion
		Decentralized floating car data - Visibility
		Decentralized floating car data - Wind
Cooperative traffic efficiency	Speed management	Regulatory / contextual speed limits notification
		Traffic light optimal speed advisory
	Co-operative navigation	Traffic information and recommended itinerary
		Enhanced route guidance and navigation
		Limited access warning and detour notification
		In-vehicle signage
Co-operative local services	Location based services	Point of Interest notification
		Automatic access control and parking management
		ITS local electronic commerce
		Media downloading
Global internet services	Communities services	Insurance and financial services
		Fleet management
		Loading zone management
	ITS station life cycle management	Vehicle software / data provisioning and update
		Vehicle and RSU data calibration.

This type of messages as their name states, is used by ITS-hosts to improve and update its sensing (e.g to evaluate the distance between two vehicles), adding a redundancy layer for ITS vehicles that should feature other ways of sensing their own surroundings (e.g sensors, cameras). Data sent through CAMs is usually about current position and status of it's source ITS host, this Data will be detailed later in this section.

As CAMs are set to be sent periodically, the timing of sending is very important quality requirements for the Applications that might use this type of messages. The CAM generation service should follow some generation rules, in order to fulfill the requirements set by ETSI for the generality of ITS scenarios in its Basic Service Profile (BSP), these rules are: [12]

- maximum time interval between CAM generations: 1 s;
- minimum time interval between CAM generations is 0,1 s.
- generate CAM when absolute difference between current heading (towards North) and last CAM heading $>4°$;
- generate CAM when distance between current position and last CAM position

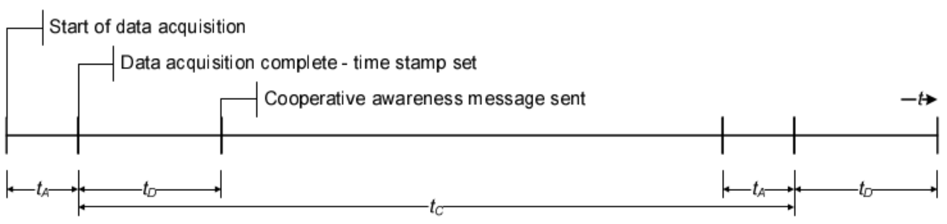
>5 m;

- generate CAM when absolute difference between current speed and last CAM speed >1 m/s;
- These rules are checked latest every 100 ms;

These rules' thresholds are very important in this thesis context, since they are tested out on COPADRIVe simulations at chapter 5.

The timing existent during CAMs generation and processing, follow a time line with a set of a couple of requirements defined, at figure 12, this time line can be seen, as well as, their requirements.

Figure 12: *Time requirements for CAM generation and CAM processing [5]*


Start of data acquisition
Data acquisition complete - time stamp set
Cooperative awareness message sent
t
t_A
t_D
t_C
t_A
t_D

The above requirements are set as:

$t_A \leq 50$ ms;

$t_D \leq 50$ ms.

Some different use case Applications were already taken into account for these requirements, as an example, at Table 3.1 some use cases are specified with regards to their corresponding minimum sending frequency whenever a ITS station takes part in their relative scenarios.

Table 3.1: *Overview Use Cases based on CAM [5]*

Use Case	min Frequency (Hz)	min Latency (ms)
Emergency Vehicle Warning	10	100
Slow Vehicle Indication	2	100
Intersection Collision Warning	10	100
Motorcycle Approaching Indication	2	100
Collision Risk Warning	10	100
Speed Limits Notification	1 to 10	100
Traffic Light Optimal Speed Advisory	2	100

In terms of Data sent by CAMs, all the data sent in a message of this type is encoded using the ASN.1 notation, so, obviously every ITS station should have a encoding/decoding service dedicated to this. ETSI previews a set of four different ITS station profiles:

- basicVehicle The profile "basicVehicle" is mainly used for private vehicles. This profile serves as basis for further profiles,e.g. emergencyVehicle.
- basicIRS The basicIRS is an ITS Roadside Station which can offer all functionality of the infrastructure. It serves as basis for more specialized profiles. The station is installed in a way it is not physically relevant and therefore no danger for the traffic.
- emergencyVehicle
- publicTransportVehicle

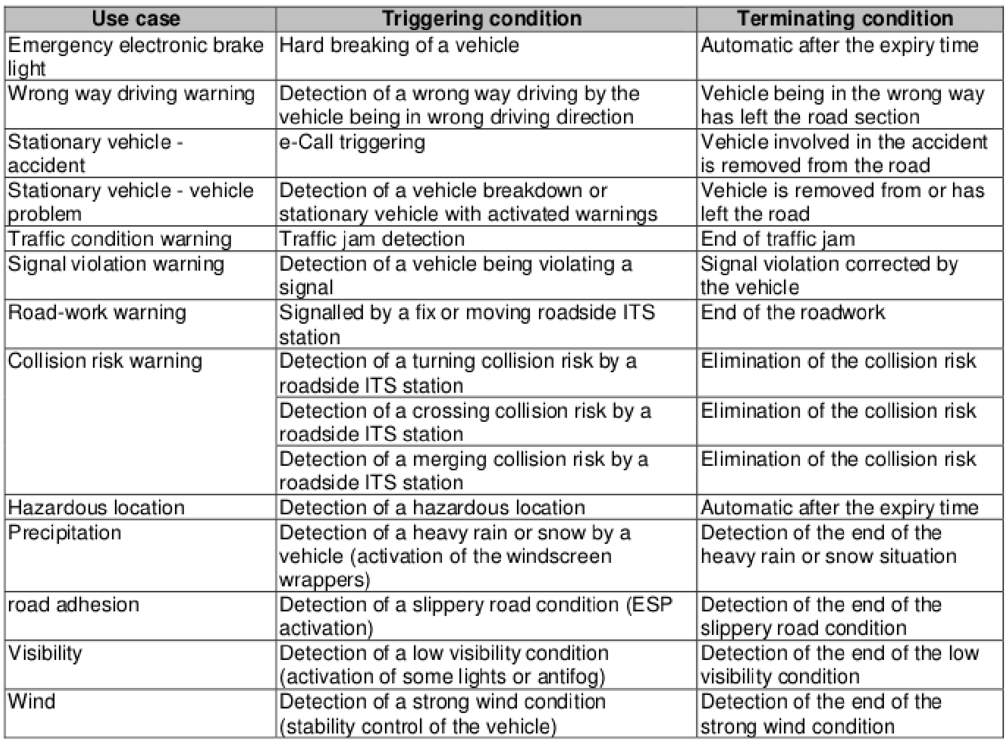
All the data (optional or obligatory) exchanged between ITS stations through CAMs is presented and referenced at Annex A and B, at the ETSI document: [13].

DENM Messages Decentralized Environmental Notification Messages (DENMs) are mainly used by the Cooperative Road Hazard Warning (RHW) application in order to alert road users of the detected events. The RHW application is an event-based application composed of multiple use cases. The general processing procedure of a RHW use case is as follows:

- Upon detection of an event that corresponds to a RHW use case, the ITS station immediately broadcasts a DENM to other ITS stations located inside a geographical area and which are concerned by the event.
- The transmission of a DENM is repeated with a certain frequency.
- This DENM broadcasting persists as long as the event is present.
- The termination of the DENM broadcasting is either automatically achieved once the event disappears after a predefined expiry time, or by an ITS station that generates a special DENM to inform that the event has disappeared.
- ITS stations, which receive the DENMs, process the information and decide to present appropriate warnings or information to users, as long as the information in the DENM is relevant for the ITS station.

At Figure 13, examples of triggering and termination conditions of DENM sending are presented.

Figure 13: *Triggering and termination conditions of DENM sending [5]*



Use case	Triggering condition	Terminating condition
Emergency electronic brake light	Hard breaking of a vehicle	Automatic after the expiry time
Wrong way driving warning	Detection of a wrong way driving by the vehicle being in wrong driving direction	Vehicle being in the wrong way has left the road section
Stationary vehicle - accident	e-Call triggering	Vehicle involved in the accident is removed from the road
Stationary vehicle - vehicle problem	Detection of a vehicle breakdown or stationary vehicle with activated warnings	Vehicle is removed from or has left the road
Traffic condition warning	Traffic jam detection	End of traffic jam
Signal violation warning	Detection of a vehicle being violating a signal	Signal violation corrected by the vehicle
Road-work warning	Signalled by a fix or moving roadside ITS station	End of the roadwork
Collision risk warning	Detection of a turning collision risk by a roadside ITS station	Elimination of the collision risk
	Detection of a crossing collision risk by a roadside ITS station	Elimination of the collision risk
	Detection of a merging collision risk by a roadside ITS station	Elimination of the collision risk
Hazardous location	Detection of a hazardous location	Automatic after the expiry time
Precipitation	Detection of a heavy rain or snow by a vehicle (activation of the windscreen wrappers)	Detection of the end of the heavy rain or snow situation
road adhesion	Detection of a slippery road condition (ESP activation)	Detection of the end of the slippery road condition
Visibility	Detection of a low visibility condition (activation of some lights or antifog)	Detection of the end of the low visibility condition
Wind	Detection of a strong wind condition (stability control of the vehicle)	Detection of the end of the strong wind condition

All the data gathered by DENM messages, similarly to what happens with CAM messages, is used for the ITS station Facilities to update its LDM. DENM messages format follows, as well, the ASN.1 representation, they can be seen at section 6.2.4 from [14].

Management

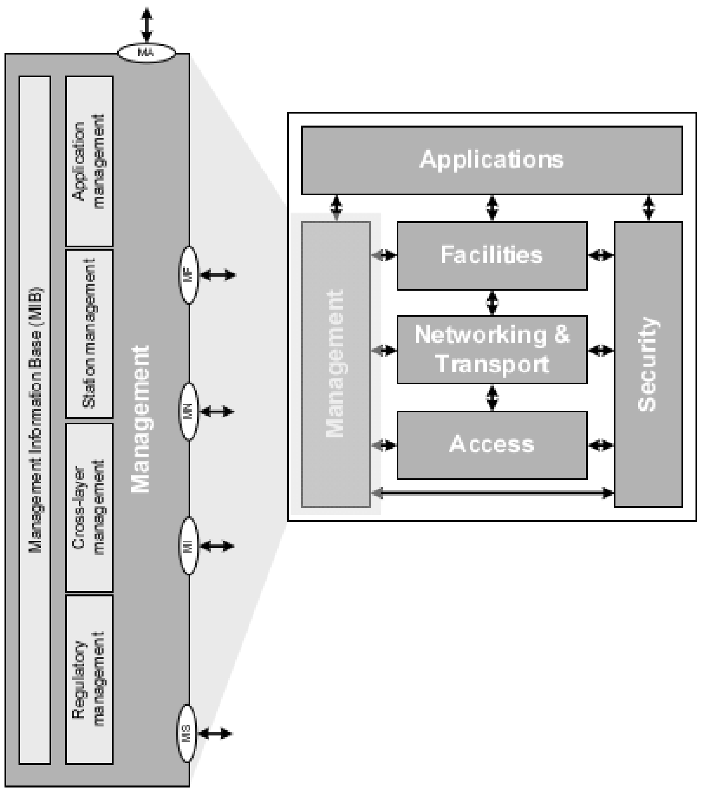
The management layer, as it self-explanatory name specifies, has as its main objective the ability to manage and make sure all the other layers are working as intended and cooperating normally, as intended. Belongs to the group of the only two layers(Security and Management) that have inter-connections with all the other ITS-G5 layers (Figure 14).

Security

Automotive scenarios and the industry itself are very dependent on security critical applications. The following, are the most relevant security functionalities implemented on the ITS-G5 standard:

- Firewall and intrusion management.

Figure 14: *ETSI ITS-G5 Management Layer connections [8]*



- Authentication, authorisation and profile management.
- Identity, crypto key and certificate management.
- A common security information base (SIB).
- Hardware security modules (HSM).
- Interface and security support for all the other ITS-G5 layers.

Facilities

The Facilities layer [8] being the most relevant layer in this thesis context will have a more in-depth focus when comparing to the other described layers in this chapter. Since this layer is the one that manages all the high-level information exchanged in between vehicles and other ITS stations. In this layer functionality from different OSI layers is present, since the session layer, going through the presentation layer(e.g ASN.1 encoding and decoding) until the application layer, these are the main supports provided by this layer:

- application support.
- information support.
- communication support.
- session support.
- Interfaces with all the other ITS-G5 layers.

With this in mind, as it's name states, this layer provides different types of facilities in order to achieve the previously stated goals:

- Generic HMI support:

  This facility presents information to the user of the system, e.g. to the car driver, via the HMI hardware and firmware.

- Support for data presentation:

  Data presentation is a basic functionality of the OSI presentation layer. Its function is to code and decode messages according to formal language being used (e.g. ASN.1).

- Position and time support:

  This facility provides information on the geographical position (longitude, latitude, altitude) of the ITS station, and the actual time.

- Support for maintenance of ITS-S applications:

  This facility supports the download and activation of new application software and the update of already installed software.

- Local dynamic map (LDM) support:

  A cooperative system for road safety critical applications benefits from using digital maps. Such maps used in ITS may include lane-specific information including curbs, pedestrian walking, bicycle paths and road furniture such as traffic signs and traffic lights. Furthermore, all dynamic objects that are directly sensed or indicated by other road users by means of cooperative awareness messages may be referenced in such a digital map, referred to as local dynamic map (LDM).

- Support for common message management for data exchange between ITS-S applications:

  Event triggered messages: Decentralized Environmental Notification Messages (DENMs).

  Periodic messages: Co-operative Awareness Messages (CAMs).

- Support of repetitive transmission of messages:

  This facility is in charge to repetitively request transmission of messages according to the requirements set up by the ITS-S application.

From the following facilities, the last two, that refer to message and data exchange between ITS stations will be focused on the next sub-subsections, mainly CAMs which are the type of message more in-depth analyzed on the tools developed.

Network and Transport layer

The networking and transport layer contains functionality from the OSI network layer and the OSI transport layer with some amendments dedicated to Intelligent Transport System Communications (ITSC):

- Networking protocols (examples shown later).

- Transport protocols (examples shown later).

- Network and transport layer management.
- Interface to all the other ITSC layers.

Within the different Network protocols, these are the ones that are currently supported by ITS-G5:

- GeoNetworking protocol as to be specified next.
- IPv6 networking[15] with IP mobility support specified in RFC 6275 [16] and optionally support for network mobility (NEMO) as defined in RFC 3963 [17] or other approaches depending on the deployment scenario.
- IPv4 support for transition into IPv6 [18].

In the case of Transport protocols that ITS-G5 supports:

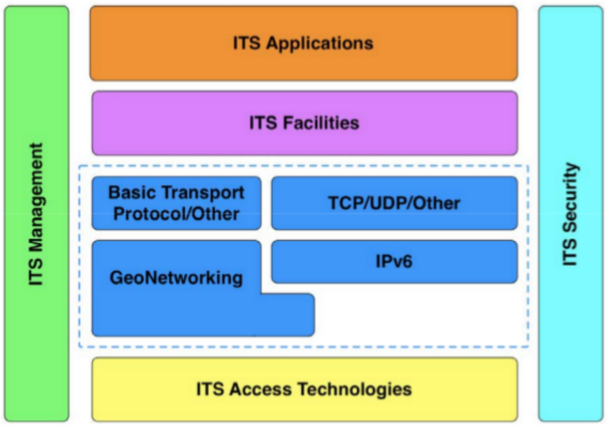
- Basic Transport Protocol (BTP)[19].
- User Datagram Protocol UDP as defined in RFC 768[20].
- Transmission Control Protocols TCP as specified in RFC 793[21].

The widely used UDP/TCP protocols are a frequent choice. However, ETSI ITS-G5 mainly uses BTP as its preferred transport protocol, it was the one used on this thesis latterly referred implementations. As instance, BTP is used over the GeoNetworking protocol, both UDP and TCP are planned to be integrated as well, on top of this protocol. Figure 15 presents an overview of GeoNetworking and IPv6 integrated in an ITS station.

GeoNetworking The ETSI GeoNetworking protocol, controls the transport of data packets from a source node to the destination, which can be either an individual node (GeoUnicast), all nodes/any node inside a geo-area (GeoBroadcast/GeoAnycast), or all nodes in a one-hop/n-hop neighborhood (single-hop/topologically-scoped broadcast). Every ad hoc router has a location table that maintains the position of its known neighbors and is used to make forwarding decisions; it also has packet buffers for location service, store-carry-and-forward and forwarding algorithms.

For safety and traffic efficiency use cases, two packet transport types are relevant: first, single-hop broadcast for the transmission of periodic CAMs, and second, Geo-Broadcast for the multi-hop distribution of event- driven messages within a geo-area, DENMs. The ETSI GeoNetworking protocol specifies three main forwarding algorithms to distribute messages in a geo-area: Simple Geo-Broadcast and

Figure 15: *Combined GeoNetworking and IPv6 stack in an ITS station [22]*



contention-based forwarding as base schemes, and advanced forwarding, which combines both base schemes and comprises a set of protocol mechanisms to improve their performance.

Geo-Broadcast and the remaining forwarding algorithms detailed description can be seen at [23].

Access layer

The access layer, provides the means to access the medium. This layer incorporates both the PHY and Data link layer(DLL).

The first one responsible for physically connecting to the communication medium.

The DLL can be sub-divided in two sub-layers, MAC and LLC, the MAC layer being responsible for the management of access to the medium, while the LLC working to provide logical link control.

This layer is defined by IEEE 802.11p [24], this standard was created based on IEEE 802.11a, however, to focus and serve, vehicular scenarios, with some objectives in mind:

- Increase the maximum distance of operation (around 1 km);

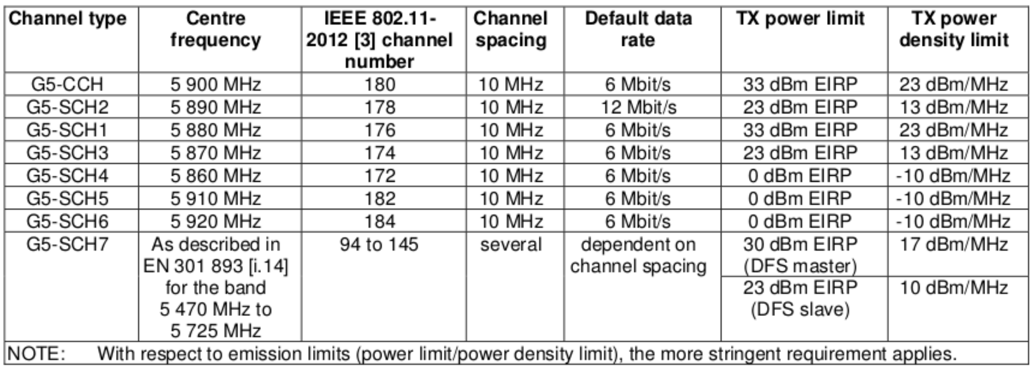
- High mobility and speed of the network nodes.

- A way to control and attenuate the Multipath effect - existence of multiple signal echos received.

-Try to guarantee the best QoS, regarding the amount of different ad-hoc networks existing in these environments.

These goals are achieved by different characteristics imposed by this technology, as an example this is the channel(10MHz channel spacing) distribution allocated within the 5.9GHz band for the ITS-G5 standard (Table 3.2). Some relevant channel characteristics are present, as well. It is important to notice that CAM messages, which are the main application focus of these implementations, are sent under the G5-CCH channel.

Table 3.2: *ETSI ITS-G5 Channel allocation [25]*



Channel type	Centre frequency	IEEE 802.11-2012 [3] channel number	Channel spacing	Default data rate	TX power limit	TX power density limit
G5-CCH	5 900 MHz	180	10 MHz	6 Mbit/s	33 dBm EIRP	23 dBm/MHz
G5-SCH2	5 890 MHz	178	10 MHz	12 Mbit/s	23 dBm EIRP	13 dBm/MHz
G5-SCH1	5 880 MHz	176	10 MHz	6 Mbit/s	33 dBm EIRP	23 dBm/MHz
G5-SCH3	5 870 MHz	174	10 MHz	6 Mbit/s	23 dBm EIRP	13 dBm/MHz
G5-SCH4	5 860 MHz	172	10 MHz	6 Mbit/s	0 dBm EIRP	-10 dBm/MHz
G5-SCH5	5 910 MHz	182	10 MHz	6 Mbit/s	0 dBm EIRP	-10 dBm/MHz
G5-SCH6	5 920 MHz	184	10 MHz	6 Mbit/s	0 dBm EIRP	-10 dBm/MHz
G5-SCH7	As described in EN 301 893 [i.14] for the band 5 470 MHz to 5 725 MHz	94 to 145	several	dependent on channel spacing	30 dBm EIRP (DFS master)	17 dBm/MHz
					23 dBm EIRP (DFS slave)	10 dBm/MHz
NOTE: With respect to emission limits (power limit/power density limit), the more stringent requirement applies.						

Due to the MAC protocol of IEEE 802.11-2012 [24], and the limited bandwidth of ITS-G5, the data load on the wireless channels can exceed the available capacity in some situations. Therefore, Decentralized Congestion Control (DCC) methods as specified in TS 102 687[26] are required in ITS-G5 stations in order to control the channel load and avoid unstable behaviour of the system.

Decentralized Congestion Control This DCC mechanism [26] is a crucial part of this access layer, DCCs main goal of implementation and usage is to maintain network stability, throughput efficiency and fair resource allocation to ITS-G5 stations. DCC requires components on several layers of the protocol stack and these components jointly work together to fulfil all the mandatory requirements. Important to notice that DCC doesn't have any control over frequency channel selection on DENM or CAM Messages, the only implementation goal of DCC on this kind of applications is to control Message delivering and guarantee a good QoS only limiting message providing within certain time constraints. A more in-depth look at how this mechanism behaves can be done at [26], Lyamin et al.[27] do a great at synthesizing this.

Medium Access Control(MAC) The MAC algorithm decides when in time a node is allowed to transmit based on the current channel status and the MAC schedules transmission with the goal to minimize the interference in the system to increase the packet reception probability. The MAC algorithm deployed by 802.11p is found in the IEEE 802.11-2012 [24] and it is called enhanced distributed coordination access (EDCA). It is based on the basic distributed coordination function (DCF) but adds QoS attributes. DCF is a carrier sense multiple access with collision avoidance (CSMA/CA) algorithm.

In CSMA/CA a node starts to listen to the channel before transmission and if the channel is perceived as idle for a predetermined listening period the node can start to transmit directly. If the channel becomes occupied during the listening period the node will perform a backoff procedure, i.e. the node has to defer its access according to a randomized time period. The predetermined listening period is called either arbitration interframe space (AIFS) or distributed interframe space (DIFS) depending upon the mode of operation (EDCA or DCF).

Figure 16 shows a drawing of how this channel access mechanism works in both broadcast and unicast modes.

Physical Layer (PHY) The PHY in 802.11p is OFDM detailed in clause 18 of 802.11. The basic idea is to divide the available frequency spectrum into narrower subchannels (subcarriers). The high-rate data stream is split into a number of lower-rate data streams transmitted simultaneously over a number of subcarriers, where each subcarrier is narrow banded. There are 52 subcarriers, where 48 are used for data and 4 are pilot carriers. The OFDM PHY layer has support for eight different transfer rates, which are achieved by using different modulation schemes and coding rates. In Table B.1 the different transfer rates together with the coding schemes used in 802.11p are tabulated for 10 MHz frequency channels. Support of three transfer rates are mandatory; 3 Mbit/s, 6 Mbit/s, and 12 Mbit/s, the last two are the ones used on the channels described at Table 3.2.

The resulting PHY packet, ready for transmission, the physical layer convergence procedure (PLCP) protocol data unit (PPDU), is composed as presented at figure 17

A description overview of the different fields present at the PPDU can be seen at figure 18.

Figure 16: *Channel Access procedure in broadcast and unicast mode [28]*

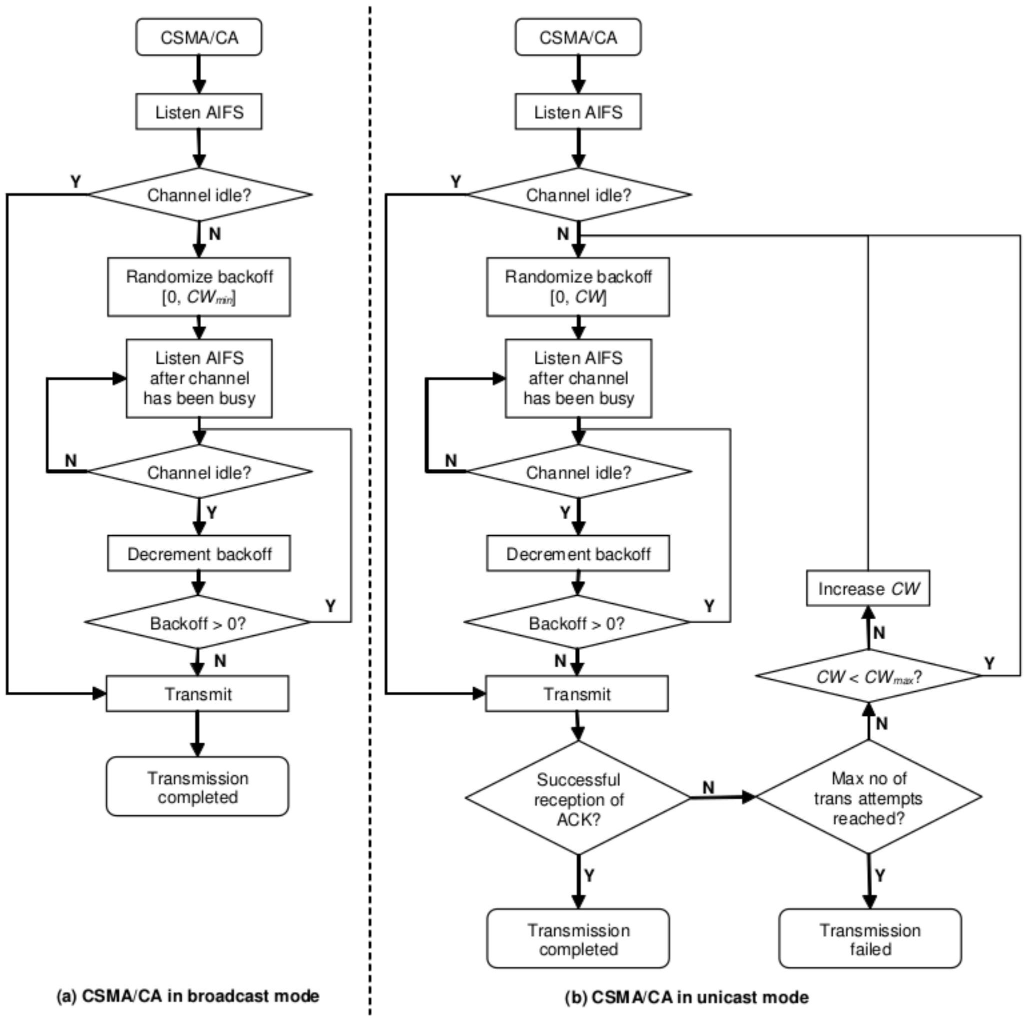


Figure 17: *PHY packet, i.e. PPDU, ready for transmission [28]*

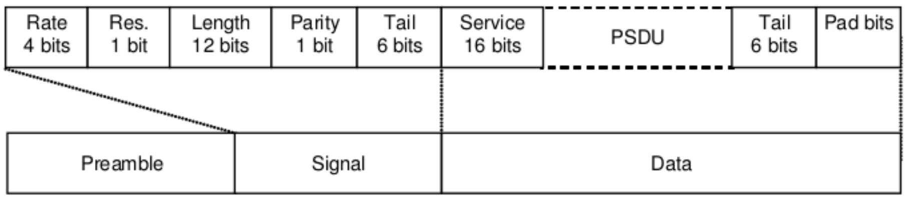
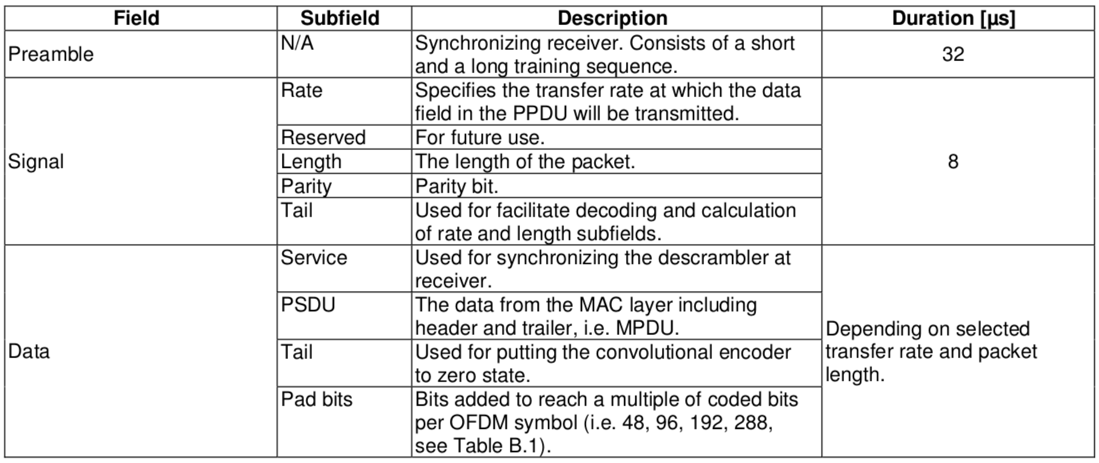


Figure 18: *Explanation of the different fields of the PPDU [28]*

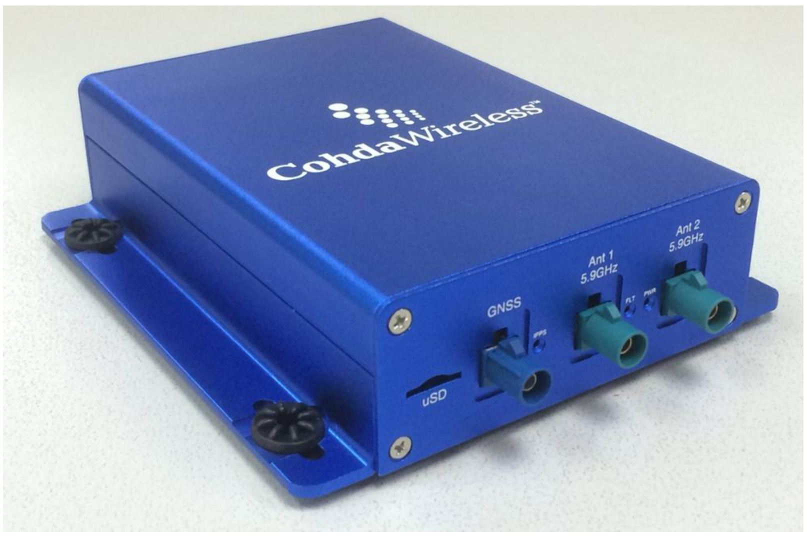


Field	Subfield	Description	Duration [µs]
Preamble	N/A	Synchronizing receiver. Consists of a short and a long training sequence.	32
Signal	Rate	Specifies the transfer rate at which the data field in the PPDU will be transmitted.	8
	Reserved	For future use.	
	Length	The length of the packet.	
	Parity	Parity bit.	
	Tail	Used for facilitate decoding and calculation of rate and length subfields.	
Data	Service	Used for synchronizing the descrambler at receiver.	Depending on selected transfer rate and packet length.
	PSDU	The data from the MAC layer including header and trailer, i.e. MPDU.	
	Tail	Used for putting the convolutional encoder to zero state.	
	Pad bits	Bits added to reach a multiple of coded bits per OFDM symbol (i.e. 48, 96, 192, 288, see Table B.1).	

3.3 Vehicular Communication Platforms

Cohda's MK5 On-Board Unit (Figure 19) is one of the two products available by Cohda Wireless for C-ITS and Smart Cities implementations that rely on V2X communication, the other one being a Road-side unit, very similar in terms of technical hardware characteristics. This OBU was the platform used in the HIL simulator described at Chapter 6.

Figure 19: *Cohda's MK5 On Board Unit*



Cohda's fifth-generation On-Board Unit (MK5) is a small module that can be retrofitted to vehicles for aftermarket deployment or field trials in a off-the-shelf deployment manner. It is based on the automotive-grade RoadLINK chipset developed by Cohda and NXP Semiconductors. These are the key features chosen by the company[29]:

- Dual IEEE 802.11p radio;
- Powerful processor running Cohda software applications;
- Global Navigation Satellite System (GNSS) that delivers lane-level accuracy;
- Integrated security;
- Hardware acceleration;
- Tamper-proof key storage;
- NXP chips with Cohda firmware;
- Supports DSRC (IEEE 802.11p), Ethernet 100 Base-T

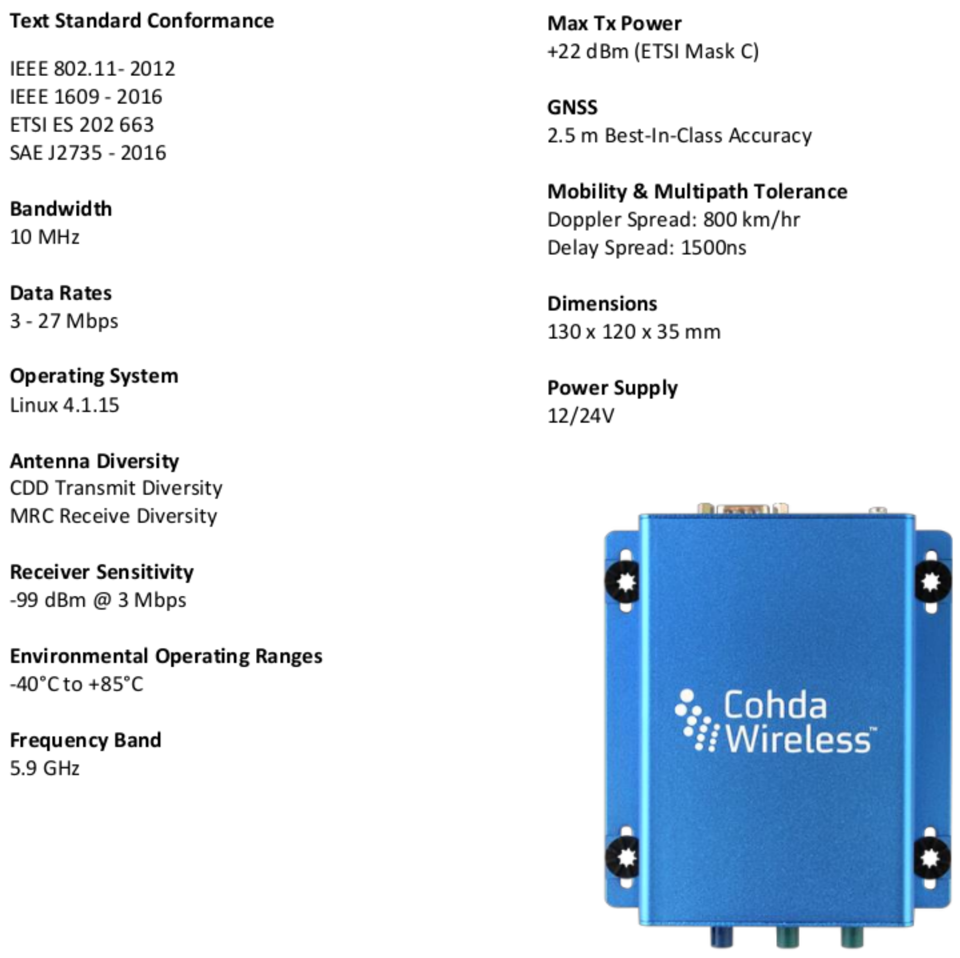
Apart from the IEEE 802.11p support, these OBUs have, as well, ETSI ITS-G5 compatibility by providing a proprietary licensed software suite that enables this[30].

At figure 20, some detailed specifications about this product can be found. The most relevant characteristics are: the running Operating System based on Linux, which enables a great platform for software development and integration, its small dimensions are, as well, a great feature since their main focus is to get implemented in a retrofitting manner on existing vehicles and the GNSS Accuracy is very important, since most of the running applications rely on this to be implemented (e.g CLW).

3.4 Conclusions

As previously referred, ETSI ITS-G5 is, currently, the most widely accepted standard for vehicular communications by the majority of European entities that affect the automotive industry. Knowing this, over viewing this standard is thus mandatory to fulfill the objective of this thesis.

Figure 20: *Cohda's MK5 OBU specifications*



Text Standard Conformance

IEEE 802.11- 2012
IEEE 1609 - 2016
ETSI ES 202 663
SAE J2735 - 2016

Bandwidth
10 MHz

Data Rates
3 - 27 Mbps

Operating System
Linux 4.1.15

Antenna Diversity
CDD Transmit Diversity
MRC Receive Diversity

Receiver Sensitivity
-99 dBm @ 3 Mbps

Environmental Operating Ranges
-40°C to +85°C

Frequency Band
5.9 GHz

Max Tx Power
+22 dBm (ETSI Mask C)

GNSS
2.5 m Best-In-Class Accuracy

Mobility & Multipath Tolerance
Doppler Spread: 800 km/hr
Delay Spread: 1500ns

Dimensions
130 x 120 x 35 mm

Power Supply
12/24V

4 Simulation technologies

In this chapter, different simulator tools and frameworks will be presented. Robotic and Network simulators are the simulation tools that are relevant in the scope of this thesis, since the main goal was to develop such a co-simulation framework.

Hence, Robotic simulators will be over viewed and, in particular, the two most relevant candidates to be part of COPADRIVe, Webots and Gazebo. Later a comparison between them and the reasoning behind the choice of Gazebo will be presented.

In addition, general Network simulators will be shown and discussed with a focus on ns-3 and OMNeT++. These two are, arguably the most used open-source and community supported that exist in the area, with already some work done in vehicular communications scenarios.

Following, a section about Artery [31] a simulation tool that joins a traffic simulator (SUMO) and OMNeT++ with the ETSI ITS-G5 stack. This is the project where most of the OMNeT++ model of COPADRIVe was based on.

Then the Robotic Operating System (ROS) framework will be over viewed, with some of its more interesting features, as the enabling technology for this integration.

Importantly, a relevant criteria for the simulation framework is that it should rely on open-source technologies and have a good community support, as much as possible.

4.1 Generic Network Simulators

4.1.1 ns-3

ns-3 is a discrete-event network simulator, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use. With a high focus on community development and support.

Furthermore, the ns-3 software infrastructure encourages the development of simulation models which are sufficiently realistic to allow ns-3 to be used as a real time network simulator, interconnected with the real world and which allows many existing real-world protocol implementations to be reused within ns-3.

The ns-3 simulation core supports research on both IP and non-IP based networks. However, the large majority of its users focuses on wireless/IP simulations which involve models for Wi-Fi, WiMAX, or LTE for layers 1 and 2 and a variety of static or dynamic routing protocols such as OLSR and AODV for IP-based applications.

ns-3 is based on C++ applications usage for its simulations set up and runs, however, it presents a framework allowing python bindings to call C++ code in order to be able to script properly simulation models directly using Python applications.

For vehicular scenarios, ns-3 has simulation models already developed and ready to use, however resembling the North American ITS communications standard, WAVE, which uses some PHY and MAC layer particularities from IEEE 802.11p like ETSI ITS-G5, however, apart from these two layers there are a lot of differences between these standards higher layer. This was the main reason on why ns-3 was not the choice for deployment within COPADRIVe (section 5).

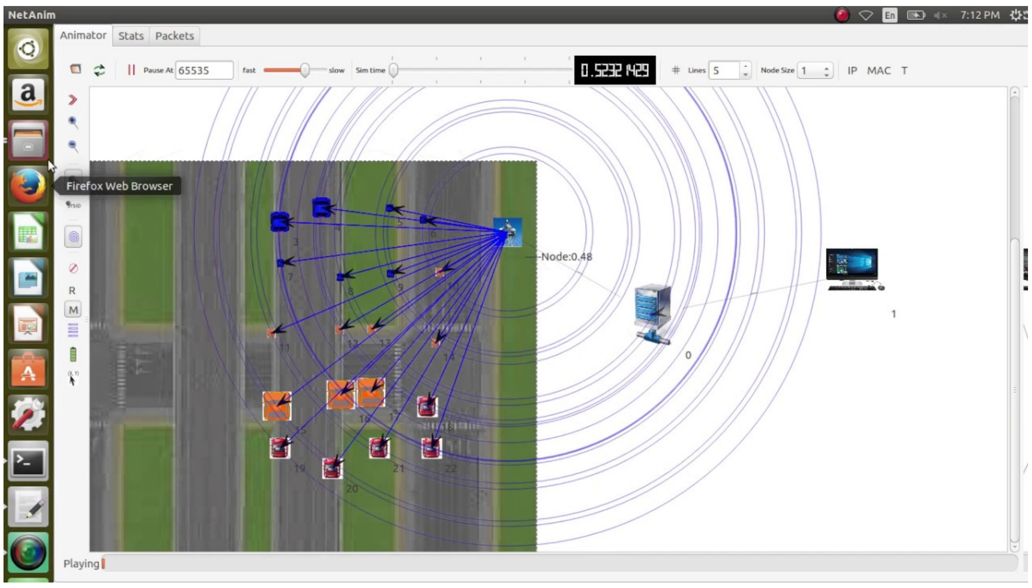
4.1.2 OMNet++

OMNeT++ is a C++-based discrete event simulator for modeling communication networks, multiprocessors and other distributed or parallel systems. OMNeT++ is open-source, and can be used under the Academic Public License that makes the software free for non-profit use.

These were the main design requirements set for OMNeT++ initial development: [32]

- To enable large-scale simulation, simulation models need to be hierarchical, and built from reusable components as much as possible.

Figure 21: *ns-3 vehicular network model*



- The simulation software should facilitate visualizing and debugging of simulation models in order to reduce debugging time, which traditionally takes up a large percentage of simulation projects.

- The simulation software itself should be modular, customizable and should allow embedding simulations into larger applications such as network planning software.

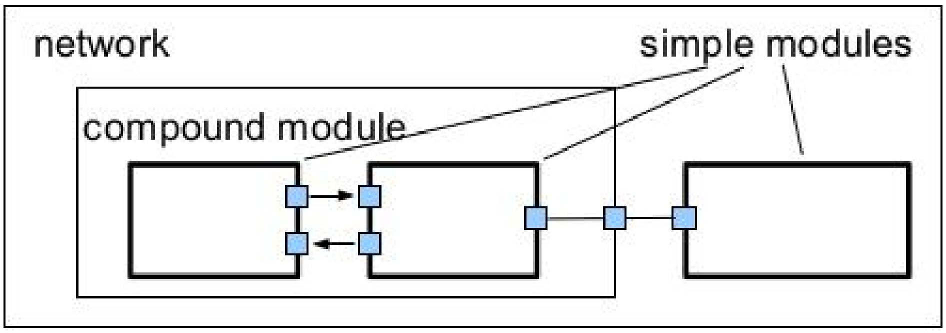
- Data interfaces should be open: it should be possible to generate and process input and output files with commonly available software tools.

- Should provide an Integrated Development Environment that largely facilitates model development and analyzing results.

Of course, most of these requirements are similar to the majority of the simulators in existance, however, for example the first requirement, being able to provide hierarchically simulation models, is something that sets OMNeT++ apart from other simulators.

An OMNeT++ model consists of modules that communicate with message passing. The active modules are termed simple modules; they are written in C++, using the simulation class library. Simple modules can be grouped into compound modules and so forth; the number of hierarchy levels is not limited. Messages can be sent either via connections that span between modules or directly to their destination modules. Figure 22 represents the model structure in OMNeT++, boxes represent simple modules (thick border), and compound modules (thin border), arrows connecting small boxes represent connections and gates.

Figure 22: *OMNeT++ model*



The structure of the model is defined in OMNeT++'s topology description language, NED. Typical members of a NED description are simple module declarations, compound module definitions and network definitions. Simple module declarations describe the interface of the module: gates and parameters. Compound module definitions consist of the declaration of the module's external interface (gates and parameters), and the definition of submodules and their interconnection. Network definitions are compound modules that qualify as self-contained simulation models.

Model behavior is captured in C++ files as code applications, while model topology and some of the parameters defining this topology are defined by the NED files. With this in mind, is normal to see OMNeT++ model's modules having three types of files defining it, one c++ code application file, a c++ header and a .ned file describing the module.

In a generic simulation scenario, one usually wants to know how the simulation behaves with different inputs. These variables neither belong to the behavior (C++ code) or the topology (NED files) as they can change from run to run. INI files are used to store these values.

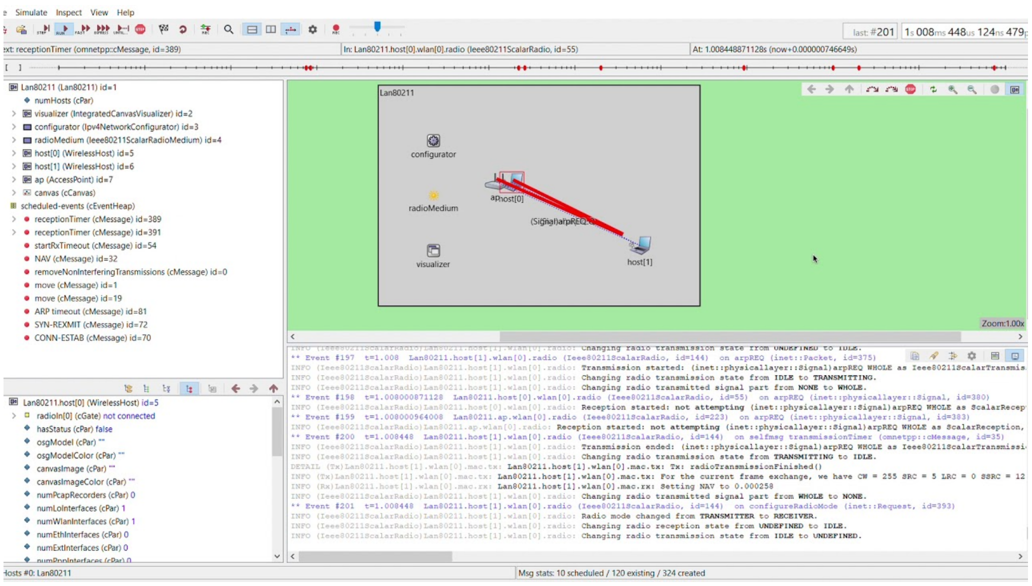
OMNeT++ features three types of user interfaces available to run simulations on:

- Qtenv: Qt-based graphical user interface, available since OMNeT++ 5.0.23
- Tkenv: the traditional, Tcl/Tk-based graphical user interface.
- Cmdenv: command-line user interface for batch execution.

COPADRIVe uses Qtenv as its default user interface for simulation running, since it requires the "Fast" running mode for it to be perfectly synchronized with Gazebo, this will be detailed later at 5.

All these features adding to the fact that Artery [33] already implemented C-ITS scenarios following the ETSI ITS-G5 standard were the reasons behind the choice of OMNeT++ for this project.

Figure 23: *OMNeT++ qtenv UI*



last: #201 | 1s 008ms 448us 124ns 479p

ext: receptionTimer (omnetpp::cMessage, id=389) | In: Lan80211.host[0].wlan[0].radio (Ieee80211ScalarRadio, id=55) | At: 1.008448871128s (now+0.000000746649s)

- Lan80211 (Lan80211) id=1
  - numHosts (cPar)
  - visualizer (IntegratedCanvasVisualizer) id=2
  - configurator (Ipv4NetworkConfigurator) id=3
  - radioMedium (Ieee80211ScalarRadioMedium) id=4
  - host[0] (WirelessHost) id=5
  - host[1] (WirelessHost) id=6
  - ap (AccessPoint) id=7
  - canvas (cCanvas)
- scheduled-events (cEventHeap)
  - receptionTimer (cMessage) id=389
  - receptionTimer (cMessage) id=391
  - startRxTimeout (cMessage) id=54
  - NAV (cMessage) id=32
  - removeNonInterferingTransmissions (cMessage) id=0
  - move (cMessage) id=1
  - move (cMessage) id=19
  - ARP timeout (cMessage) id=81
  - SYN-REXMIT (cMessage) id=72
  - CONN-ESTAB (cMessage) id=70

- Lan80211.host[0] (WirelessHost) id=5
  - radioIn[0] (cGate) not connected
  - hasStatus (cPar) false
  - osgModel (cPar) ""
  - osgModelColor (cPar) ""
  - canvasImage (cPar) ""
  - canvasImageColor (cPar) ""
  - numPcapRecorders (cPar) 0
  - numLoInterfaces (cPar) 1
  - numWlanInterfaces (cPar) 1
  - numEthInterfaces (cPar) 0
  - numExtInterfaces (cPar) 0

```
** Event #197  t=1.008  Lan80211.host[1].wlan[0].radio (Ieee80211ScalarRadio, id=144)  on arpREQ (inet::Packet, id=375)
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Transmission started: (inet::physicallayer::Signal)arpREQ WHOLE as Ieee80211ScalarTransmis
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Changing radio transmission state from IDLE to TRANSMITTING.
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Changing radio transmitted signal part from NONE to WHOLE.
** Event #198  t=1.008000871128  Lan80211.host[0].wlan[0].radio (Ieee80211ScalarRadio, id=55)  on arpREQ (inet::physicallayer::Signal, id=380)
INFO (Ieee80211ScalarRadio)Lan80211.host[0].wlan[0].radio: Reception started: not attempting (inet::physicallayer::Signal)arpREQ WHOLE as ScalarRecep
** Event #199  t=1.008000964008  Lan80211.ap.wlan[0].radio (Ieee80211ScalarRadio, id=223)  on arpREQ (inet::physicallayer::Signal, id=383)
INFO (Ieee80211ScalarRadio)Lan80211.ap.wlan[0].radio: Reception started: not attempting (inet::physicallayer::Signal)arpREQ WHOLE as ScalarReception,
** Event #200  t=1.008448  Lan80211.host[1].wlan[0].radio (Ieee80211ScalarRadio, id=144)  on selfmsg transmissionTimer (omnetpp::cMessage, id=35)
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Transmission ended: (inet::physicallayer::Signal)arpREQ WHOLE as Ieee80211ScalarTransmissi
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Changing radio transmission state from TRANSMITTING to IDLE.
DETAIL (Tx)Lan80211.host[1].wlan[0].mac.tx: Lan80211.host[1].wlan[0].mac.tx: Tx: radioTransmissionFinished()
INFO (Tx)Lan80211.host[1].wlan[0].mac.tx: Lan80211.host[1].wlan[0].mac.tx: For the current frame exchange, we have CW = 255 SRC = 5 LRC = 0 SSRC = 12
INFO (Rx)Lan80211.host[1].wlan[0].mac.rx: Lan80211.host[1].wlan[0].mac.rx: Setting NAV to 0.000258
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Changing radio transmitted signal part from WHOLE to NONE.
** Event #201  t=1.008448  Lan80211.host[1].wlan[0].radio (Ieee80211ScalarRadio, id=144)  on configureRadioMode (inet::Request, id=393)
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Radio mode changed from TRANSMITTER to RECEIVER.
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Changing radio reception state from UNDEFINED to IDLE.
INFO (Ieee80211ScalarRadio)Lan80211.host[1].wlan[0].radio: Changing radio transmission state from IDLE to UNDEFINED.
```

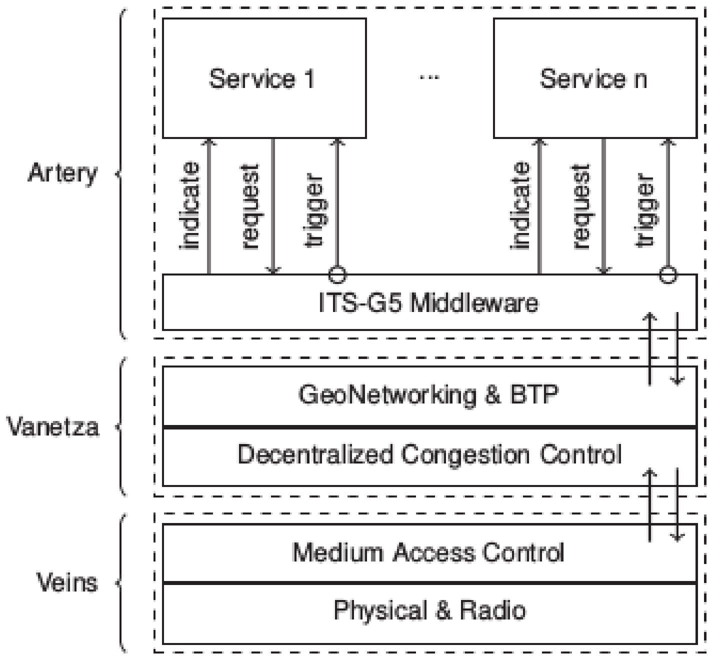
Hosts #0: Lan80211 | Msg stats: 10 scheduled / 120 existing / 324 created

4.2 Vehicular Network Simulation tools - Artery Project

Artery[31], was the project where the development of COPADRIVe was based on. It is the first and unique project that integrates a Traffic simulator (SUMO) with a Network Simulator (OMNeT++) by using Veins [34] and adding the compliance with the ETSI ITS-G5 standard by integrating the Vanetza project, which is the "only open-source implementation of the ETSI C-ITS protocol suite".

Figure 24 presents an overview of the Artery software architecture. Referencing back to chapter 3, Artery's stack contribution on top of Veins and Vanetza, can be translated as the Facilities and Applications layer of the ETSI ITS-G5 protocol. "ITS-G5 Middleware" is the key component in this tool, since it provides the necessary interfaces between the SUMO application and its required running service(s), i.e CAM sending between OMNet++ nodes translating into communication between vehicles in the traffic scenario on SUMO.

Figure 24: *Artery Architecture[31]*



The three concepts presented between the ITS-G5 Middleware and the Services deployed are specified to be three main methods for message sending/receiving by the Services provided. The "trigger" method works as the way for the facility existent to be able to update the vehicle current status(i.e speed, heading) so the Service can be updated through the time in order to decide whether or not he should decide to send a message.

This message providing is done by the "request" method that works as the way for the running service to notify the lower layers that it has intent of sending a message into its surroundings (i.e CAM broadcasting). Obviously this message will be queued, and transmitted into the physical channel whenever the Access layer allows it.

The "indicate" method between the Middleware and an existing running service, is the, as it name states, indication from the lower layers into the Service that a message from other C-ITS member was received and lets it access to its contents. Translating into a easier explanation, is the way a Vehicle can have access to data coming from another vehicle in the case of a CAM service running.

Different kinds of services, can be implemented using this topology, however, both Artery and COPADRIVe only have fully compliancy with the CAM service, from now on mentioned as CAService (Cooperative Awareness Service). Mostly, because different kind of messaging services are still under development for ETSI ITS-G5, even DENM messages are still, not fully defined, since they can fit a wide variety of possible scenarios, and development is still ongoing.

4.3 Robotic Simulators

4.3.1 Webots

Webots [35] is a open-source (since 2019) Robotic simulator developed by the EPFL since 1996. It claims itself as a a complete development environment to model, program and simulate robots. Different members of both Industry and Academia are users of this software for robotic simulation and modelling, the company behind Webots development offers, as well, consulting to develop and provide simulation models off-the-shelf.

Different multi-agent robotic scenarios have been the focus of most of the work and models done under Webots, however, lately, autonomous cars and different vehicular scenarios involving autonomous vehicles have been a big target of the tool development.

Webots has a lot of great features for this kind of simulation scenarios, accurate physical properties representation is their great advantage for the robotics community. Multiple plugins and APIs are already available, the simulation control models can be written as C, C++, Python and Java applications, plugins to integrate Webots with Matlab and even ROS are as well, developed. Webots has, as well, several import features, such as, import 3D models from major modeling software (SolidWorks, AutoCAD, Blender, etc.) and import maps from OpenStreetMap, GoogleMaps... Which makes it a great.

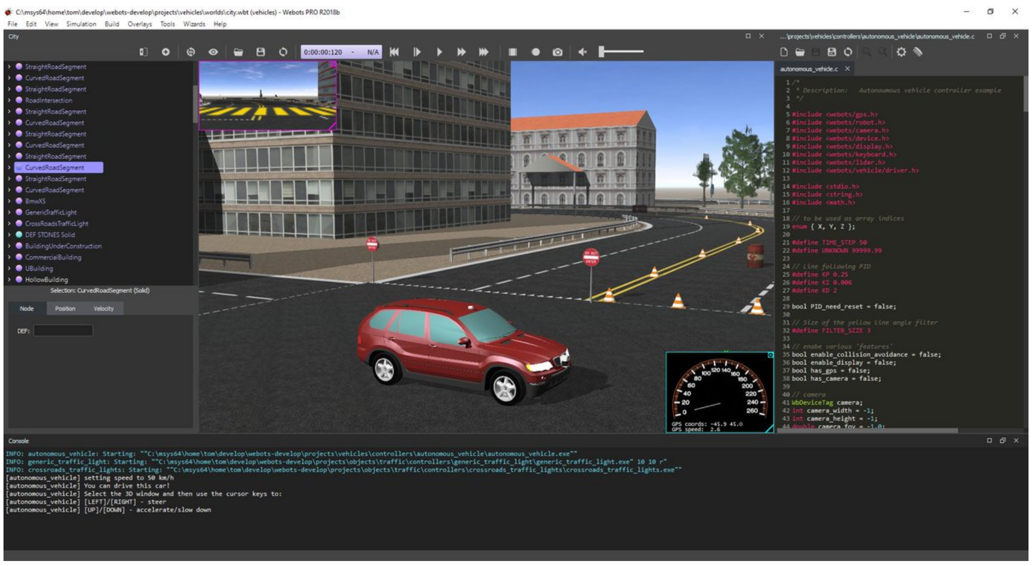
The webots GUI is shown at figure 25. At figure 26 a screen shot from a Webots simulation modelling a famous NASA robot is presented.

4.3.2 Gazebo

Gazebo[36] started to get developed in 2002 at the University of Southern California, it is a ROS-based robotic simulator, which means it isn't possible to decouple it from ROS, Gazebo uses ROS features in many ways to achieve proper simulation synchronization and communication between different software modules present on simulation scenarios. Gazebo classifies itself as a simulator with "a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces".

Gazebo software versions are released once a year by January, currently it is at

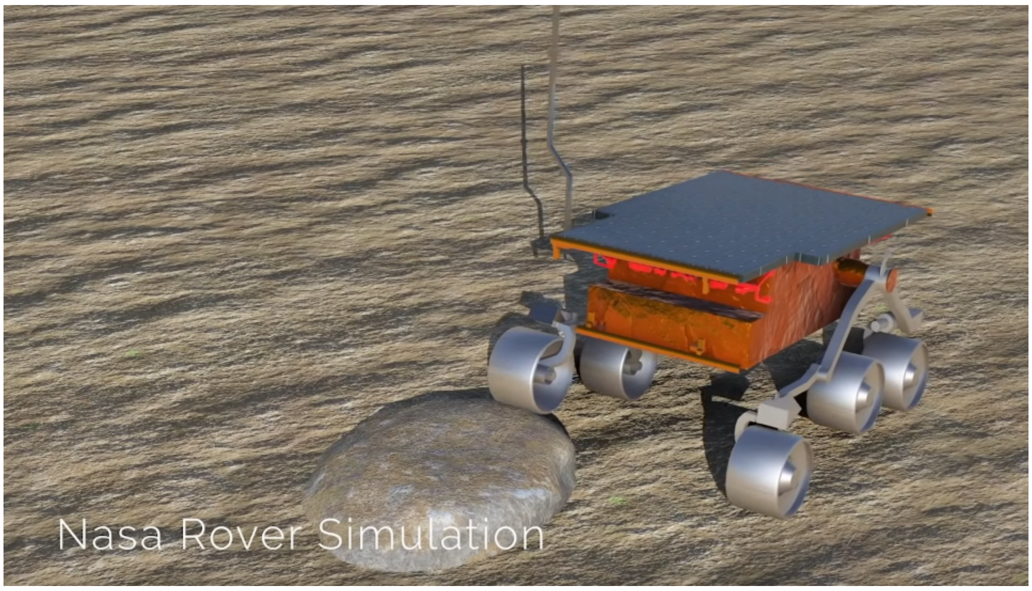
Figure 25: *Webots GUI*



```c
/*
 * Description:   Autonoumous vehicle controller example
 */

#include <webots/gps.h>
#include <webots/robot.h>
#include <webots/camera.h>
#include <webots/device.h>
#include <webots/display.h>
#include <webots/keyboard.h>
#include <webots/lidar.h>
#include <webots/vehicle/driver.h>

#include <stdio.h>
#include <string.h>
#include <math.h>

// to be used as array indices
enum { X, Y, Z };

#define TIME_STEP 50
#define UNKNOWN 99999.99

// Line following PID
#define KP 0.25
#define KI 0.006
#define KD 2

bool PID_need_reset = false;

// Size of the yellow line angle filter
#define FILTER_SIZE 3

// enabe various 'features'
bool enable_collision_avoidance = false;
bool enable_display = false;
bool has_gps = false;
bool has_camera = false;

// camera
WbDeviceTag camera;
int camera_width = -1;
int camera_height = -1;
double camera_fov = -1.0;
```

```
INFO: autonomous_vehicle: Starting: ""C:\msys64\home\tom\develop\webots-develop\projects\vehicles\controllers\autonomous_vehicle\autonomous_vehicle.exe""
INFO: generic_traffic_light: Starting: ""C:\msys64\home\tom\develop\webots-develop\projects\objects\traffic\controllers\generic_traffic_light\generic_traffic_light.exe" 10 10 r"
INFO: crossroads_traffic_lights: Starting: ""C:\msys64\home\tom\develop\webots-develop\projects\objects\traffic\controllers\crossroads_traffic_lights\crossroads_traffic_lights.exe""
[autonomous_vehicle] setting speed to 50 km/h
[autonomous_vehicle] You can drive this car!
[autonomous_vehicle] Select the 3D window and then use the cursor keys to:
[autonomous_vehicle] [LEFT]/[RIGHT] - steer
[autonomous_vehicle] [UP]/[DOWN] - accelerate/slow down
```

Figure 26: *Webots simulation*



its version number 10, version 11 is expected to be released by January/2020. It is compatible with different ROS distributions, trying to stay up-to-date with the most recent ones for each Gazebo superior version. Gazebo versions are compatible only with Ubuntu versions, for instance, Gazebo 8, the version used for the platooning model implemented for both projects developed in this thesis, is only compatible with Ubuntu's Xenial(16.04) and Yakkety(16.10) versions.

The following are the most relevant features present in this tool:

- Dynamics Simulation - Access to multiple high-performance physics engines
- Advanced 3D Graphics
- Sensors and Noise - Generate sensor data, optionally with noise, from laser range finders, 2D/3D cameras, contact sensors, force-torque, and more...
- Plugins for robots/environment control, developed with Gazebo's extensive API
- Robot Models already developed for off-the-shelf usage
- TCP/IP Transport for simulation running on remote servers
- Cloud Simulation on Amazon AWS and GzWeb for browser-based interaction
- Command Line Tools - most gazebo simulations are set up and run by command line tools (all of them under ROS)

Adding to this, not being a feature, Gazebo being open-source and ROS integrated brings to the table the argument of having excellent community support both for help and development of newly created scenarios openly-shared within the community. Which is very good in areas like robotics where there are a lot of similarities between environments and robotic systems/models within different scenarios. This community help and ROS integration, made Gazebo the best contender in comparison to Webots as the candidate to feature as the Robotic simulator for this thesis projects.

Figure 27, it's possible to see a mobile Robot model in a simulation scenario featuring the traces of different distance sensors(both Sonars and LRFs). While in figure 28, we can see a Toyota Prius resembled model with a wide variety of sensors deployed, this model is very similar to the one used in our project, with only some changes to the sensors present and the plugins deployed within itself.

Figure 27: *Gazebo simulation*

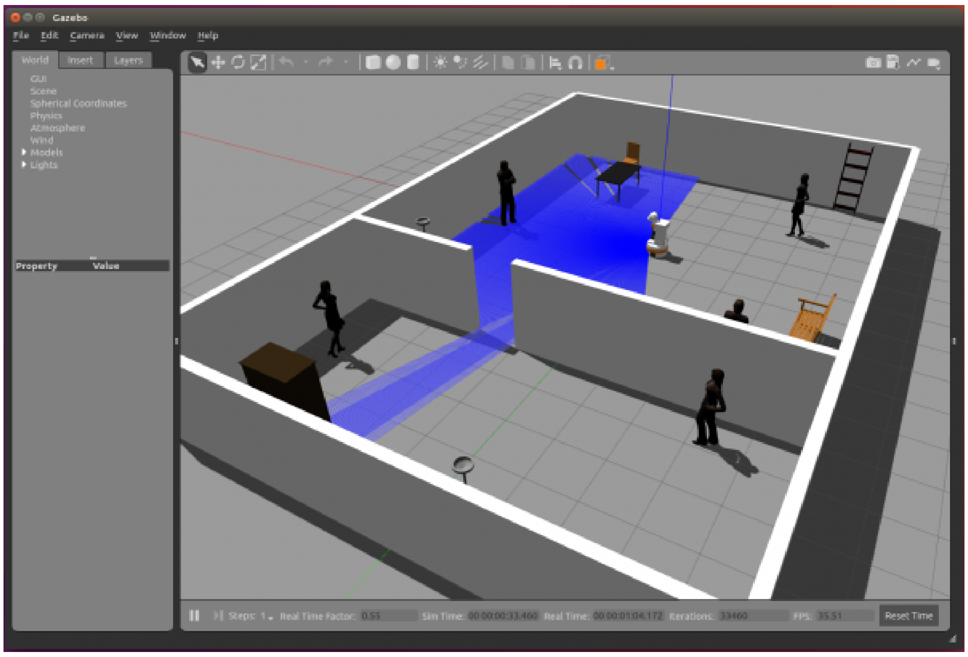
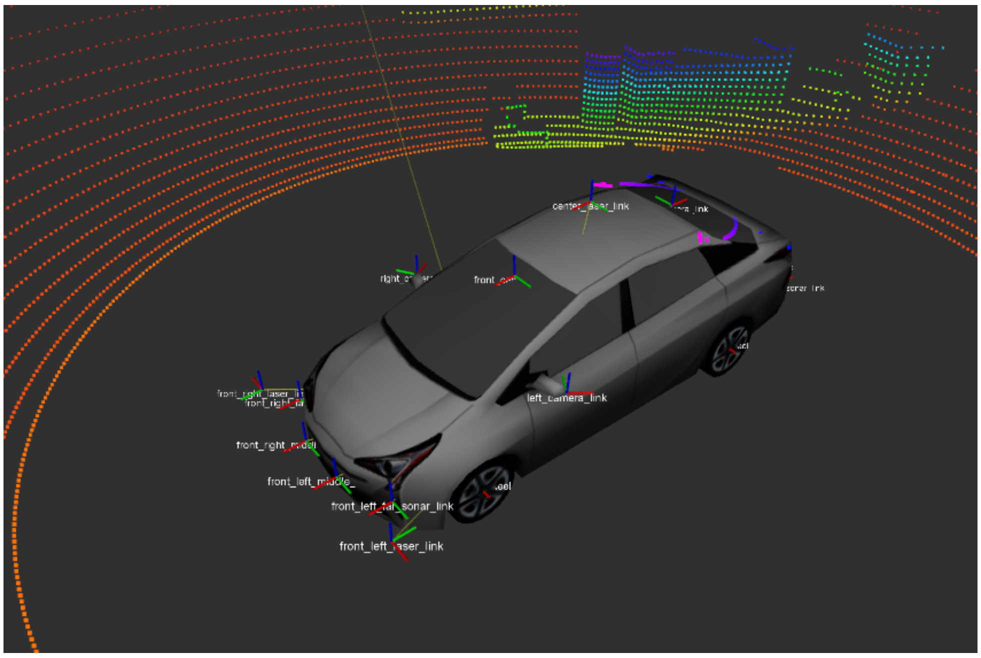

Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.55 Sim Time: 00 00:00:33.460 Real Time: 00 00:01:04.172 Iterations: 33460 FPS: 35.51
Reset Time

Figure 28: *Gazebo Prius model*



4.4 ROS - Robotic Operating System

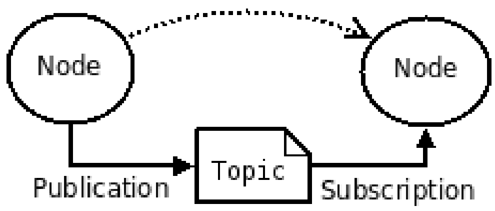
ROS (Robot Operating System) is a robotic development framework that provides libraries and tools to help developers create robot applications being them using real hardware and interconnecting different components or in simulation where all the hardware is abstracted in realistic way. It provides device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license.

The message-passing and visualizers, as well as the great community support and usage of this framework were the main reasons why COPADRIVe and the HIL simulation framework were built on top of ROS and used its main features as a way to integrate the co-simulation environment.

The following are the most relevant concepts used in ROS, and that are crucial to enable COPADRIVe and the HIL simulation framework software components:

- Nodes - Nodes are processes that perform computation. A ROS-based system usually comprises many nodes, each and every one controlling the way a computation module works.
- Messages - Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields.
- Topics - Topics' data is defined by Messages, they are routed via a transport system with publish / subscribe topology. Each node can publish and/or subscribe to any topic and exchange information with other nodes like this.
- Bags - Bags are a format for saving and replaying ROS message data. These are very important in simulation to "replay" scenarios.

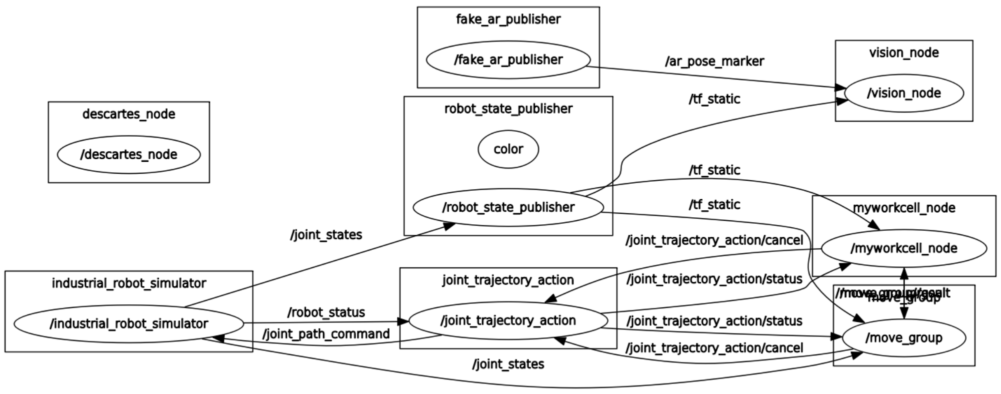
Figure 29: *ROS concepts*



In terms of application development, ROS supports code-writing in C++ and Python, some experiments were already done with Lisp and Java libraries. However, C++ and Python remain as the most used languages for ROS development.

Adding to this, the different tools provided by ROS, both command-line tools and graphical ones(rqt), as an example, rqt_graph displays an interactive graph of ROS nodes and topics(Figure 30) are a great feature for development analysis and testing.

Figure 30: *rqt_ graph example*



This subsection, is present in this thesis as a way to define and explain ROS concepts, as needed, for comprehension of the next chapters, since both implementations are done on top of this Operating System.

4.5 Conclusions

As already mentioned on the specific subsections, both projects that will follow in this thesis were developed on top of the ROS(4.4) framework with a Gazebo(4.3.2) based simulation platooning scenario, and in the case of the COPADRIVe(5) tool, OMNeT++(4.1.2) was the chosen Network simulator for the V2V communications to be deployed following the ETSI ITS-G5 standard.

5

COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications

This chapter presents the COPADRIVe - "A Realistic Simulation Framework for Cooperative Autonomous Driving Applications" - simulation tool.

It starts by overviewing the implementation of the platooning control model in the robotic simulation. Then, it presents the OMNeT++ network model and reports on the effort carried out.

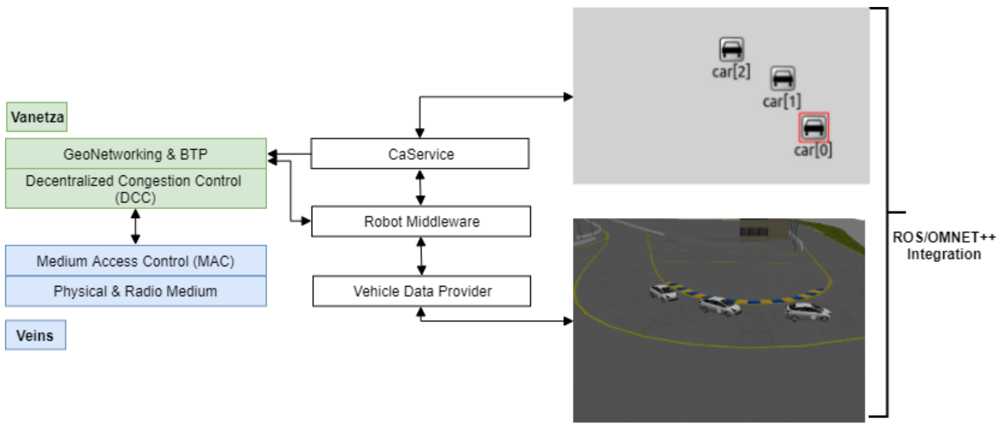
Before this chapter's conclusions, there's still a section showcasing some Experimental results and posterior analysis regarding simulation runs done on COPADRIVe and the impacts of some parameters tweaking on the Communication stack.

5.1 General System Architecture

Figure 31 shows an overview on COPADRIVe's architecture and most important modules, as well as, two screenshots of both simulators GUI's running on a co-simulation.

Starting by having a glance how this architecture compares to the Artery one (Figure 24), it's clear to see that the whole "Artery" part was replaced by a new one represented now, by the 3 most relevant modules in this integration: *CaService*,

Figure 31: *COPADRIVe architecture*



Robot Middleware and *Vehicle Data Provider.*

CaService, is the application providing the CAM service which gives vehicles the ability send CAMs whenever it is required (following the ITS-G5 standard). Robot Middleware, works as a double-sided bridge between ROS/Gazebo nodes and OMNeT++ modules, it's where cars(robots) in ROS receive a new message from their surroundings, as well as, where all the initialization necessary for the remaining OMNeT++ modules is taken care of. The Vehicle Data Provider module, is the module representing the interface between the vehicles sensors, ECUs and all data gathering components with the Networking modules, it allows OMNeT++ nodes to be aware of how their relative Gazebo car is behaving. These 3 modules will be explained in more detail on the next subsection.

The ROS/OMNet++ integration part showcased at figure 31 represents all the ROS-based applications and messages exchanged for the well-being of the COPADRIVe implementation. At section 5.4, the way OMNeT++ is integrated with Gazebo on top of the ROS framework is shown and detailed.

5.2 Gazebo Platooning Model

The Gazebo Platooning model developed, as previously stated, is based on the only-V2V reliant platooning algorithm mentioned at 2.2, featuring a platoon of 3 vehicles inside a scenario resembling a motor sports track.

Since data exchange between leader and follower of a platoon pair is the only information gathered by the follower to apply it's control algorithms, the way information is generated and exchanged on a "simple" Gazebo simulation, should be explained and mentioned into detail, in order to understand how these data work flows are comparable and reproduced on the full COPADRIVe implementation model. A rqt_graph showing all these communication links are organized is shown at Figure 32.

Analyzing Figure 32, the biggest boxes named "carX" represent the namespace for every vehicle, inside them smaller boxes represent topics published by those vehicles plugins and ellipses represent ROS nodes that might be publishing or subscribing, accordingly to the direction of their connections, to different topics.

Each vehicle has two important applications running in their node, one plugin responsible for the topics publishing, like "/carX/carINFO" that is a topic providing data regarding the current status of the vehicle like, steering angle, speed, GPS coordinates, Heading... This topic is published in a 20Hz frequency. In the instance of this model running solely with gazebo and ROS topics as the only communication mechanisms available, the topics "/carX/RXNetwork" are the ones that simulate a OBU in the vehicle that work as the way for vehicles to provide and receive data from the other vehicles, important to note that this topic is published by the plugin in a 10Hz frequency, meaning that a V2V message would be exchanged each 0.1 seconds between leader and follower in a platoon pairing.

Apart from this plugin, all the vehicles feature a control application that uses data from these topics in order to control the vehicle actuators to follow it's trajectory accordingly. In the case, of the leading vehicle (car1), it's control application is dependant on the scenario simulated, it can be autonomously controlled, manually controlled(using keyboard inputs), or even controlled using ROS Bags to be able to repeat the same trajectory as in previous simulation instances.

Figure 32: *rqt_ graph Gazebo platoon simulation*

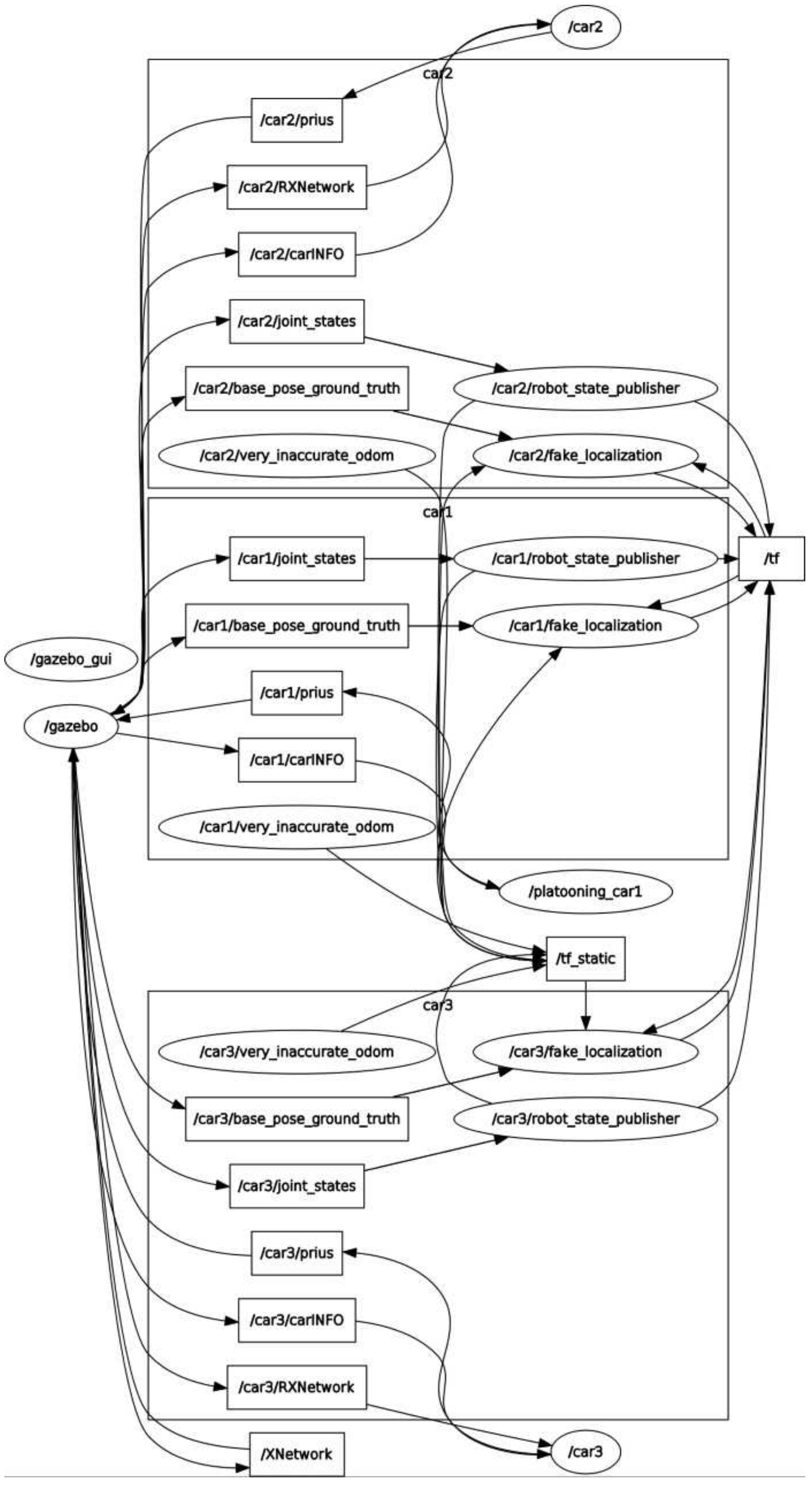


Figure 33: *Gazebo platoon simulation*

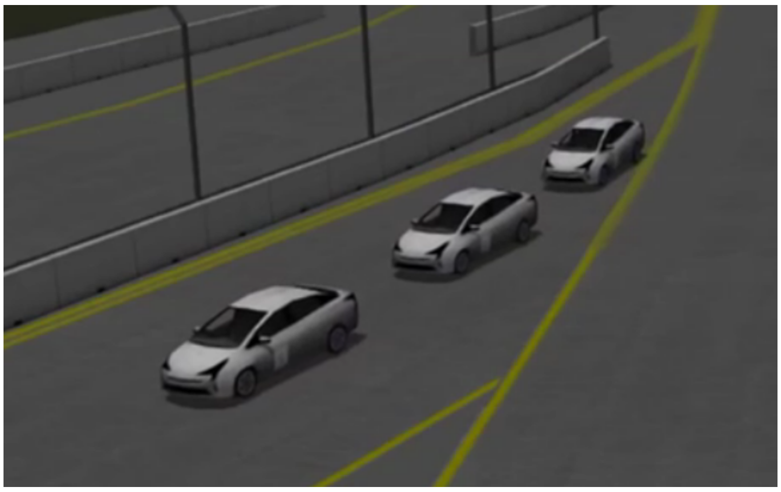


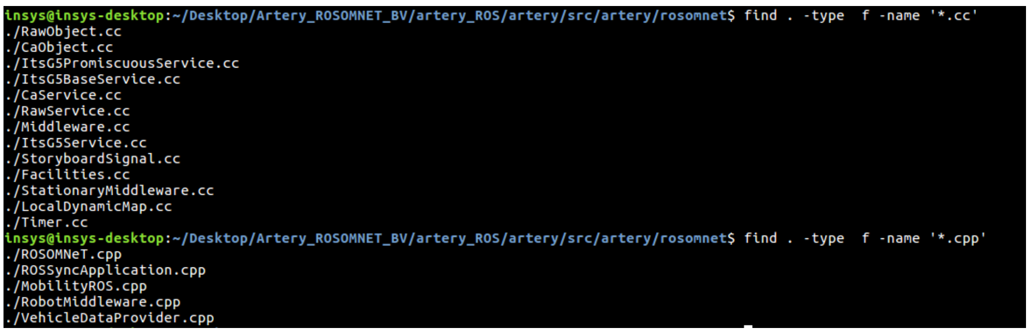
Figure 33, shows a screen shot of the three vehicles running the platoon model during a simulation run done only using Gazebo for both physical and networking simulation.

5.3 OMNeT++ Simulation Model

In this section, COPADRIVe's OMNeT++ Model will be overviewed trying, as much, to compare it to its Artery's counterparts as way to properly display the major contributions of a project against each other.

In Figure 34, the source code files for the most relevant developed modules is shown. The COPADRIVe tool was developed following the original artery projects code organization, as it's possible to see by the path used to search for these files.

Figure 34: *COPADRIVe modules*



```
insys@insys-desktop:~/Desktop/Artery_ROSOMNET_BV/artery_ROS/artery/src/artery/rosomnet$ find . -type  f -name '*.cc'
./RawObject.cc
./CaObject.cc
./ItsG5PromiscuousService.cc
./ItsG5BaseService.cc
./CaService.cc
./RawService.cc
./Middleware.cc
./ItsG5Service.cc
./StoryboardSignal.cc
./Facilities.cc
./StationaryMiddleware.cc
./LocalDynamicMap.cc
./Timer.cc
insys@insys-desktop:~/Desktop/Artery_ROSOMNET_BV/artery_ROS/artery/src/artery/rosomnet$ find . -type  f -name '*.cpp'
./ROSOMNeT.cpp
./ROSSyncApplication.cpp
./MobilityROS.cpp
./RobotMiddleware.cpp
./VehicleDataProvider.cpp
```

The functionality of the modules mentioned next, will not be detailed in this Thesis, even though their c++ code, headers and .ned files have been tweaked to be able to properly integrate within COPADRIVe's scope, since most of them only provide classes and methods for the remaining modules. Some more details on their usefulness can be found in [31] and on Artery project documentation [33].

- ItsG5Service
- Facilities
- CaObject
- LocalDynamicMap
- Middleware
- ItsG5BaseService
- StationaryMiddleware
- StoryboardSignal
- Timer

The three modules that have the prefix/suffix "ROS" within their name, are all, software modules focused on the allowance of the ROS and OMNeT++ integration to be done properly:

- ROSOMNeT
- ROSSyncApplication
- MobilityROS

All of them will be showcased at next section 5.4, as well as, all the mechanisms that made this integration possible.

Focusing now on the three main components, already previously mentioned:

- VehicleDataProvider
- RobotMiddleware
- CaService

The first one, the Vehicle Data Provider (VDP) module, as already said, is the interface between the vehicle simulated on Gazebo and its respective OMNeT++ "car" node. Within these 3 this is the only software module that isn't properly a OMNeT++ module by definition, these VDP module is an entity that is used by the

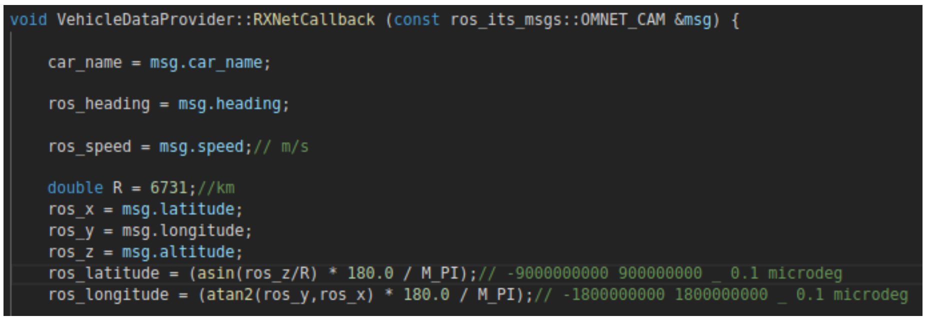
CaService, in order for this service to be able to properly fill CAM messages with the necessary (and real) information.

The VDP, receives data from the vehicle within Gazebo, by subscribing to the topic "/carX/carINFO" mentioned at section 5.2.

At Figure 35 it's shown part of the method that is called whenever new data comes from the Gazebo vehicle, only some relevant data is stored on this class variables. As we can see, the car "name" (position in the platoon), its most recent heading and speed, as well as, its current GPS coordinates are stored inside the VDP. A quick note on the GPS coordinates, since the Gazebo simulation model only supports cartesian coordinates, a "translation" was made, in order to fill the GPS fields defined on the ETSI ITS-G5 CAM message that are mandatory to be filled using GNSS latitude and longitude values, since the Gazebo model is small and neutral on the Z coordinate, we assume the CAM altitude value is unavailable.

By using the MobilityROS module functionalities, VDP is responsible to update the car's OMNeT++ mobility module accordingly to data from Gazebo.

Figure 35: *VDP topic callback*



```
void VehicleDataProvider::RXNetCallback (const ros_its_msgs::OMNET_CAM &msg) {

    car_name = msg.car_name;

    ros_heading = msg.heading;

    ros_speed = msg.speed;// m/s

    double R = 6731;//km
    ros_x = msg.latitude;
    ros_y = msg.longitude;
    ros_z = msg.altitude;
    ros_latitude = (asin(ros_z/R) * 180.0 / M_PI);// -9000000000 900000000 _ 0.1 microdeg
    ros_longitude = (atan2(ros_y,ros_x) * 180.0 / M_PI);// -1800000000 1800000000 _ 0.1 microdeg
```

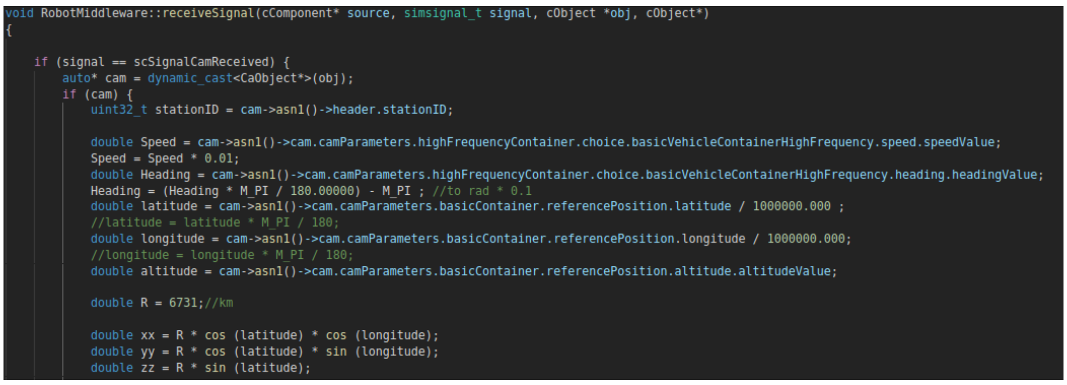
The Robot Middleware module, in this case, it is a OMNeT++ simple module by definition extending Artery's "Middleware" module is responsible to provide typical ITS-G5 Facilities middleware capabilities.

In case of the COPADRIVe implementation, it is the module responsible to update the car module position during simulation-time using the mobility module that gets updated by the VDP.

Apart from this, it is responsible, as well, by receiving and storing data that come from other vehicle's CAM messages. Following this, this data is sent to ROS by means of a topic publishing, that gets subscripted by its relative Gazebo vehicle. In figure 36 a piece of code of the method that receives a signal, confirms it is a message in CAM format and stores some of its data into local variables.

The CaService module, it states for "Cooperative Awareness Service. This service, uses the VDP class as the it's way of receiving the most recent status from the

Figure 36: *Robot Middleware code*



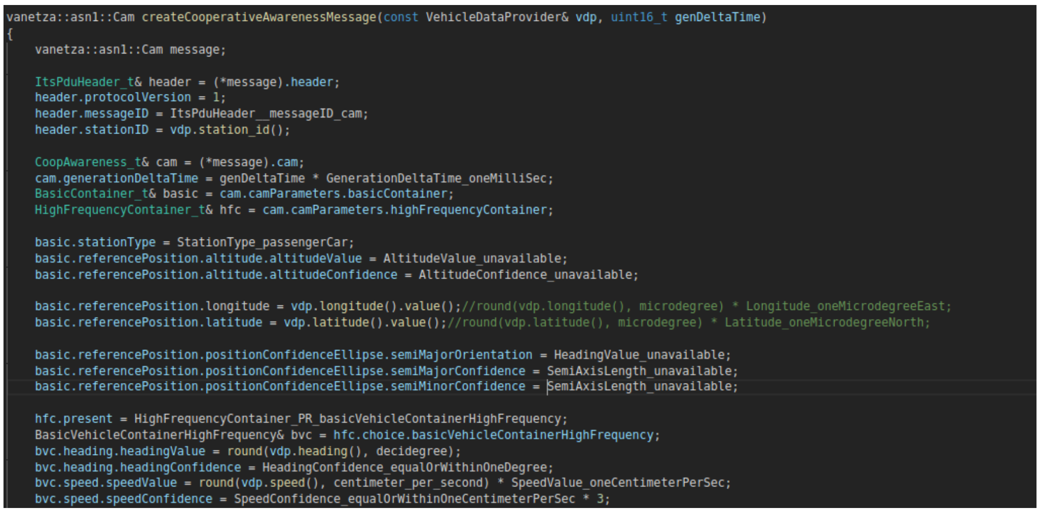
```
void RobotMiddleware::receiveSignal(cComponent* source, simsignal_t signal, cObject *obj, cObject*)
{

    if (signal == scSignalCamReceived) {
        auto* cam = dynamic_cast<CaObject*>(obj);
        if (cam) {
            uint32_t stationID = cam->asn1()->header.stationID;

            double Speed = cam->asn1()->cam.camParameters.highFrequencyContainer.choice.basicVehicleContainerHighFrequency.speed.speedValue;
            Speed = Speed * 0.01;
            double Heading = cam->asn1()->cam.camParameters.highFrequencyContainer.choice.basicVehicleContainerHighFrequency.heading.headingValue;
            Heading = (Heading * M_PI / 180.00000) - M_PI ; //to rad * 0.1
            double latitude = cam->asn1()->cam.camParameters.basicContainer.referencePosition.latitude / 1000000.000 ;
            //latitude = latitude * M_PI / 180;
            double longitude = cam->asn1()->cam.camParameters.basicContainer.referencePosition.longitude / 1000000.000;
            //longitude = longitude * M_PI / 180;
            double altitude = cam->asn1()->cam.camParameters.basicContainer.referencePosition.altitude.altitudeValue;

            double R = 6731;//km

            double xx = R * cos (latitude) * cos (longitude);
            double yy = R * cos (latitude) * sin (longitude);
            double zz = R * sin (latitude);
```

vehicle data fields.

The two main features of this service are what data to send on CAM messages and when to send them.

In order to fulfill the required data it uses the VDP gathered information, as shown in figure 37

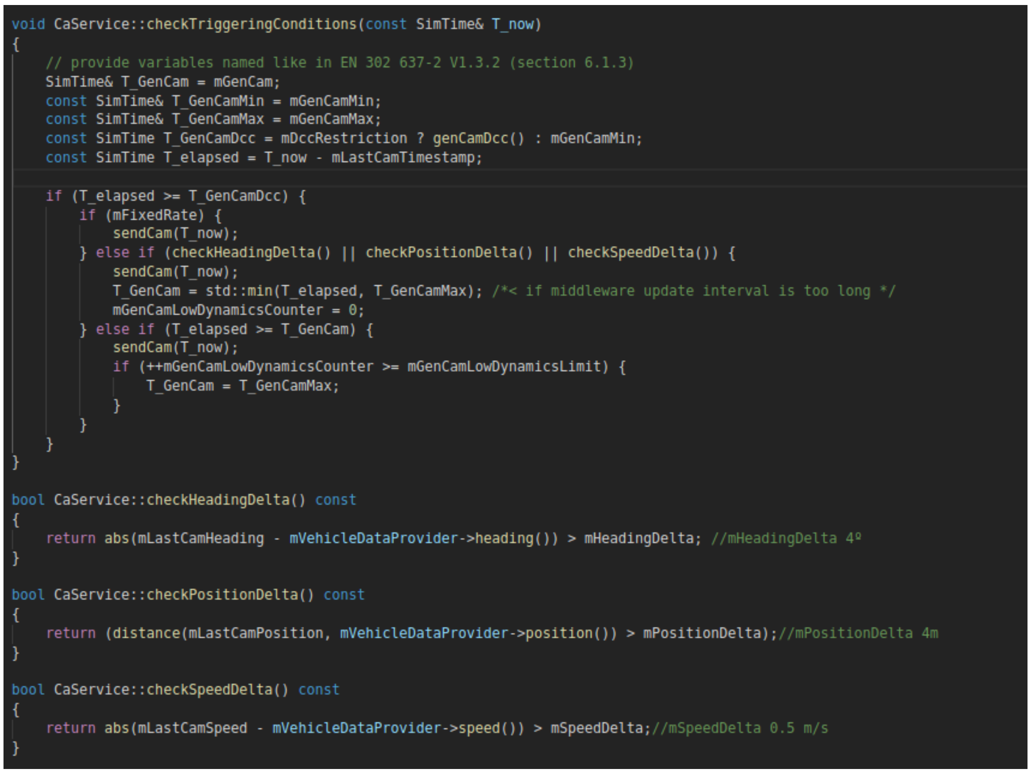
Figure 37: *createCooperativeAwarenessMessage method*



```
vanetza::asn1::Cam createCooperativeAwarenessMessage(const VehicleDataProvider& vdp, uint16_t genDeltaTime)
{
    vanetza::asn1::Cam message;

    ItsPduHeader_t& header = (*message).header;
    header.protocolVersion = 1;
    header.messageID = ItsPduHeader__messageID_cam;
    header.stationID = vdp.station_id();

    CoopAwareness_t& cam = (*message).cam;
    cam.generationDeltaTime = genDeltaTime * GenerationDeltaTime_oneMilliSec;
    BasicContainer_t& basic = cam.camParameters.basicContainer;
    HighFrequencyContainer_t& hfc = cam.camParameters.highFrequencyContainer;

    basic.stationType = StationType_passengerCar;
    basic.referencePosition.altitude.altitudeValue = AltitudeValue_unavailable;
    basic.referencePosition.altitude.altitudeConfidence = AltitudeConfidence_unavailable;

    basic.referencePosition.longitude = vdp.longitude().value();//round(vdp.longitude(), microdegree) * Longitude_oneMicrodegreeEast;
    basic.referencePosition.latitude = vdp.latitude().value();//round(vdp.latitude(), microdegree) * Latitude_oneMicrodegreeNorth;

    basic.referencePosition.positionConfidenceEllipse.semiMajorOrientation = HeadingValue_unavailable;
    basic.referencePosition.positionConfidenceEllipse.semiMajorConfidence = SemiAxisLength_unavailable;
    basic.referencePosition.positionConfidenceEllipse.semiMinorConfidence = SemiAxisLength_unavailable;

    hfc.present = HighFrequencyContainer_PR_basicVehicleContainerHighFrequency;
    BasicVehicleContainerHighFrequency& bvc = hfc.choice.basicVehicleContainerHighFrequency;
    bvc.heading.headingValue = round(vdp.heading(), decidegree);
    bvc.heading.headingConfidence = HeadingConfidence_equalOrWithinOneDegree;
    bvc.speed.speedValue = round(vdp.speed(), centimeter_per_second) * SpeedValue_oneCentimeterPerSec;
    bvc.speed.speedConfidence = SpeedConfidence_equalOrWithinOneCentimeterPerSec * 3;
```

To be able to specify when a CAM message should be sent or not, the CaService follows a specified profile to trigger this message sending. According to the ETSI ITS-G5 the Basic Service Profile (BSP) provides some threshold limited rules in order for this service to decide whether or not a CAM message shall be sent, these limits for the BSP were specified at 3.2. Figure 38, shows the methods responsible for this.

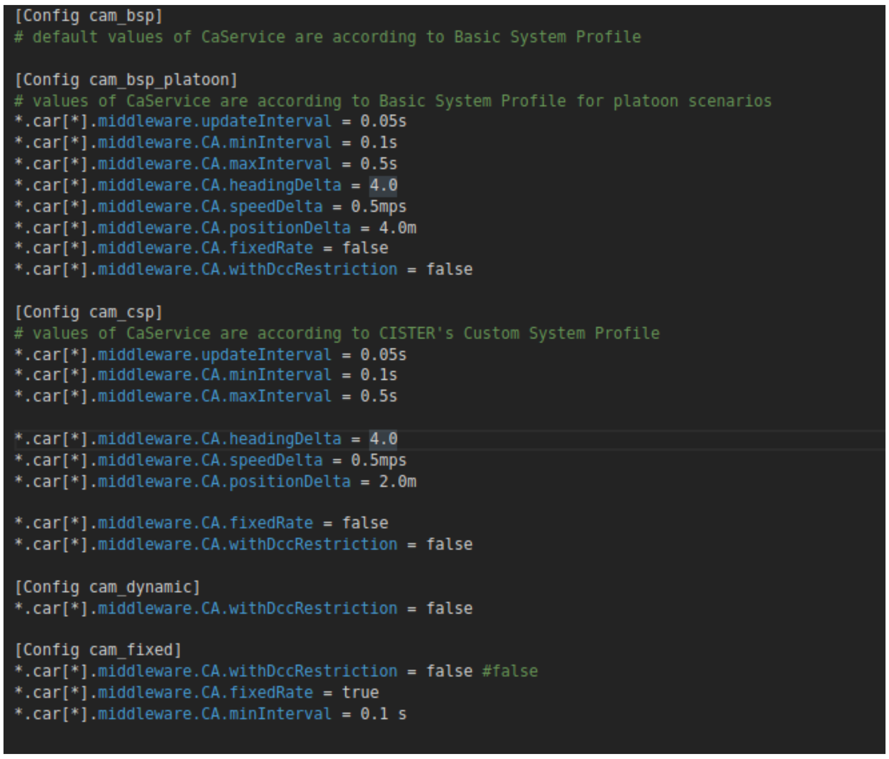
Figure 38: *CAM trigger profiling*



```cpp
void CaService::checkTriggeringConditions(const SimTime& T_now)
{
    // provide variables named like in EN 302 637-2 V1.3.2 (section 6.1.3)
    SimTime& T_GenCam = mGenCam;
    const SimTime& T_GenCamMin = mGenCamMin;
    const SimTime& T_GenCamMax = mGenCamMax;
    const SimTime T_GenCamDcc = mDccRestriction ? genCamDcc() : mGenCamMin;
    const SimTime T_elapsed = T_now - mLastCamTimestamp;

    if (T_elapsed >= T_GenCamDcc) {
        if (mFixedRate) {
            sendCam(T_now);
        } else if (checkHeadingDelta() || checkPositionDelta() || checkSpeedDelta()) {
            sendCam(T_now);
            T_GenCam = std::min(T_elapsed, T_GenCamMax); /*< if middleware update interval is too long */
            mGenCamLowDynamicsCounter = 0;
        } else if (T_elapsed >= T_GenCam) {
            sendCam(T_now);
            if (++mGenCamLowDynamicsCounter >= mGenCamLowDynamicsLimit) {
                T_GenCam = T_GenCamMax;
            }
        }
    }
}

bool CaService::checkHeadingDelta() const
{
    return abs(mLastCamHeading - mVehicleDataProvider->heading()) > mHeadingDelta; //mHeadingDelta 4º
}

bool CaService::checkPositionDelta() const
{
    return (distance(mLastCamPosition, mVehicleDataProvider->position()) > mPositionDelta);//mPositionDelta 4m
}

bool CaService::checkSpeedDelta() const
{
    return abs(mLastCamSpeed - mVehicleDataProvider->speed()) > mSpeedDelta;//mSpeedDelta 0.5 m/s
}
```

The thresholds present at the above methods, can be adjusted in a customized way by the user, by editing the .ini file that launches this simulation. Figure 39 shows the parameters present at this file, that should be used to change the profile set to use by the CaService. Already organized as different configured profiles set for testing.

Figure 39: *.ini file Profiling*



```
[Config cam_bsp]
# default values of CaService are according to Basic System Profile

[Config cam_bsp_platoon]
# values of CaService are according to Basic System Profile for platoon scenarios
*.car[*].middleware.updateInterval = 0.05s
*.car[*].middleware.CA.minInterval = 0.1s
*.car[*].middleware.CA.maxInterval = 0.5s
*.car[*].middleware.CA.headingDelta = 4.0
*.car[*].middleware.CA.speedDelta = 0.5mps
*.car[*].middleware.CA.positionDelta = 4.0m
*.car[*].middleware.CA.fixedRate = false
*.car[*].middleware.CA.withDccRestriction = false

[Config cam_csp]
# values of CaService are according to CISTER's Custom System Profile
*.car[*].middleware.updateInterval = 0.05s
*.car[*].middleware.CA.minInterval = 0.1s
*.car[*].middleware.CA.maxInterval = 0.5s

*.car[*].middleware.CA.headingDelta = 4.0
*.car[*].middleware.CA.speedDelta = 0.5mps
*.car[*].middleware.CA.positionDelta = 2.0m

*.car[*].middleware.CA.fixedRate = false
*.car[*].middleware.CA.withDccRestriction = false

[Config cam_dynamic]
*.car[*].middleware.CA.withDccRestriction = false

[Config cam_fixed]
*.car[*].middleware.CA.withDccRestriction = false #false
*.car[*].middleware.CA.fixedRate = true
*.car[*].middleware.CA.minInterval = 0.1 s
```

On the next section, the interface/integration developed to properly implement this project will be shown and detailed, together with a description of how the previously defined modules interact with it.

5.4 ROS-OMNeT++ Interface

5.4.1 Synchronization Approach

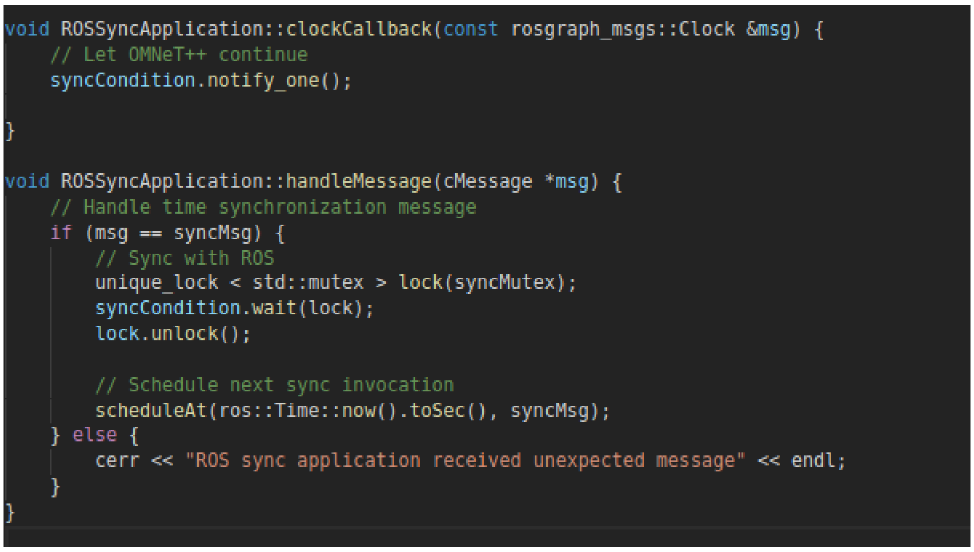
For this Gazebo/ OMNeT++ integration to be done, the first crucial step that had to be implemented was the synchronization between both simulators' clocks, if this wasn't accomplished, the platoon control could be compromised, even though the messages were being exchanged in OMNeT++.

OMNeT++ is a discrete-event simulator, which means that its internal clock only "advances" if any simulation event happens (i.e a message being sent), and Gazebo has a clock based on time-steps, which means its clock "advances" in a fixed frequency/period according to the defined time-step (1ms in COPADRIVe's case). Because of this difference on both clocks, a methodology to make them synchronized had to be implemented.

The previously mentioned ROSSyncApplication module, was developed with this in mind, it is a simple OMNeT++ module that runs on the same hierarchy simulation level as the vehicular nodes.

Its functionality is quite simple, after initialization, it sends a internal message "syncMsg", created only for this purpose, at clock time 00:00:00. Since it is the only module receiving this message, it handles it, by locking a mutex, making all the simulation stop, since it can only advance if any kind of event is scheduled. By subscribing to the ROS "/Clock" topic, that gets published every 1ms, whenever it gets to the callback method of this topic it unlocks the mutex, enabling the previously mentioned method to advance and schedule a new message to the current Gazebo/ROS timestamp, repeating this cycle again. At figure 40, the code for the methods that guarantee this synchronization loop gets going, is shown.

Figure 40: *Clock syncronization approach between OMNeT++ and Gazebo*



```
void ROSSyncApplication::clockCallback(const rosgraph_msgs::Clock &msg) {
    // Let OMNeT++ continue
    syncCondition.notify_one();

}

void ROSSyncApplication::handleMessage(cMessage *msg) {
    // Handle time synchronization message
    if (msg == syncMsg) {
        // Sync with ROS
        unique_lock < std::mutex > lock(syncMutex);
        syncCondition.wait(lock);
        lock.unlock();

        // Schedule next sync invocation
        scheduleAt(ros::Time::now().toSec(), syncMsg);
    } else {
        cerr << "ROS sync application received unexpected message" << endl;
    }
}
```

5.4.2 Interface Architecture

Apart from the time synchronization between simulators, ROS is the framework base on how both simulators can exchange data. Already some examples on how ROS enables this were done on the modules description done on the previous section.

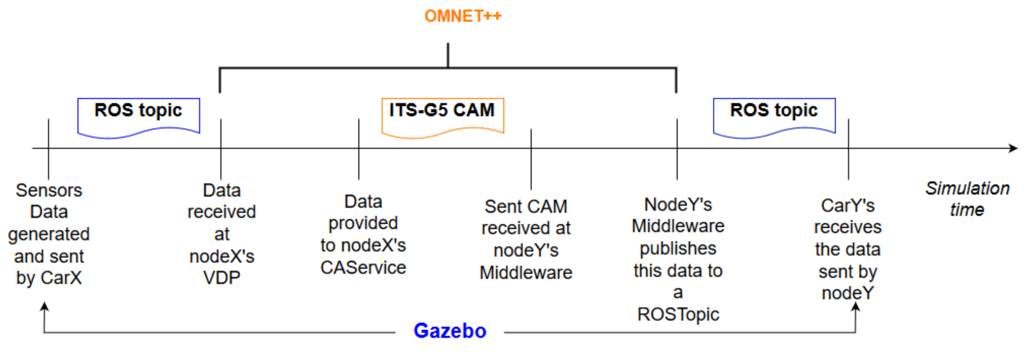
Starting by mentioning the remaining software modules still not defined: ROSOMNeT and MobilityROS.

The first one, is a class only providing some ROS functionalities for module development that might need to use ROS integration for their usefulness to be fulfilled (i.e CaService), within the methods provided by this module is worth to mention, the rosMain and runROSNode methods that are used by all the OMNeT++ modules that require ROS integration. rosMain is used as the ROS spin loop that is usually used on ROS applications, runROSNode is used by these modules to initialize, handle and create a rosNode whenever a module requires it.

MobilityROS, is the software module that guarantees that any OMNeT++ module that is able to access GPS coordinates from gazebo vehicles (subscribing to "/carX/carINFO") can set the OMNeT++ representative module to be positioned correctly on the simulation UI.

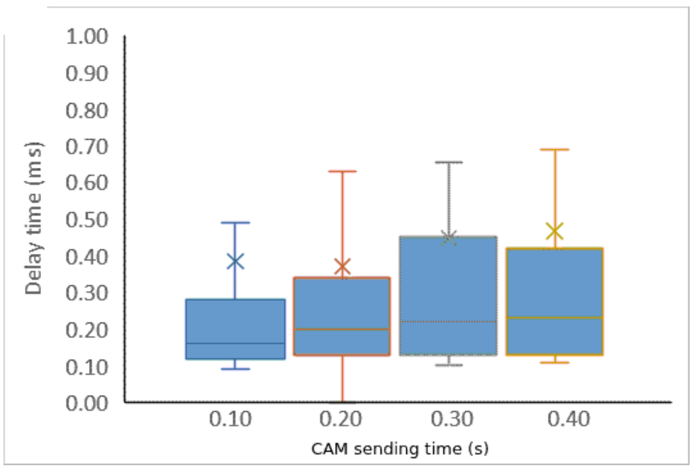
In terms of how and when data moves around this tool software stack, figure 41 has a good over view on it. Data flows from carX's sensors into carY's control application, working its way through Gazebo into OMNeT++ and then into other Gazebo's car following a CAM transmission between different nodes in OMNeT++. To note that nodeX and nodeY represent the network interface of both carX and carY, respectively.

Figure 41: *Data Workflow*


OMNET++
ROS topic
ITS-G5 CAM
ROS topic
Sensors Data generated and sent by CarX
Data received at nodeX's VDP
Data provided to nodeX's CAService
Sent CAM received at nodeY's Middleware
NodeY's Middleware publishes this data to a ROSTopic
CarY's receives the data sent by nodeY
Simulation time
Gazebo

To evaluate the framework's stability and limits, an analysis was done on its inherent latency and/or computing delays. Figure 42 presents the delay between an OMNeT++ node reception of a CAM (from a network transmission) and its reception by the Gazebo's vehicle model, after receiving it via ROS Pub/Sub mechanisms. Following Figure 41 time line, the time stamps recorded were taken at CAM reception at the node's Middleware and upon Gazebo's car application callback on this referred ROS topic, at different CAM sending frequencies (10, 5, 3.3 and 2.5 Hz). From what we can extract from the recorded data, this delay mostly coming from the ROS underlying Pub/Sub mechanisms, doesn't seem to be severely affected by the CAM sending frequency. Despite the maximum obtained delay slightly increased with traffic, the observed latency close to 0.25 milliseconds, is not sufficient to impact or compromise the application under test.

Figure 42: *Implementation delay impact on CAM exchanging*



5.5 Conclusions

The COPADRIVe project development was challenging in many different ways, but, apart from software development, many different concepts, tools and standards had to be well understood, in order to be able to produce this simulation tool. As already stated, this is a kind of co-simulation tool that didn't exist till this date, and that provides capabilities to support C-ITS systems. This tool has an immense potential for analyzing different Network and Control models pending forward research in C-ITS scenarios. At section 7.1 some results will be shown using this tool, while analyzing, different Network settings.

6 Hardware-in-the-Loop simulation framework for Cooperative Platooning safety assurance

6.1 Scope

In this chapter, a hardware-in-the-loop simulation architecture for C-ITS scenarios with real On-board-unit hardware will be presented. Since COPADRIVe can fully simulate both the communications between different C-ITS nodes and join it with microscopic level analysis of control scenarios, slowly transitioning into real-life implementations is the next step towards complete validation of these use-cases.

One of the challenges of SAFECOP was to improve the safety of cooperative platooning. If on the one hand COPADRIVe allowed us to measure and evaluate the performance of a network and control strategy, on the other hand, to address the safety issue, it was crucial to implement reactive services that could monitor the integrity of the platooning. To do this, GMV Skysoft, S.A. proposed a CLW (6.2), a piece of monitoring software that overviews the safety of the cooperative platooning system and triggers an emergency state upon detecting an error. This mechanism was tested and continuously developed with support from COPADRIVe.

To achieve this, a hardware-in-the-loop simulation framework was developed as a way to simulate a Platooning scenario under the Gazebo simulator and integrating it with Cohda's MK5 On-board-units in order to get the communications between vehicles to be done, by relying on real OBUs and compliant with the ITS-G5 standard.

Regarding the CLW some of the software used within this project context is proprietary by GMV Skysoft S.A. and will not be fully disclosed. However, the primarily tested out software components on this tool will be presented at section 6.2, together with the Runtime monitor software framework developed by CISTER.

6.2 Control Loss Warning / Runtime Monitor

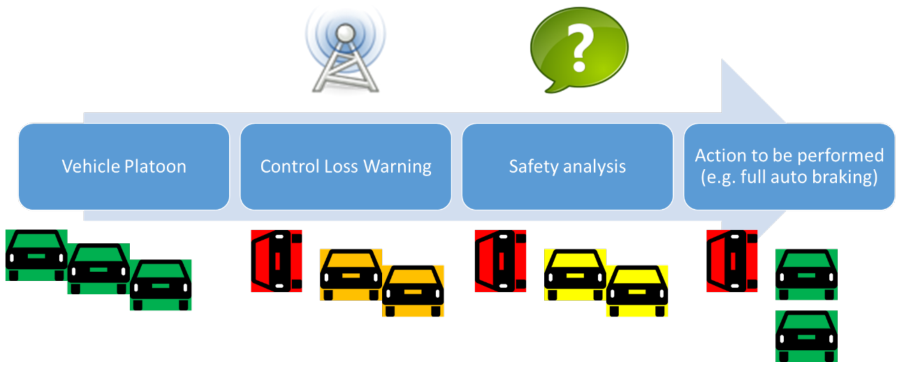
The Control Loss Warning (CLW)(figure 43) and Runtime Monitor (RTM) systems are designed to detect and alert about control loss situations. A CLW/RTM system has the ability to monitor system's status, and trigger alerts if a control loss situation is detected. CLW/RTM mechanisms are particularly useful to monitor individual nodes that work together to create a complex autonomous system. The main difference between these systems is on which part of this project component these mechanisms are running.

The RTM works on the computer running the Gazebo simulation, since it tracks and monitors the data being published on ROS topics published by each vehicles nodes, the RTM can be running, as well, on a different computer that has access to the computer running the ROS system, however this can bring some excessive delays on the alerting system.

The CLW is a software module running on the OBU, that receives data from the ETSI module that come either from its dedicated vehicle or from other OBUs ackowledging the position and physical properties of the other platoon members. The CLW module uses and compares this data to be able to properly detect (or even predict) a failure (e.g loss of brakes).

As these system are appropriate to monitor the status of individual nodes, they can be applied to a platoon of vehicles travelling along a motorway.

Figure 43: *Control Loss Warning*



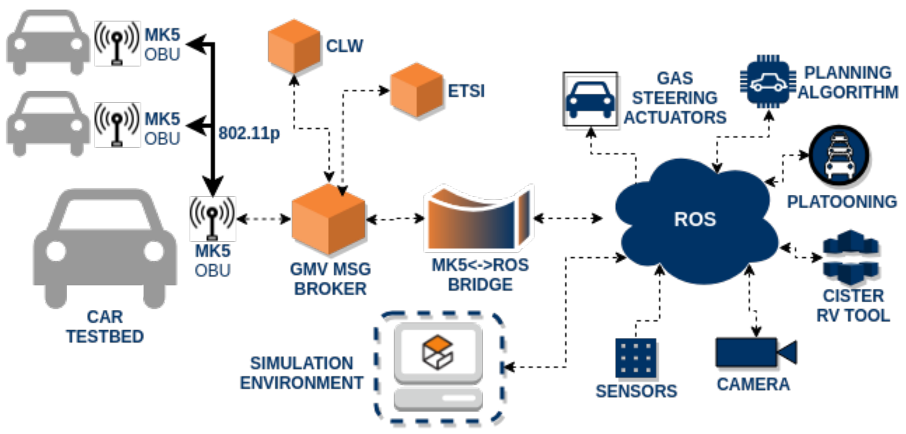
6.3 Testbed Architecture

The presented system (figure 44) architecture refers to the fully implemented cooperative tool, where the robots running at Gazebo, plan and run a platoon, following the platoon model defined at 2.2 with V2V communications being accomplished by the OBUs where the "CLW" and "ETSI" software modules guarantee safety and security compliancy with the ITS-G5 standard.

The OBUs connect to the vehicles ROS-based control systems, via the MK5-ROS bridge, using TCP sockets, this sockets were tested using IPv4 on top of CISTER's internal network access, different connection tests were made, like connecting directly the PC running the ROS-based simulation through Ethernet to the OBU.

On the OBU side, the received messages are processed by the message broker and used by the above modules. On the vehicles' side, the ROS-based control system, uses the information received through the bridge to get a good awareness of its surrounding environment and cars involved in the platoon. Here, the ROS system serve as the framework that support the different control algorithms to enable the platooning functionality, connecting sensors and actuators, the platooning module, and the Runtime Monitor that is used to ensure the overall safety of the system.

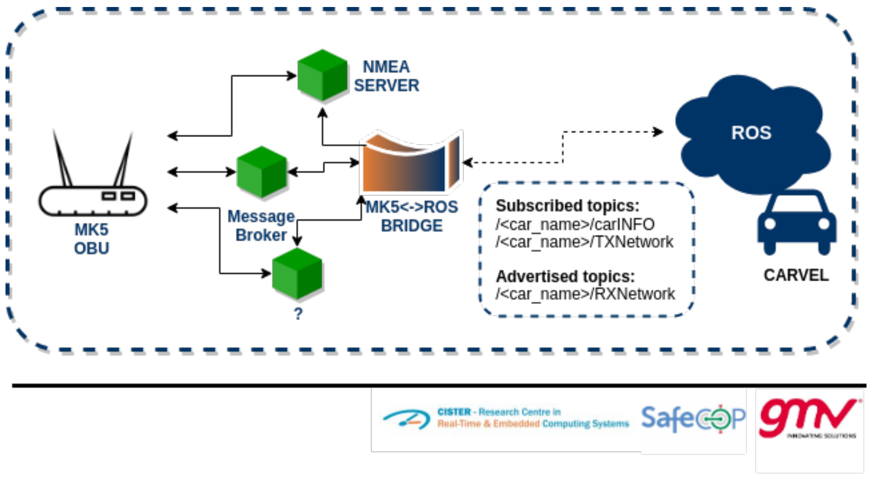
Figure 44: *HIL Testbed Architecture*



The ROS_MK5_bridge provides a bi-directional bridge between ROS systems implementing a platoon simulation model running on Gazebo, and the MK5 OBUs from Cohda Wireless. This allows a ROS environment to connect and communicate with other ROS environments through MK5 802.11p platforms. An overview on the ROS_MK5_bridge's main interfaces is presented at figure 45.

In the previous diagram, ROS/CARVEL represent the vehicle platooning environment, simulated on Gazebo, this Bridge end-side subscribes to topics coming from

Figure 45: *Bridge Architecture*



the simulator, in order to provide their information through the bridge to feed the OBUs. In the opposite direction, it also, advertises a topic filled with information coming from the Bridge, so the simulator can properly acknowledge this.

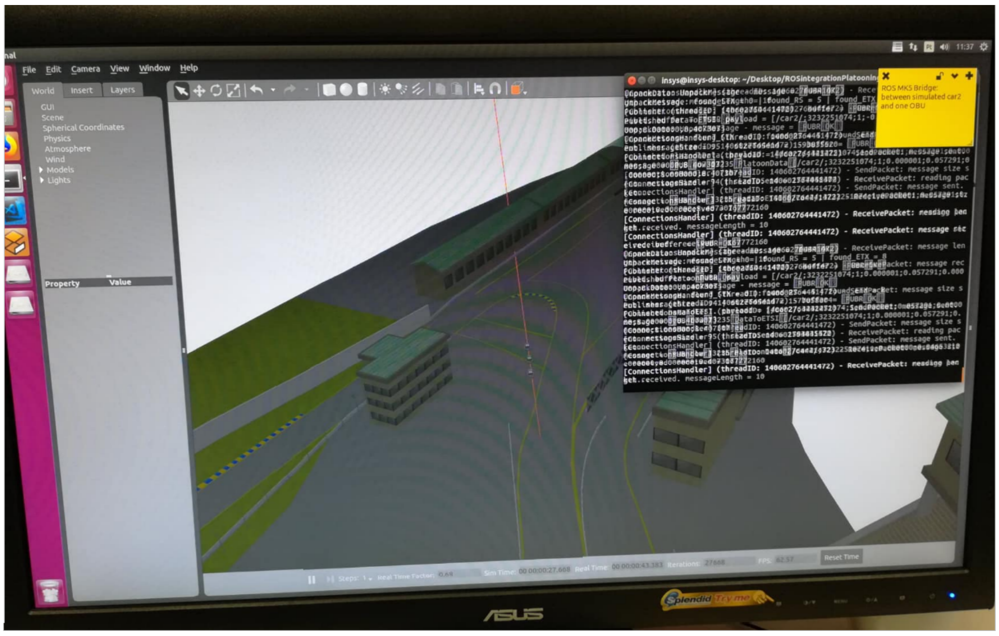
On the OBUs end-side of the Bridge, all the information coming from it is fed into the MessageBroker module, which segregates and processes this data, in order to, feed it into the remaining OBU modules. NMEA messages sent through the Bridge are used to provide GPS coordinates, speed and heading of the correspondent vehicle that is feeding the Bridge, the NMEA Server running in the OBUs computes them, so their information can be used by the remaining MK5 ran applications, this module replaces the existent NMEA server present on the MK5 by default, simulating the positioning of the OBUs on the simulator "world". If this wasn't done, OBUs, since they're indoor would not be able to get access to their GPS positioning and so, most of their software modules wouldn't be able to properly work.

The module represented by the question mark "?", represents all the software module present at this system that can use information received by the MessageBroker modules, in the case of this simulation tool, the CLW and the ETSI module previously stated are obvious examples of this.

Figures 46 and 47 show screen captures of a HIL simulation running with some ssh connections required done into two OBUs that are connected through the bridge into the computer running the gazebo simulation and communicating between them with data coming from the vehicles on the simulator. Figure 48 shows, the hardware setup of OBUs with a simulation run being streamed.

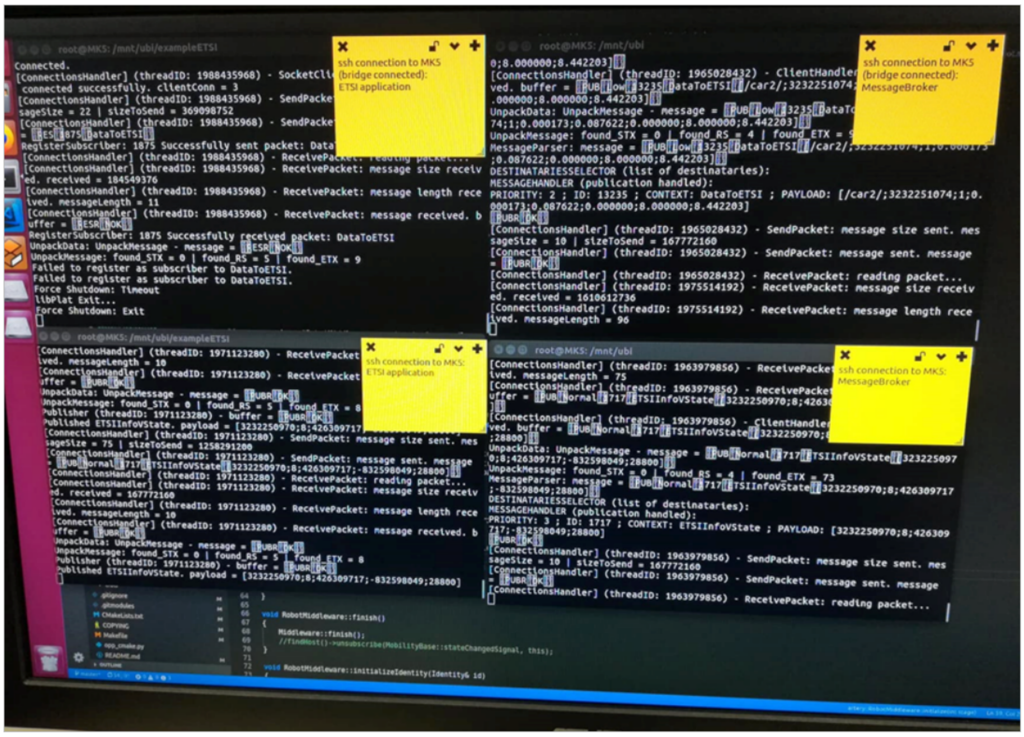
The CLW module will be explained on the next section, the ETSI module is responsible for gathering data coming from, either the ROS_MK5_Bridge associated

Figure 46: *HiL simulation w/ ROS_ MK5_ bridge*


insys@insys-desktop: ~/Desktop/ROSintegrationPlatooning
ROS MK5 Bridge: between simulated car2 and one OBU
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Reset Time
11:37

```
[illegible]
[ConnectionsHandler] (threadID: 140602764441472) - ReceivePacket: reading pac
ket.received. messageLength = 10
[ConnectionsHandler] (threadID: 140602764441472) - ReceivePacket: message rec
eived. buffer = UBR OK 67772160
[illegible]
[ConnectionsHandler] (threadID: 140602764441472) - ReceivePacket: reading pac
ket.received. messageLength = 10
```

Figure 47: *SSH connections into 2 OBUs*


root@MK5: /mnt/ubi/exampleETSI
ssh connection to MK5 (bridge connected): ETSI application
root@MK5: /mnt/ubi
ssh connection to MK5 (bridge connected): MessageBroker
root@MK5: /mnt/ubi/exampleETSI
ssh connection to MK5: ETSI application
root@MK5: /mnt/ubi
ssh connection to MK5: MessageBroker

```
Connected.
[ConnectionsHandler] (threadID: 1988435968) - SocketCli
 connected successfully. clientConn = 3
[ConnectionsHandler] (threadID: 1988435968) - SendPacke
sageSize = 22 | sizeToSend = 369098752
[ConnectionsHandler] (threadID: 1988435968) - SendPacke
= ES 1875 DataToETSI
RegisterSubscriber: 1875 Successfully sent packet: Data
[ConnectionsHandler] (threadID: 1988435968) - ReceivePacket: reading packet...
[ConnectionsHandler] (threadID: 1988435968) - ReceivePacket: message size receiv
ed. received = 184549376
[ConnectionsHandler] (threadID: 1988435968) - ReceivePacket: message length rece
ived. messageLength = 11
[ConnectionsHandler] (threadID: 1988435968) - ReceivePacket: message received. b
uffer = ESR NOK
RegisterSubscriber: 1875 Successfully received packet: DataToETSI
UnpackData: UnpackMessage - message = ESR NOK
UnpackMessage: found_STX = 0 | found_RS = 5 | found_ETX = 9
Failed to register as subscriber to DataToETSI.
Failed to register as subscriber to DataToETSI.
Force Shutdown: Timeout
libPlat Exit...
Force Shutdown: Exit
```

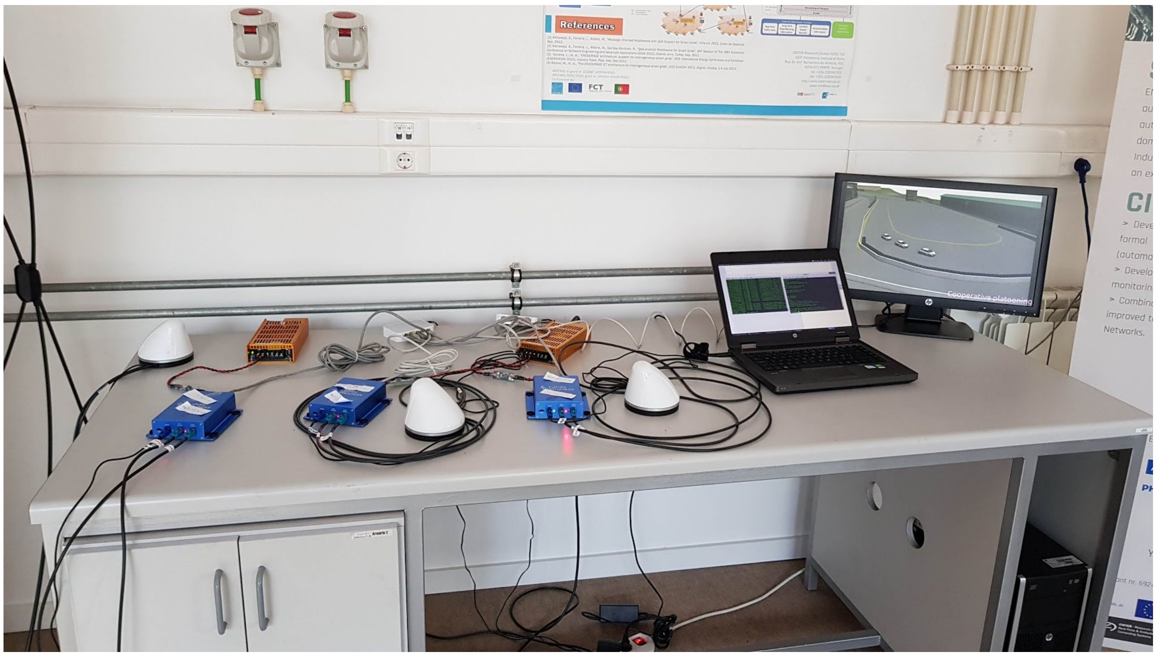
```
0;8.000000;8.442203]
[ConnectionsHandler] (threadID: 1965028432) - ClientHandler
ved. buffer = PUB Low 3235 DataToETSI /car2/;3232251074;
.000000;8.000000;8.442203]
UnpackData: UnpackMessage - message = PUB Low 3235 DataTo
74;1;0.000173;0.087622;0.000000;8.000000;8.442203]
UnpackMessage: found_STX = 0 | found_RS = 4 | found_ETX =
MessageParser: message = PUB Low 3235 DataToETSI /car2/;3232251074;1;0.000173
;0.087622;0.000000;8.000000;8.442203]
DESTINATARIESSELECTOR (list of destinataries):
MESSAGEHANDLER (publication handled):
PRIORITY: 2 ; ID: 13235 ; CONTEXT: DataToETSI ; PAYLOAD: [/car2/;3232251074;1;0.
000173;0.087622;0.000000;8.000000;8.442203]
PUBR OK
[ConnectionsHandler] (threadID: 1965028432) - SendPacket: message size sent. mes
sageSize = 10 | sizeToSend = 167772160
[ConnectionsHandler] (threadID: 1965028432) - SendPacket: message sent. message
= PUBR OK
[ConnectionsHandler] (threadID: 1965028432) - ReceivePacket: reading packet...
[ConnectionsHandler] (threadID: 1975514192) - ReceivePacket: message size receiv
ed. received = 1610612736
[ConnectionsHandler] (threadID: 1975514192) - ReceivePacket: message length rece
ived. messageLength = 96
```

```
[ConnectionsHandler] (threadID: 1971123280) - ReceivePacket
ived. messageLength = 10
[ConnectionsHandler] (threadID: 1971123280) - ReceivePacket
uffer = PUBR OK
UnpackData: UnpackMessage - message = PUBR OK
UnpackMessage: found_STX = 0 | found_RS = 5 | found_ETX = 8
Publisher (threadID: 1971123280) - buffer = PUBR OK
Published ETSIInfoVState. payload = [3232250970;8;426309717
[ConnectionsHandler] (threadID: 1971123280) - SendPacket: message size sent. mes
sageSize = 75 | sizeToSend = 1258291200
[ConnectionsHandler] (threadID: 1971123280) - SendPacket: message sent. message
= PUB Normal 1717 ETSIInfoVState 3232250970;8;426309717;-832598049;28800]
[ConnectionsHandler] (threadID: 1971123280) - ReceivePacket: reading packet...
[ConnectionsHandler] (threadID: 1971123280) - ReceivePacket: message size receiv
ed. received = 167772160
[ConnectionsHandler] (threadID: 1971123280) - ReceivePacket: message length rece
ived. messageLength = 10
[ConnectionsHandler] (threadID: 1971123280) - ReceivePacket: message received. b
uffer = PUBR OK
UnpackData: UnpackMessage - message = PUBR OK
UnpackMessage: found_STX = 0 | found_RS = 5 | found_ETX = 8
Publisher (threadID: 1971123280) - buffer = PUBR OK
Published ETSIInfoVState. payload = [3232250970;8;426309717;-832598049;28800]
```

```
[ConnectionsHandler] (threadID: 1963979856) - ReceivePacket
ived. messageLength = 75
[ConnectionsHandler] (threadID: 1963979856) - ReceivePacket
uffer = PUB Normal 1717 ETSIInfoVState 3232250970;8;42630
]
[ConnectionsHandler] (threadID: 1963979856) - ClientHandler
ved. buffer = PUB Normal 1717 ETSIInfoVState 3232250970;
;28800]
UnpackData: UnpackMessage - message = PUB Normal 1717 ETSIInfoVState 323225097
0;8;426309717;-832598049;28800]
UnpackMessage: found_STX = 0 | found_RS = 4 | found_ETX = 73
MessageParser: message = PUB Normal 1717 ETSIInfoVState 3232250970;8;426309717
;-832598049;28800]
DESTINATARIESSELECTOR (list of destinataries):
MESSAGEHANDLER (publication handled):
PRIORITY: 3 ; ID: 1717 ; CONTEXT: ETSIInfoVState ; PAYLOAD: [3232250970;8;426309
717;-832598049;28800]
PUBR OK
[ConnectionsHandler] (threadID: 1963979856) - SendPacket: message size sent. mes
sageSize = 10 | sizeToSend = 167772160
[ConnectionsHandler] (threadID: 1963979856) - SendPacket: message sent. message
= PUBR OK
[ConnectionsHandler] (threadID: 1963979856) - ReceivePacket: reading packet...
```

```
}

void RobotMiddleware::finish()
{
    Middleware::finish();
    //findHost()->unsubscribe(MobilityBase::stateChangedSignal, this);
}

void RobotMiddleware::initializeIdentity(Identity& id)
```

Figure 48: *HiL simulation setup*



with its OBU, or data coming from other OBUs that broadcast messages, in the same way CAMs are sent, but in a fixed transmission frequency.

6.4 Conclusions

Preliminary results showed to be very close in terms of platoon control to what this model achieved with communications being carried out via internal ROS topics.

As presented in this section , the implementation of both the CLW and RTM software modules within this tool was done with success and some evaluation could be done on top of the results gathered via 30 simulation runs on different scenarios, assuring the good performance of both the simulation tool, as well as, the safety assurance of these modules for these platoon-specific deployments.

7
Experimental results

7.1 COPADRIVe results

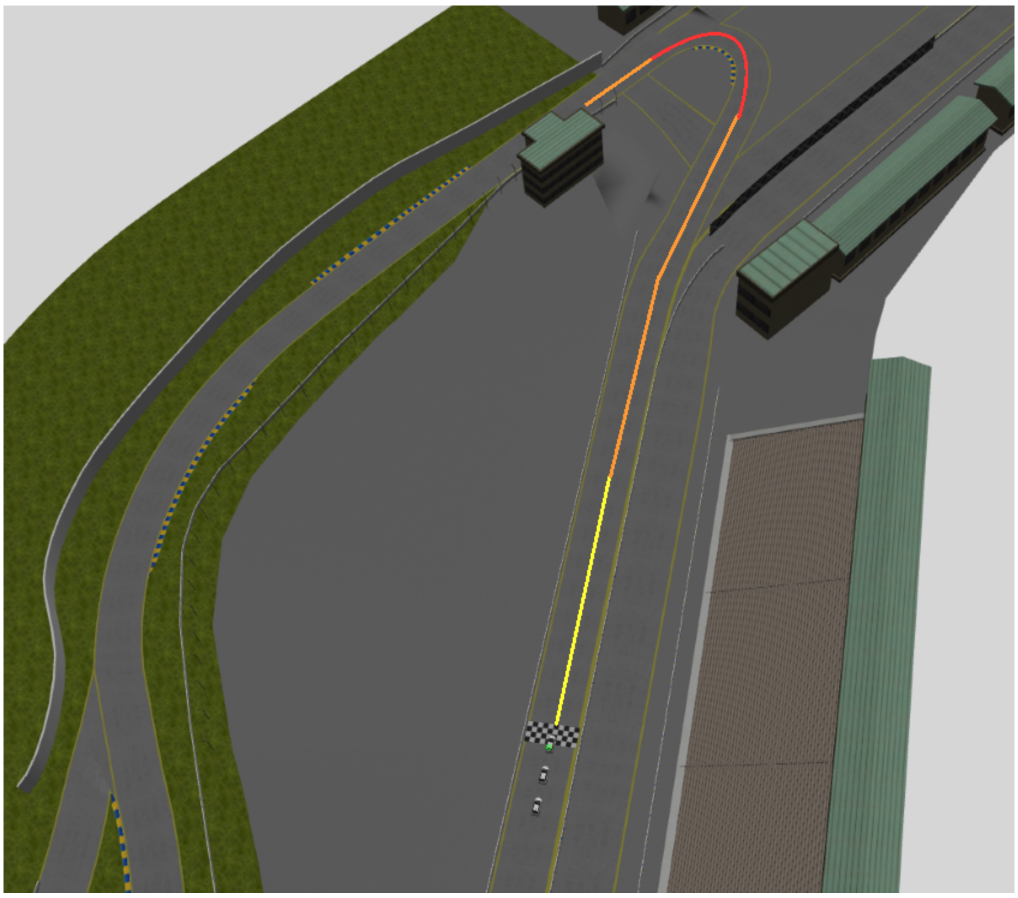
In this section, experimental results of different simulation scenarios ran using COPADRIVe will be displayed and analyzed:

We consider the simulation model defined at section 5.2, with a safe distance setpoint set to 8 meters (GPS values, not physical distance between car's chassis). The simulation results were extracted from 45-seconds long runs, in four different scenarios where the platoon safety was assessed, on different CAService profiles:

- scenario A - fixed Cooperative Awareness Messages (CAM) exchanging frequencies were set.
- scenario B - using the standard Basic Service Profile (BSP) from ETSI following [12].
- scenario C - using the BSP with platooning-defined specifications [5] C.2.11 .
- scenario D - customized settings were defined and evaluated for a profile that tweaks the BSP one.

The simulation environment and simulated platooning trajectory in the 45-second run is represented in Figure 49. The yellow line represents the initial acceleration path, where the follower car is still accelerating to reach the setpoint distance (8

Figure 49: *Platoon Path*



meters) between itself and its leader. In orange, it's represented the path in which the platooning is stable, and in red, a hard turn in which platooning behaviour is greatly dependant on the number of CAMs exchanged. The presented experimental charts also contain this color reference to help relating them with the relevant portion on the track.

7.1.1 Scenario A - Fixed CAM frequencies

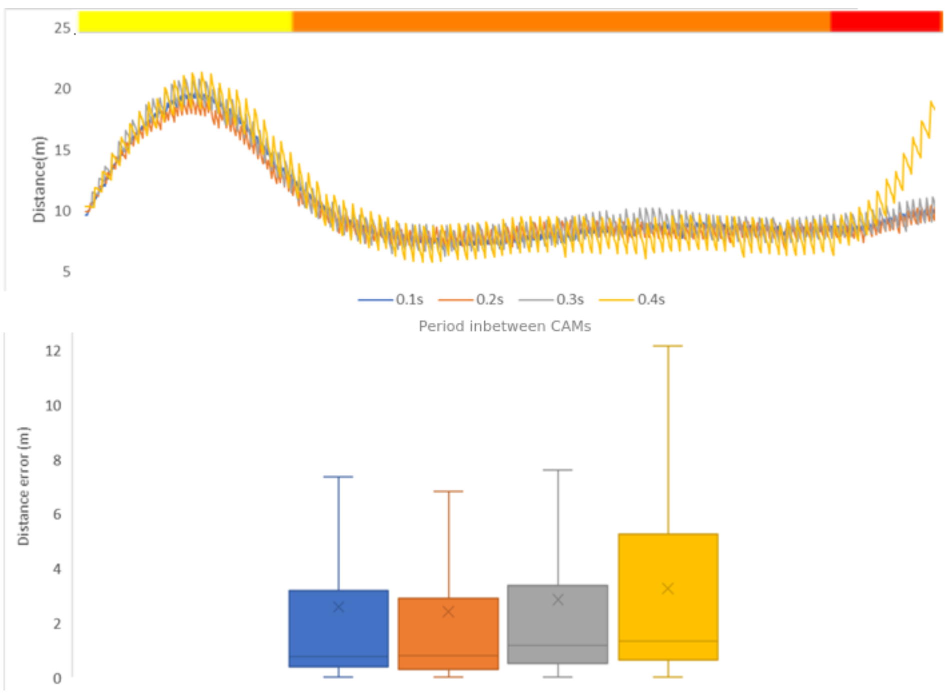


Figure 50: *Vehicle inter-distances - Scenario A*

Four CAM sending frequencies were evaluated (i.e. 10, 5, 3.3 and 2.5 Hz), guaranteeing that at the highest CAM frequency, CAM messages will always be provided with new data, since the "/carX/carINFO" topic is published in a 20Hz frequency. Figure 50 shows the vehicle inter distance in each test and Figure 51 presents the steering angles. The impact of different CAM exchanging frequencies on the behavior of the second car was analyzed, regarding the forward distance and steering angles to analyze how different CAM exchanging frequencies affected the Platoon control. The Platoon starts from stopped position, and the follower only engages platooning after the leader starts moving forward, thus the follower needs to accelerate to catch up to its leader. It is also clearly noticeable that, for a CAM inter arrival time of 0.4 s, while approaching the left hard turn (in red), the follower lost track of the leader vehicle, making a full-stop. At higher CAM sending frequencies, it is possible to ob-

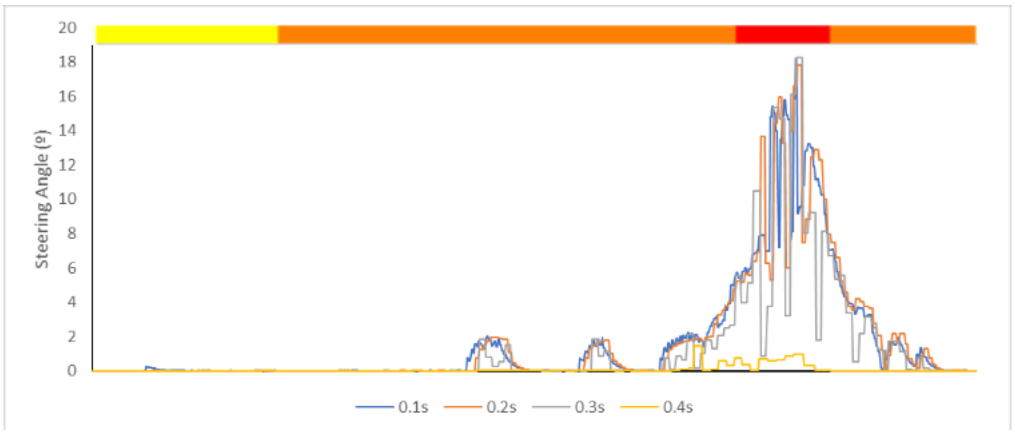


Figure 51: *Steering Angles - Scenario A*

serve that the Platoon PID controller shows better stability, and the inter-distance stability improves. These issues are also particularly visible regarding the steering behavior. For the first three CAM inter arrival times, the steering angles follow the leader's with a slight delay, which increases with frequency. For an inter arrival time of 0.4 s, the steering angles of the follower are no longer inline with the leader's (Figure 51). CAM sending frequency is too low to keep the follower updated with leader's steering corrections, resulting in minimal or nearly non-existent steering inputs. Upon entering the left U turn, the follower's controller struggles to keep up with the steering of the leader, while it completely fails to do so for inter arrival times of 0.4 s. Among the several runs, at different frequencies, it is noticeable a consistent behaviour, in such way that the higher the CAM sending frequency, the stable the PID steering control. However, fixing a CAM frequency represents a sub-optimal approach for Platooning scenarios, considering that excessive CAM traffic will often be generated, which can negatively impact the throughput of the network. With this in mind, ITS-G5 proposed BSP to dynamically trigger CAMs. In the following scenarios its performance in the same context will be evaluated.

7.1.2 Scenario B - standard Basic Service Profile (BSP)

For this Scenario the BSP, as standardized in ITS-G5 [12] and already referenced at 3.2, will be analyzed.

CAM reception intervals are presented in Figure 52. CAM sending frequencies approach 2.0 Hz, mostly due to the second triggering condition, since the Platoon speed during the orange straight part of the track is constant around 8 m/s. However, there are some high frequency triggers in the early iterations, resulting from the quick acceleration at the initial portion of the track, while trying to close the distance gap

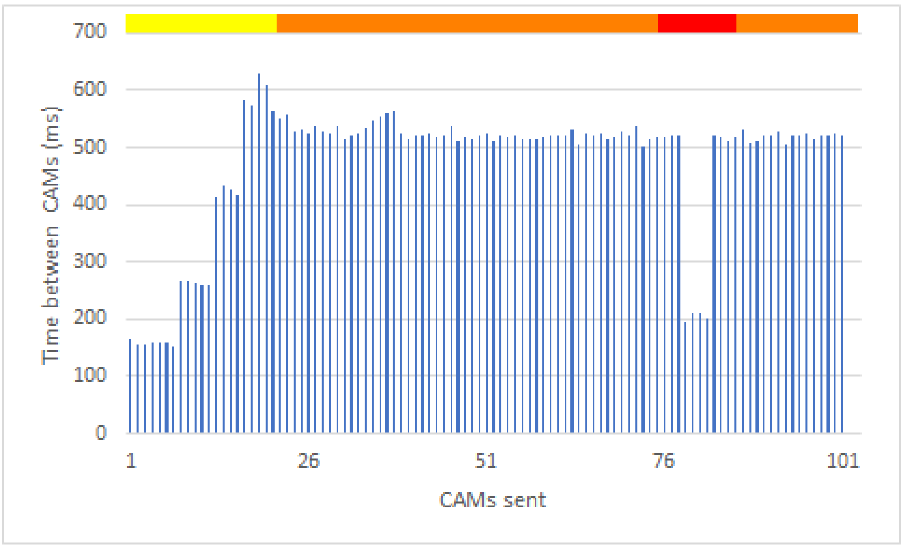


Figure 52: *Period CAM BSP*

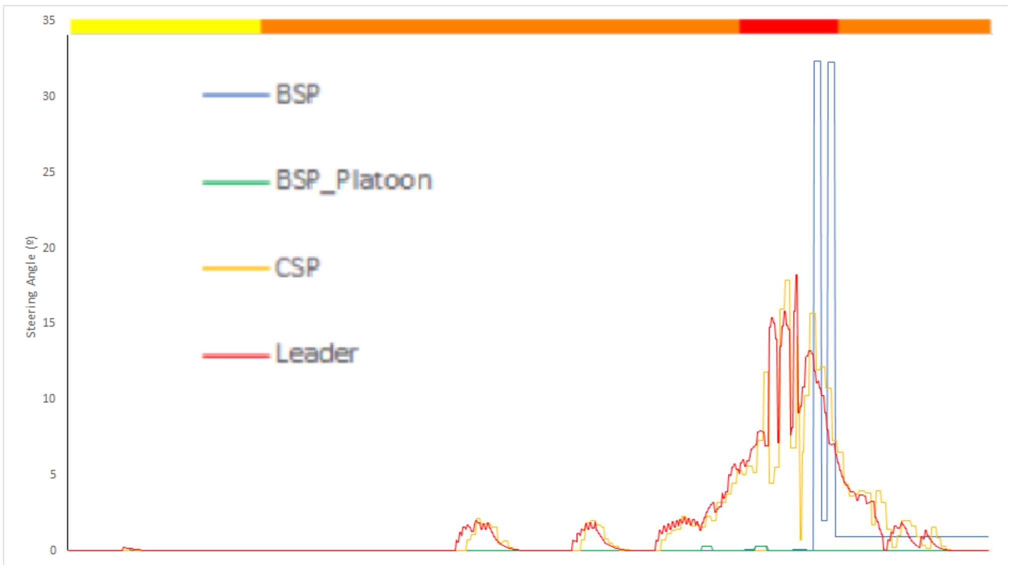


Figure 53: *Steering Analysis for different scenarios*

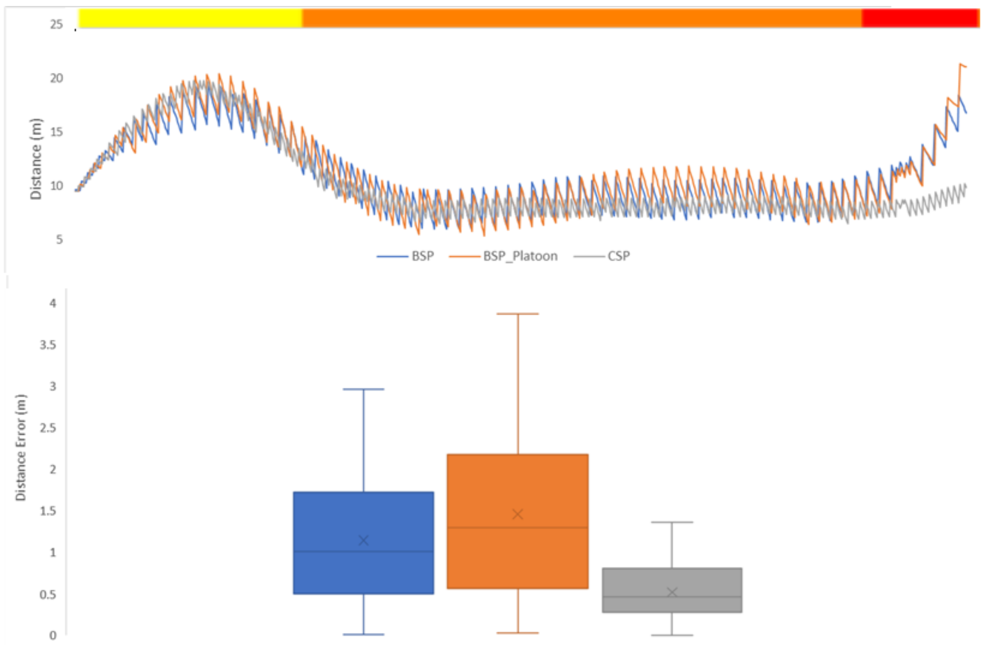


Figure 54: *Longitudinal distances analysis in different scenarios*

to the leader. It is also possible to observe that BSP triggers higher frequencies in response to the hard left turn (red portion of the track), that quickly shifts the leader's heading. Still, as observed in Figures 53 and 54 this increase in frequency was insufficient to maintain a stable Platooning control using this control model, and fails to follow leader's steering control. Therefore, we conclude that BSP is not well-tuned for more demanding Platoon scenarios, in which the control models exclusively rely upon cooperative support. While still trying to minimize network usage, and maintaining stable platooning control, an analysis on this will be done for the next scenarios' implemented profiles.

7.1.3 Scenario C - Basic service profile for platoon scenarios

In this scenario, we analyze an extension to the ITS-G5's BSP specified in [5] C.2.11, which recommends improved BSP settings for platooning scenarios. One of its most significant changes, was to limit the minimum frequency between CAM transmission to 2 Hz, double of the one defined for the original BSP. Test results are quite similar to the usage of the original BSP settings, as triggering conditions remain the same. As shown in Figure 55 and similarly to scenario B, CAM inter arrival times remain around 2Hz(0.5s period), in this case as a result of the minimum frequency limit set. Concerning the Platoon behaviour, Figures 53 and 54 present a similar behaviour to scenario B, which results in a failure to execute the U turn. This is a consequence

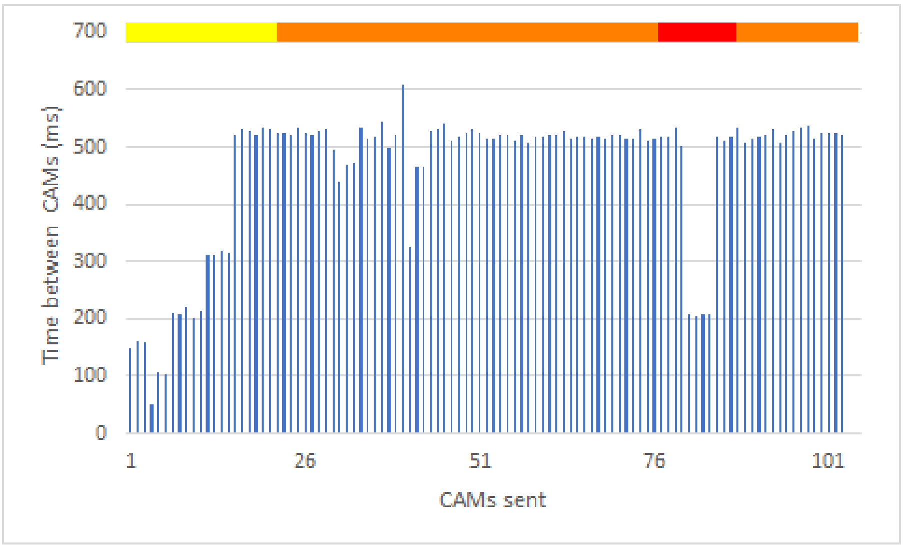


Figure 55: *Period CAM BSP for Platoon*

of platoon instability. Concerning distance error in regards to the set point, for instance, both scenarios B and C, present similar and significant errors, resulting from low CAM update frequency.

7.1.4 Scenario D - Custom Service Profile

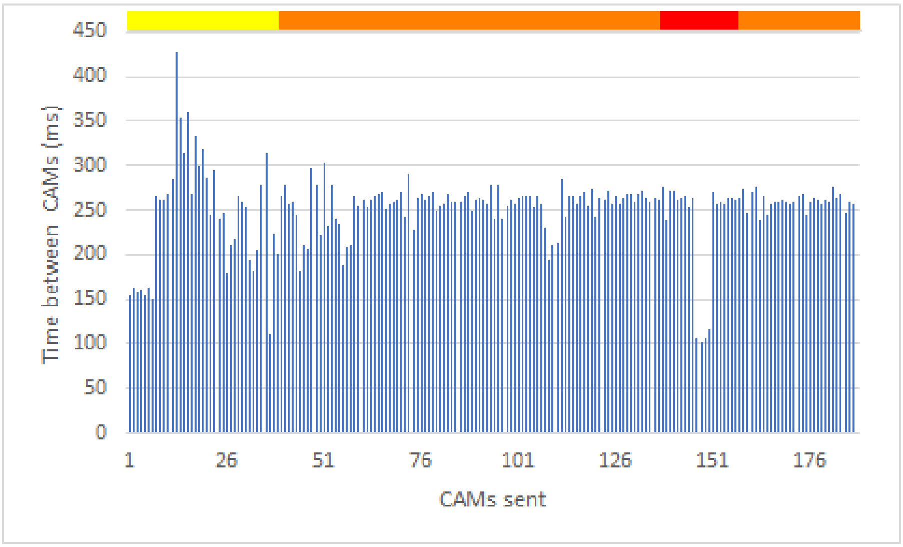


Figure 56: *Period CAM Custom SP*

For this scenario a Custom Service Profile was setup aiming at balancing the network load originated by CAM exchanging, while guaranteeing stability. With this in

Table 7.1: *Comparison between Scenarios - Number of messages and Safety Guarantee*

Scenario	**Fixed Frequencies (Hz)**				**BSP**	**BSP Plat.**	**CSP**
	10	5	3.3	2.5			
Number of messages	441	227	151	113	101	101	181
Safety assurance	OK	OK	OK	NOK	NOK	NOK	OK

mind, the approach was to adapt the second CAM triggering condition mentioned at scenario B, by changing it to 2 meters instead of 4 meters. This change impacted the CoVP behaviour considerably, both in the number of CAMs sent and its frequency, as it's possible to check at Figure 56. As shown in Figures 52 and 55, the CSP conditions caused CAM triggering to happen much more frequently than in previous cases, resulting on a more stable Platoon control when compared to scenarios B and C (Figures 53 and 54). This also translates into a significant decrease of distance errors to the leader, leading to a smoother control. In facto, only this change enabled the Platoon to successfully complete the hard left turn (Figure 49).

Table 7.1 presents the number of CAM messages sent during simulation for each scenario. As shown, a fixed frequency between 3.3 Hz and 2.5 Hz should be at the threshold borderline balance to maintain CoVP control. However, fixing this frequency is not the most reasonable approach since it can cause unnecessary CAM message transmissions. With this in mind, a Service profile approach as defined in ITS-G5 should be the optimal way to handle this, however, as we are able to confirm with scenarios B,C and D this kind of profiling should be adapted to the use-case and particularly to the control model. For this particular control model under test, CAM information availability is crucial to maintain a stable behaviour. This kind of profiling can be easily carried out using this tool, by fully-specifying the simulation environment and Platoon control model over ROS/Gazebo, while using OMNeT++'s capabilities to analyze the network performance and to provide new extensions to the ITS-G5 communications stack, carrying out an integrated in depth analysis of different C-ITS behaviours.

7.2 Hardware-in-the-loop results

In this section, some experimental results, gathered from a variety of simulations ran, will be presented.

The simulations ran in these results context were based on the HiL simulation integration already descripted, with a platoon simulation model running on Gazebo with the vehicle's V2V communications being done through real OBUs that are connected through the ROS_MK5_bridge into their correspondent vehicle node.

For the shown results, 30 simulations were run. 15 with only the CLW module running on the OBUs as the alert system providing the vehicle's control a reliant safety layer. For the remaining 15 runs, the RT Monitor was added as an addition for the alert system redundancy by working in parallel with the CLW module, however, obviously running in different platforms.

These 15 (+15) simulations were split in between 3 different scenarios, all of them simulating a different type of failure on the follower vehicle control:

- Loss of Acceleration (LOA) - In this scenario, the leader, on a straight path, accelerates and its follower is not able to respond to this because of a failure in its acceleration system. The CLW/RT monitor detect the failure and alert its control application.

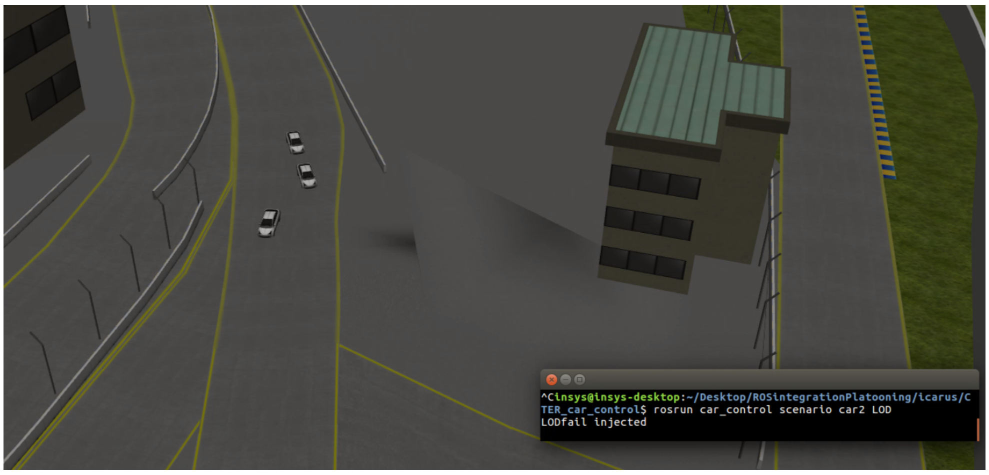
- Loss of Brakes (LOB) - In this scenario, the leader, on a straight path, brakes and its follower is not able to respond to this because of a failure in its braking system. The CLW/RT monitor detect the failure and alert its control application.

- Loss of Direction (LOD) - In this scenario, the leader takes a turn and its follower is not able to respond to this because of a failure in its steering system. The CLW/RT monitor detect the failure and alert its control application.

All these failure scenarios were implemented as a trigger ROS application for their occurrence on the 2nd platoon member (3 vehicles platoon). A LOD scenario run can be seen at figure 57 after the failure trigger happening. In all simulation scenarios, the vehicle's control response to the alert reception from the CLW/RTM was always to emergency break. In the case of the Loss of brakes, the failure detected was a missing braking input (e.g the pedal not working), not a real mechanical brake problem.

At table 7.2 some results gathered from, the 30 simulation runs done to test out the CLW/RTM modules within the previously stated scenarios, are shown. These tests have been ran under the SafeCOP project in order to evaluate and guarantee some safety assurance metrics. These results, apart from guaranteeing the well-functioning of these modules also assure the correct functioning of this tool as a testing tool for C-ITS scenarios where the Cohda MK5 OBU is used.

Figure 57: *Loss of Direction scenario*



```
^Cinsys@insys-desktop:~/Desktop/ROSintegrationPlatooning/icarus/C
TER_car_control$ rosrun car_control scenario car2 LOD
LODfail injected
```

Table 7.2: *HiL simulations - Analysis of CLW/RTM alert systems*

Metrics	Results without RT Monitor	Results w/ RT Monitor	Units
Percentage of test runs where a crash between two or more platooning nodes occurred	13,3333	0	%
Average number of communication failures detected per test run	0,2353	0,1176	Avg
Average number of corrupted messages detected per test run	47	47	Avg
Average number of delivery delays detected per test run	0,2674	0,1429	Avg
Percentage of test time where communications were performed under an acceptable level of latency	98,8889	91,9012	%
Percentage of successful full stops after a catastrophic failure	86,6667	100	%
Downtime percentage of any SW component	1,2069	1,9023	%
Average response time for warnings	30,004	4,78	ms
Average response time for non warnings	35,302	14,79	ms

7.3 Conclusions

Regarding the results gathered in this section, both in the case of COPADRIVe, as well as, in the HiL simulation case, these results prove the usefulness of these tools in enabling testing and analyzing C-ITS scenarios in regards to the impact of communications in their behaviour, as well as, in supporting the test of several safety mechanisms within them.

COPADRIVe results, clearly depicted the impact of the CaService settings on the behaviour of the platooning control model. Adding to this, some baseline values for CAM exchanging frequency were gathered for us to be able to properly design future cooperative platooning implementations. Regarding ETSI ITS-G5, these results can be, as well, helpful for properly adjust CaService settings for platooning scenarios, or other safety critical scenarios.

About the HiL simulation results, these provided a good insight on how to properly test out monitoring software like CLW and Runtime Monitoring in specific scenarios, as platooning, where they proved their usefulness in guaranteeing safety to the vehicles involved.

8 Conclusions and Future Work

C-ITSs still remain, in their most complex automated scenarios, a distant reality. Mainly for political and ethical uncertainties, the intents of advancing for deployments of these kind of systems remain delayed. Obviously, some of these pending decisions, remain not clarified and waiting for technological advances on a wide variety of areas.

In the case of vehicular cooperative scenarios (e.g platooning), research and development regarding communications, decentralized decision making and control of vehicles, will probably become hot areas of interest in the automotive industry for the next few years. Since these kind of testing during development are very cost-heavy for deployment, implementation using simulation tools is, and will keep on being the easiest and cheapest approach on this.

In this thesis, an overview, regarding the most impactful ITS scenarios that can rely on V2X communications, was done with a focus on the platooning scenario that was used as a baseline for the 2 co-simulation tools developed. This overview and analysis relying on the ETSI ITS-G5 standard, which is, today, the most widely accepted V2X standard for communications.

About the developed tools, COPADRIVe presents itself as a unique approach as a co-simulation tool joining together a robotic simulator and a network one, with a ITS-G5 stack implemented, which is a very useful contribution for the analysis of different vehicular scenarios that might use V2V communications. The HIL tool provides a lower level of abstraction in what regards simulation, since it uses real hardware and real physical communications in order to provide data into the simulated vehicles. This tool, was fundamental to analyse and demonstrate the monitoring software that ensured platooning safety.

COPADRIVe is the most flexible of both projects in terms of future work and developments. There are many paths possible to follow on this. Focusing on V2V communications, the development and integration of different scenarios, like cross-road management is a strong option, since Gazebo is a simulation that guarantees easily development of different models. Exploring the communications stack should be a focus as well, extending CAM message usage to different scenarios or even implementing DENM messages to provide inputs necessary in some C-ITS scenarios, can be viable options. Of course, continuous work on network analysis and tool reliability are the obvious developments for close future.

Regarding the HIL simulation tool, the extension of the tool to other ITS scenarios can be quite interesting. Also, studying the impact of noise and other radio physical layer issues can be carried out in the future with realistic communication platforms.

Bibliography

[1] E. Larsson, G. Sennton, and J. Larson, “The vehicle platooning problem: Computational complexity and heuristics,” Transportation Research Part C: Emerging Technologies **60,** 258–277 (2015).

[2] P. Fernandes and U. Nunes, “Platooning With IVC-Enabled Autonomous Vehicles: Strategies to Mitigate Communication Delays, Improve Safety and Traffic Flow,” IEEE Transactions on Intelligent Transportation Systems **13,** 91–106 (2012).

[3] B. Vieira, R. Severino, A. Koubaa, and E. Tovar, “Towards a Realistic Simulation Framework for Vehicular Platooning Applications,” CoRR abs/1904.02994 (2019).

[4] B. Vieira, R. Severino, E. Vasconcelos Filho, A. Koubaa, and E. Tovar, “COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications,” In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE) (IEEE ICCVE 2019)*, (Graz, Austria, 2019).

[5] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions*, 2010.

[6] A. Talebpour and H. S. Mahmassani, “Influence of connected and autonomous vehicles on traffic flow stability and throughput,” Transportation Research Part C: Emerging Technologies **71,** 143–163 (2016).

[7] O. Karoui, M. Khalgui, A. Koubaa, E. Guerfala, Z. Li, and E. Tovar, “Dual mode for vehicular platoon safety: Simulation and formal verification,” Information Sciences **402,** 216 – 232 (2017).

[8] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Communications Architecture*, 2010.

[9] A. Festag, “Standards for vehicular communication—from IEEE 802.11p to 5G,” e & i Elektrotechnik und Informationstechnik **132,** 409–416 (2015).

[10] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, 2019.

[11] H. Zimmermann, “OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection,” IEEE Transactions on Communications **28,** 425–432 (1980).

[12] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, 2014.

[13] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, 2014.

[14] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*, 2010.

[15] B. Hinden and D. S. E. Deering, "Internet Protocol, Version 6 (IPv6) Specification,", RFC 2460, 1998.

[16] D. B. Johnson, J. Arkko, and C. E. Perkins, "Mobility Support in IPv6,", RFC 6275, 2011.

[17] P. Thubert, A. Petrescu, R. Wakikawa, and V. Devarapalli, "Network Mobility (NEMO) Basic Support Protocol,", RFC 3963, 2005.

[18] "Internet Protocol,", RFC 791, 1981.

[19] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol*, 2017.

[20] "User Datagram Protocol,", RFC 768, 1980.

[21] "Transmission Control Protocol,", RFC 793, 1981.

[22] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 3: Network Architecture*, 2014.

[23] S. Kuhlmorgen, I. Llatser, A. Festag, and G. Fettweis, "Performance Evaluation of ETSI GeoNetworking for Vehicular Ad Hoc Networks," In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pp. 1–6 (2015).

[24] "IEEE Standard for Information technologyâTelecommunications and information exchange between systems Local and metropolitan area networksâSpecific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012) pp. 1–3534 (2016).

[25] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*, 2010.

[26] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part*, 2010.

[27] N. Lyamin, A. Vinel, D. Smely, and B. Bellalta, "ETSI DCC: Decentralized Congestion Control in C-ITS," **56,** 112–118 .

[28] E. T. S. I. (ETSI), *Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*, 2012.

[29] Cohda, "https://cohdawireless.com/solutions/hardware/mk5-obu/,".

[30] Cohda, "https://cohdawireless.com/solutions/v2x-stack/,".

[31] R. Riebl, H.-J. Gunther, C. Facchi, and L. Wolf, "Artery: Extending Veins for VANET applications," In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*,

[32] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08 pp. 60:1–60:10 (ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2008).

[33] Artery, "https://github.com/riebl/artery,", artery project official git repository.

[34] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," IEEE Transactions on Mobile Computing **10,** 3–15 (2011).

[35] Webots, "http://www.cyberbotics.com,", commercial Mobile Robot Simulation Software.

[36] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149–2154 (Sendai, Japan, 2004).

[37] B. Leiding, P. Memarmoshrefi, and D. Hogrefe, "Self-managed and blockchain-based vehicular ad-hoc networks," In , pp. 137–140 (2016).

[38] A. M. S. Abdelgader and W. Lenan, "The Physical Layer of the IEEE 802.11p WAVE Communication Standard: The Specifications and Challenges," p. 8 .

[39] D. Jiang and L. Delgrossi, "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments," In *VTC Spring 2008 - IEEE Vehicular Technology Conference*,

[40] D. Eckhoff, N. Sofra, and R. German, "A performance study of cooperative awareness in ETSI ITS G5 and IEEE WAVE," In *2013 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*,

[41] B. Lehmann, H.-J. Gunther, and L. Wolf, "A Generic Approach towards Maneuver Coordination for Automated Vehicles," In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*,

[42] A. Maria, M. Biagi, and R. Cusani, “Smart Vehicles, Technologies and Main Applications in Vehicular Ad hoc Networks,” in *Vehicular Technologies - Deployment and Applications*,

[43] I. Mavromatis, A. Tassi, R. J. Piechocki, and A. Nix, “A City-Scale ITS-G5 Network for Next-Generation Intelligent Transportation Systems: Design Insights and Challenges,” in *Ad-hoc, Mobile, and Wireless Networks*,

[44] I. Rashdan, F. d. P. MÃ$\frac{1}{4}$ller, and S. Sand, “ITS-G5 Challenges and 5G Solutions for Vehicular Platooning,” p. 7 .

[45] N. P. S. Andersen, “C-ITS ACTIVITIES IN EUROPE,” p. 52 .

[46] W. Broeders, “Overview of standards for first deployment of C-ITS,” p. 33 .

[47] A. K. Saluja, S. A. Dargad, and K. Mistry, “A Detailed Analogy of Network Simulators â Ns1, Ns2, Ns3 and Ns4,” **3,** 5 .

[48] “Comparison of Different Network Simulators,” p. 6 .

[49] C. Obermaier and C. Facchi, “Observations on OMNeT++ Real-Time Behaviour,” .

[50] A. Wegener, M. PiÃ^3*rkowski*, *M.Raya*, *H.Hellbr*$\frac{1}{4}$*ck*, *S.Fischer*, *andJ.* − *P.Hubaux*, “*TraCI* : *aninterfaceforcouplingroadtrafficandnetworksimulators*,″ *InProceedingso*

[51] K. Garlichs, M. Wegner, and L. C. Wolf, “Realizing Collective Perception in the Artery Simulation Framework,” In *2018 IEEE Vehicular Networking Conference (VNC)*,

[52] H. Hartenstein and L. P. Laberteaux, “A tutorial survey on vehicular ad hoc networks,” IEEE Communications Magazine **46,** 164–171 (2008).

A

Appendix A - Towards a Realistic Simulation Framework for Vehicular Platooning Applications

Towards a Realistic Simulation Framework for Vehicular Platooning Applications

Bruno Vieira [1], Ricardo Severino [1], Anis Koubaa [1,2], Eduardo Tovar [1]
[1] CISTER Research Centre, ISEP, Polytechnic Institute of Porto
[2] Prince Sultan University, Saudi Arabia
{bffbv, rarss, emt}@isep.ipp.pt, akoubaa@psu.edu.sa

Abstract— **Cooperative vehicle platooning applications increasingly demand realistic simulation tools to ease their validation, and to bridge the gap between development and real-word deployment. However, their complexity and cost, often hinders its validation in the real-world. In this paper we propose a realistic simulation framework for vehicular platoons that integrates Gazebo with OMNeT++ over Robot Operating System (ROS) to support the simulation of realistic scenarios of autonomous vehicular platoons and their cooperative control.**

I. Introduction

Vehicular Platooning (VP) is an emerging application of the new generation of safety-critical Cooperating Cyber Physical Systems. Although VP can increase fuel efficiency and road capacity, by having vehicle groups traveling close together, VP presents several safety challenges, considering it heavily relies on wireless communications, and upon a set of sensors that can be affected by noise. The ETSI ITS-G5[10] is considered as the enabler ready-to-go communications technology for such applications, and although there has been extensive analysis of its performance [8], [9], [11], the understanding of its impact upon the safety of these Systems of Systems (SoS) is rather immature. Hence, extensive testing and validation must be carried out to understand the safety limits of such SoS by encompassing communications. However, the expensive equipment and safety risks involved in testing, demands for comprehensive simulation tools that can as accurate as possible mimic the real-life scenarios, from the autonomous driving perspective as well as from the communications perspective. The Robotic Operating System (ROS) framework is already widely used to design robotics applications, and aims at easing the development process by providing multiple libraries, tools and algorithms, and a publish/subscribe transport mechanism. On the other hand, several network simulators are available and capable of carrying out network simulation of vehicular networks. Nonetheless, these remain mostly separated from the autonomous driving reality offering none or very limited capabilities in terms of evaluating cooperative autonomous driving systems. In this work, we carried out the integration of a well-known ROS-based robotics simulator (Gazebo) with a network simulator (OMNeT++), enabling a powerful framework to test and validate cooperative autonomous driving applications. Currently, most relevant simulation frameworks focus on enabling an integration between traffic and network simulators, supporting the evaluation of Inteligent Traffic Systems (ITS), e.g., VSimRTI [1], Artery [3]. Some support vehicular platooning applications, such as Webots[4], VISSIM[6], CORSIM[7]. However, all these come short in comparison with the advanced simulation and capabilities of a robotics simulation framework such as Gazebo, which is developed from scratch to enable a realistic simulation environment for autonomous systems via accurate physical modelling of sensors, actuators and vehicles, while harnessing the power of the ROS development environment. Plexe [2], extension of Veins, aims at enabling platooning simulation, by integrating OMNeT++, SUMO [5] together with a few control and engine models. However, similarly to previous examples, it lacks the power of ROS and only enables the test of longitudinal platooning i.e., no lateral control, and lacks support of a ITS-G5 communication stack, the standard for C-ITS applications in Europe. Hence, this work, is the first to integrate ROS and a network simulator supporting a full stack of the ETSI ITS-G5.

II. Framework Architecture

Our simulation framework builds over the Veins simulator and the Vanetza communications stack implementation, borrowing and extending much of the middle-ware components from the Artery framework. It relies on ROS publish/subscribe mechanisms to integrate OMNeT++ with Gazebo, represented by blue arrows at Figure 1. Each OMNeT++ node represents a cars network interface and contain a Vehicle Data Provider (VDP) and a Robot Middleware (RM). VDP is the bridge that supplies RM data from the Gazebo simulator. RM uses this data to fill ITS-G5 CAMs data fields (i.e Heading, Speed values) through the CaService that proceeds to encode this data fields in order to comply with ITS-G5 ASN-1 definitions. RM also provides GPS positions to position the nodes in the INET mobility module. Regarding synchronization, OMNeT++ is an event-driven simulator and Gazebo is a time-driven simulator, thus synchronizing both simulators represented a challenge. A synchronization module was implemented in OMNeT++, relying on ROS /Clock topic as clock reference, which schedules a custom made OMNeT++ message for this purpose (syncMsg) to an exact ROS time, forcing the OMNeT++ simulator engine to generate an event upon reaching that timestamp.

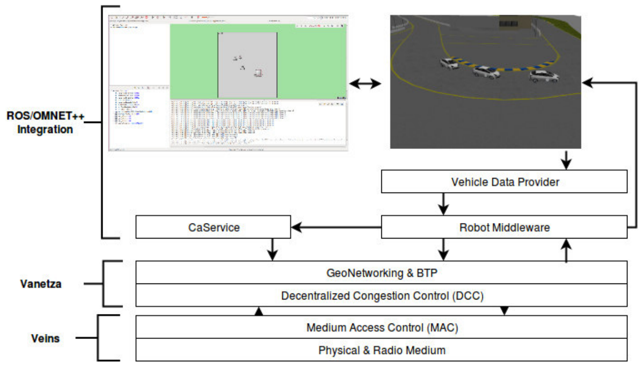


Fig. 1. Framework Architecture

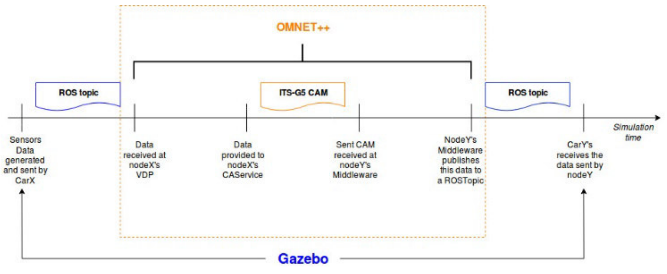

OMNET++
ROS topic
ITS-G5 CAM
ROS topic
Simulation time
Sensors Data generated and sent by CarX
Data received at nodeX's VDP
Data provided to nodeX's CAService
Sent CAM received at nodeY's Middleware
NodeY's Middleware publishes this data to a ROSTopic
CarY's receives the data sent by nodeY
Gazebo

Fig. 2. Data workflow

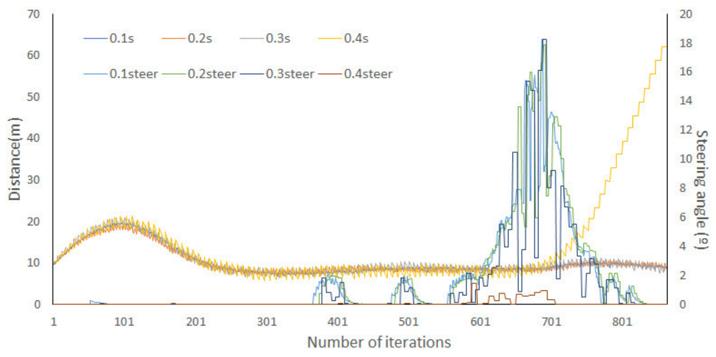


Fig. 3. Vehicle inter-distances and steering angles

III. Experimental Results

The simulation is composed of three vehicles modeled from a Toyota Prius running a PID-based platooning control model [9] that solely relies on Cooperative Awareness Messages (CAM) to maintain the platooning service, with a safe distance set to 8 meters. Different CAM sending frequencies were evaluated (10 Hz - 2.5 Hz). At the lower frequency value, the second car fails to keep up with the leader vehicle. Fig. 2 presents a quick overview on how data flows from carXs sensors into carYs control application, following a CAM transmission between different nodes in OMNeT++. NodeX and nodeY represent the vehicles network interfaces. We analyzed the impact of different CAM exchanging frequencies (Fig.3) on the platoon-following behavior of the second car, regarding the longitudinal distance and steering angles. This provides us with a good perception on how the different frequencies affect the PID longitudinal control of the car. Earlier iteration values confirm that the platooning starts from parked position, and the follower only starts platooning after the leader starts moving, thus the follower needs to accelerate to catch up to its leader. It is also clearly noticeable, in the 0.4s period around iteration 700, the car lost track of the leader vehicle, making a stop and the leader kept going forward. At higher CAM sending frequencies, we can observe that the PID controller shows better stability, and the inter-distance stability improves. The same is visible for the steering behavior. Again, like in the previous graph, around the 700 iterations mark, we can notice the steering angle of the car staying at zero, as it lost track of its leader and stopped. Regarding the other runs on different frequencies, we can get a similar comparison to the previous one, where we can, notice that the PID steering control becomes more stable the higher the CAM sending frequency.

IV. Conclusions and Future Work

Initial validation confirms a positive feedback between the network simulation parameters and its effect in the platooning control model. Further validation is to be carried out regarding the impact of the introduced simulator delay, impact of different network parameters and the performance of other platooning control models.

V. Acknowledgments

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UID/CEC/04234); also by the FCT and the EU ECSEL JU under the H2020 Framework Programme, within project(s) ECSEL/0002/2015, JU grant nr. 692529-2 (SAFECOP) and ECSEL/0006/2015, JU grant nr. 692455-2 (ENABLE-S3).

References

[1] B. Schuenemann, V2X Simulation Runtime Infrastructure VSimRTI: An Assessment Tool to Design Smart Traffic Management Systems , Computer Networks, Volume 55, pp. 3189-3198, October 2011.
[2] M. Segata,et al., "PLEXE: A Platooning Extension for Veins," Proceedings of 6th IEEE VNC 2014, Dec 2014, pp. 53-60.
[3] R. Riebl, et al. , "Artery: Extending Veins for VANET applications," Int. Conf. Models and Technologies for Intelligent Transportation Systems (MT-ITS), Budapest, 2015.
[4] Michel, O. Webots: Professional Mobile Robot Simulation, Int. Journal of Advanced Robotic Systems, Vol. 1, Num. 1, pages 39-42.
[5] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 2575-2582.
[6] VISSIM: microscopic, behavior-based multi-purpose traffic simulation program. http://www.ptvamerica.com/vissim.html.
[7] L. Owen et al., Traffic Flow Simulation Using CORSIM, Proc.2000 Winter Simulation Conf., 2000, pp. 1143147.
[8] O. Karoui, et al. , Mohamed Khalgui, Anis Kouba, Emna Guerfala, Zhiwu Li, Eduardo Tovar, Dual mode for vehicular platoon safety: Simulation and formal verification, Information Sciences, Volume 402, 2017, Pages 216-232
[9] O. Karoui, et al. , Performance evaluation of vehicular platoons using Webots, IET Intelligent Transport Systems, Volume 11, Issue 8, 201
[10] ETSI, "ETSI EN 302 665 V1.1.1(09-2010):" Intelligent Transport Systems (ITS); Communications Architecture"
[11] A. Koubaa et al. , 'System and method for operating a follower vehicle in a vehicle platoon ', US9927816B2, 2016

B

Appendix B - COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications

COPADRIVe - A Realistic Simulation Framework for Cooperative Autonomous Driving Applications

Bruno Vieira [§] , Ricardo Severino [§] ,Enio Vasconcelos Filho [§] , Anis Koubaa [§] [*], Eduardo Tovar[§]
[§]Cister Research Centre
ISEP, Polytechnic Institute of Porto
Porto, Portugal
[*] Prince Sultan University, Saudi Arabia.
[§](bffbv, enpvf, rarss, emt)@isep.ipp.pt, [*]akoubaa@psu.edu.sa

Abstract—**Safety-critical cooperative vehicle applications such as platooning, require extensive testing, however, the complexity and cost involved in this process, increasingly demands for realistic simulation tools to ease the validation of such technologies, helping to bridge the gap between development and real-word deployment. In this paper we propose a realistic co-simulation framework for cooperative vehicles, that integrates Gazebo, an advanced robotics simulator, with the OMNeT++ network simulator, over the Robot Operating System (ROS) framework, supporting the simulation of advanced cooperative applications such as platooning, in realistic scenarios.**

Index Terms—**C-ITS; Vehicular Platooning; ROS; OMNeT++**

I. Introduction

Modern embedded systems, coupled with the advancements of digital communication technologies, have been enabling a new generation of systems, tightly interacting with the physical environment via sensing and actuating actions: Cyber Physical Systems (CPS) [1]. These systems, characterized by an unprecedented levels of ubiquity, have been increasingly relying upon wireless communication technologies to provide seamless services via flexible cooperation, enabling true Systems-of-Systems (SoS).

Cooperative Vehicular Platooning (CoVP) is one of these emerging applications among the new generation of safety-critical Cooperating CPS. CoVP can potentiate several benefits, such as increasing road capacity and fuel efficiency and even reducing accidents [2], by having vehicle groups traveling close together. However, CoVP presents several safety challenges, considering it heavily relies on wireless communications to exchange safety-critical information, and upon a set of sensors that can be affected by noise. For instance, quite often in CoVP, wireless exchanged messages contribute to maintain the inter-vehicle safety distance, or to relay safety alarms to the following vehicles. Message losses or delays may lead to serious crashes among the vehicles in the platoon with dramatic consequences to them and to the remaining road members [3]. It is thus a truism, that timeliness and reliability of the communications are critic aspects for ensuring the safe operation of the platoon.

The ETSI ITS-G5 [4] is increasingly considered the enabler, ready-to-go communications technology for such applications, and although there has been extensive analysis of its performance [5]–[7], the understanding of its impact upon the safety of these SoS is rather immature. Hence, extensive testing and validation must be carried out to understand the safety limits of such SoS by encompassing communications. However, the expensive equipment and safety risks involved in testing, demands for comprehensive simulation tools that can as accurate as possible mimic the real-life scenarios, from the autonomous driving or control perspective, as well as from the communications perspective. The Robotic Operating System (ROS) framework is already widely used to design robotics applications, and aims at easing the development process by providing multiple libraries, tools and algorithms, and a publish/subscribe transport mechanism. On top of it, several simulation tools are capable to simulate the physics and several of the sensor/actuator and control components of these vehicles. On the other hand, several network simulators are available and capable of carrying out network simulation of vehicular networks. Nonetheless, these tools remain mostly separated from the autonomous driving reality, offering none or very limited capabilities in terms of evaluating cooperative autonomous driving systems.

In this work, we carried out the integration of a well-known ROS-based robotics simulator (Gazebo) with a network simulator (OMNeT++), by extending Artery [8], enabling a powerful framework to test and validate cooperative autonomous driving applications. In the one hand, we leverage upon Gazebo's robotic simulation most prominent features, such as its support for multiple physics engines, and its rich library of components and vehicles in integration with ROS, which enables us to build realistic vehicle control scenarios. On the other hand, OMNet++ supports the underlying network simulation relying on an ITS-G5 communications stack which is, currently, the *de-facto* standard for C-ITS applications in Europe. This integration provides the support for an accurate

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UID/CEC/04234); also by the Operational Competitiveness Programme and Internationalization (COMPETE 2020) under the PT2020 Partnership Agreement, through the European Regional Development Fund (ERDF), and by national funds through the FCT, within project POCI-01-0145-FEDER-032218 (5GSDN).

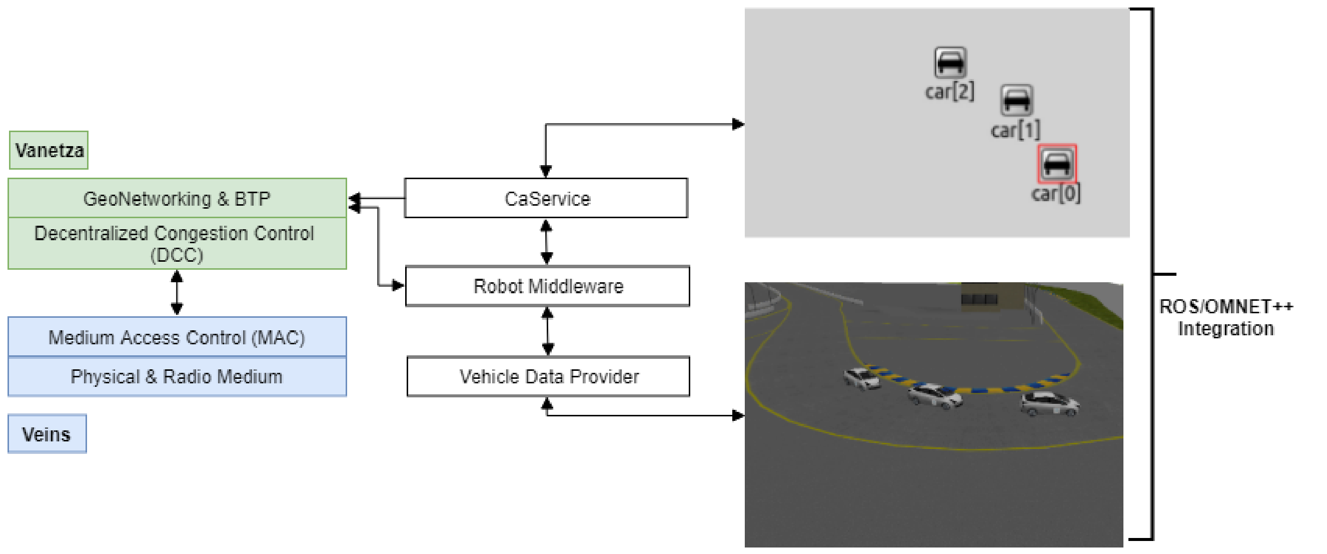


Fig. 1: Framework Architecture

analysis of the communications impact upon the cooperative application, and on the other hand, the tools to carry out a thorough evaluation of the network performance using the OMNet++/INET framework.

We present several contributions in this paper: (1) we implement a co-simulation framework for cooperative autonomous driving applications, relying on ITS-G5 communications and integrating other open-source tools with ROS support; (2) as a prove-of-concept, we implement a platooning control model, solely dependant on V2V communications, to assess the simulation framework, and (3) we analyse the impact of typical CAM settings, provided by the Basic System Profile (BSP) as standardized in ITS-G5 [4], upon the platooning behaviour, and evaluate its adequacy for a fully CoVP model.

In the remaining of this paper, in Section II we overview the related work, in particular the current state-of-the-art regarding connected vehicles simulation. In Section III, we describe the framework architecture, focusing on several implementation details. Finally, in Section IV we present different test scenarios and the experimental results, and conclude the paper in Section V.

II. Related Work

Currently, most relevant simulation frameworks focus on enabling an integration between traffic and network simulators to support the evaluation of Intelligent Traffic Systems (ITS). Some examples include iTETRIS [9] which integrates SUMO and ns-3, but it is pretty much stagnant, and unresponsive; VSimRTI [10] using an ambassador concept to support integration of virtually any simulator. Different traffic simulators and communication simulators have already been integrated, such as, the traffic simulators SUMO and PHABMACS and the network simulators ns-3 and OMNeT++; Artery [8], provides an integration of Veins (integrating SUMO traffic simulator and OMNet++) and the Vanetza ITS-G5 implementation. We identified Artery as the most mature project, which supported the best features to serve as guideline for our integration. In general, although these simulators may suffice to analyze macroscopic/mesoscopic, or even microscopic vehicular models, they are inadequate to support an evaluation of sub-microscopic models, which focus on the physics and particular characteristics of each vehicle. An exception is Plexe [11], an extension of Veins, which aims at enabling platooning simulation by integrating OMNeT++ and SUMO [12] together with a few control and engine models. However, it only enables the test of longitudinal platooning i.e., no lateral control and lacks support of a ITS-G5 communication stack. It is also limited in its capabilities to simulate a rich autonomous driving environment, when compared to robotic simulators, which primary objective is to mimic a realistic deployment environment to validate autonomous control models, encompassing sensors and actuators, rich simulation environments, as well as realistic physics models. There are other robotic simulators available that can support this kind of simulation. Perhaps the most prominent one, due its support of V2V communication via ns-3 is Webots [13]. However, the tool only very recently became open-source, and it does not support ROS out-of-the-box. Also, the ns-3 plugin does not support simulation of a ITS-G5 communications stack. Our proposed framework provides a clear advantage regarding the analysis of cooperative autonomous driving applications, in particular CoVP. It relies on Gazebo to enable a realistic simulation environment for autonomous systems via accurate modelling of sensors, actuators and vehicles, while harnessing the power of the ROS development environment, for developing new and complex algorithms from scratch using its ROS C++/Python framework.

III. Framework Architecture

A. *OMNeT++'s Modules Overview*

Our simulation framework was built over the Veins simulator and the Vanetza communications' stack implementation,

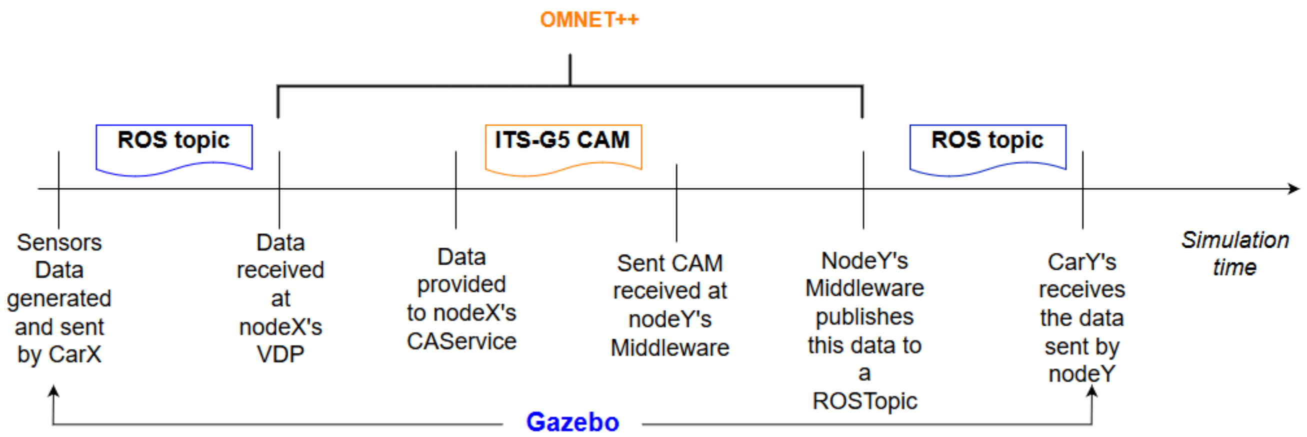

OMNET++
ROS topic
ITS-G5 CAM
ROS topic
Sensors Data generated and sent by CarX
Data received at nodeX's VDP
Data provided to nodeX's CAService
Sent CAM received at nodeY's Middleware
NodeY's Middleware publishes this data to a ROSTopic
CarY's receives the data sent by nodeY
Simulation time
Gazebo

Fig. 2: Data workflow

borrowing and extending much of the middleware components from the Artery framework. It relies on ROS publish/subscribe mechanisms to integrate OMNeT++ with Gazebo, represented at Fig. 1. Each OMNeT++ node represents a car's network interface and contains a Vehicle Data Provider (VDP) and a Robot Middleware (RM). VDP is the bridge that supplies RM data from the Gazebo simulator. RM uses this data to fill ITS-G5 Cooperative Awareness Messages (CAM's) data fields (i.e Heading, Speed values) through the Cooperative Awareness Service (CaService) that proceeds to encode this data fields in order to comply with ITS-G5 ASN-1 definitions. RM also provides GPS coordinates to define the position of the nodes in the INET mobility module.

B. *Synchronization Approach*

OMNeT++ is an event-driven simulator and Gazebo a time-driven simulator, therefore synchronizing both simulators represented a key challenge. In order to accomplish this, a synchronization module was implemented in OMNeT++, to carry out this task, relying upon ROS "/Clock" topic as clock reference. The OMNeT++ synchronization module subscribes to ROS' "/Clock" topic, published at every Gazebo simulation step (i.e. every 1ms) and proceeds to schedule a custom made OMNeT++ message for this purpose ("syncMsg") to an exact ROS time, which allows the OMNeT++ simulator engine to generate an event upon reaching that timestamp and so to be able to proceed with any other simulation process that should be running at the same time (e.g. CAM generation by CAService).

C. *Data Workflow*

Fig. 2 presents a quick overview on how data flows from carX's sensors into carY's control application, working its way through Gazebo into OMNeT++ and then into other Gazebo's car following a CAM transmission between different nodes in OMNeT++. To note that nodeX and nodeY represent the network interface of both carX and carY, respectively.

To evaluate the framework's stability and limits, we analysed its inherent latency and/or computing delays. Fig. 4 presents the delay between an OMNeT++ node reception of a CAM (from a network transmission) and its reception by the Gazebo's vehicle model, after receiving it via ROS Pub/Sub mechanisms. Following Fig. 2 timeline, the timestamps recorded were taken at CAM reception at the node's Middleware and upon Gazebo's car application callback on this referred ROS topic, at different CAM sending frequencies (10, 5, 3.3 and 2.5 Hz). From what we can extract from the recorded data, this delay mostly coming from the ROS underlying Pub/Sub mechanisms, doesn't seem to be severely affected by the CAM sending frequency. Despite the maximum obtained delay slightly increased with traffic, the observed latency close to 0.25 milliseconds, is not sufficient to impact or compromise the application under test.

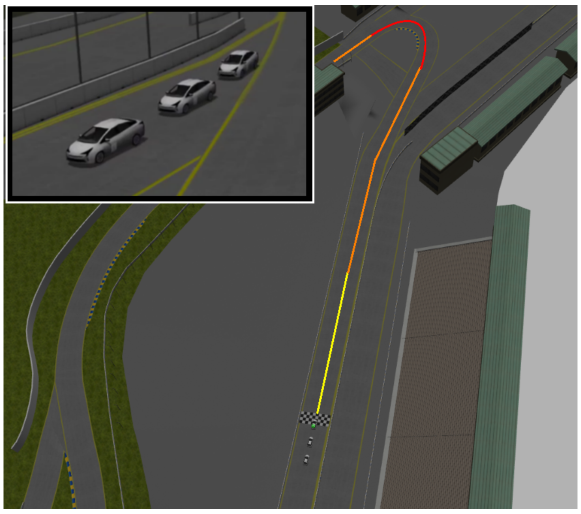


Fig. 3: Platooning Trajectory

IV. Experimental Results

The simulation is composed of three vehicles, modeled from a Toyota Prius, running a PID-based platooning control model [6] that solely relies on CAM messages to maintain the platooning service, with a safe distance set to 8 meters. The simulation results were extracted from 45-seconds long

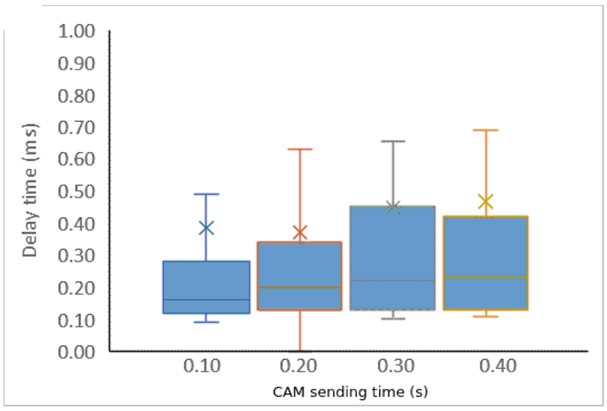


Fig. 4: CAM Exchanging Delay - Scenario A

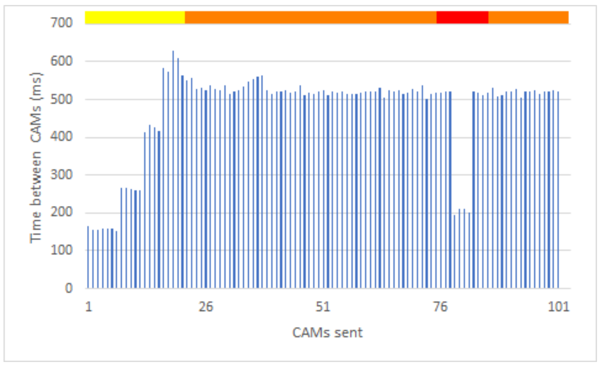


Fig. 5: Period CAM BSP

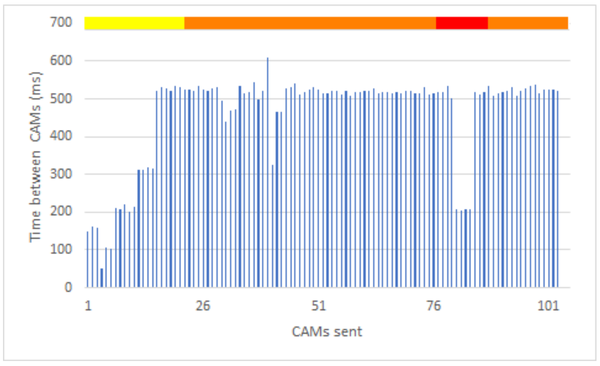


Fig. 6: Period CAM BSP for Platoon

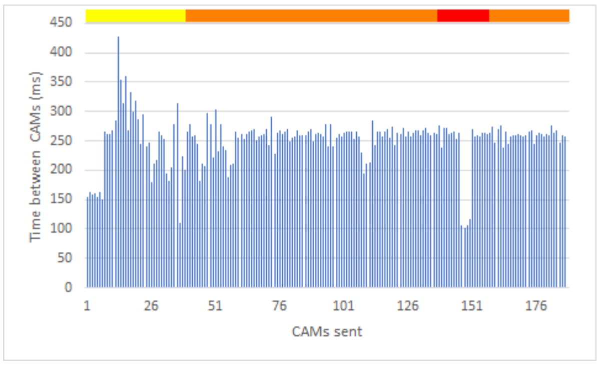


Fig. 7: Period CAM Custom SP

runs, in four different scenarios where the platoon safety was assessed: in scenario A, we setup fixed CAM frequencies, in scenario B the standard Basic System Profile (BSP) from ETSI [14] was used, in C we used the BSP with platooning-defined specifications [4] and in D we defined and evaluated customized settings for BSP. The simulation environment and simulated platooning trajectory in the 45-second run is represented in Fig. 3. The yellow line represents the initial acceleration path, where the follower car is still accelerating to reach the setpoint distance (8 meters) between itself and its leader. In orange, we represented the path in which the platooning is stable, and in red, a hard turn in which platooning behaviour is greatly dependant on the number of CAMs exchanged. The presented experimental results also contain this color reference to help relating them with the relevant portion on the track.

A. *Scenario A: Fixed CAM frequencies*

Four CAM sending frequencies were evaluated (i.e. 10, 5, 3.3 and 2.5 Hz), guaranteeing that at the highest CAM frequency, CAM messages will always be provided with fresh information. Fig. 8 shows the vehicle inter distance in each test and Fig. 10 presents the steering angles. We analyzed the impact of different CAM exchanging frequencies on the behavior of the second car, regarding the forward distance and steering angles to analyze how different CAM exchanging frequencies affected the CoVP control. The CoVP starts from parked position, and the follower only engages platooning after the leader starts moving forward, thus the follower needs to accelerate to catch up to its leader. It is also clearly noticeable that, for a CAM inter arrival time of 0.4 s, while approaching the left hard turn (in red), the follower lost track of the leader vehicle, making a full-stop. At higher CAM sending frequencies, we can observe that the CoVP PID controller shows better stability, and the inter-distance stability improves. These issues are also particularly visible regarding the steering behavior. For the first three CAM inter arrival times, the steering angles follow the leader's with a slight delay, which increases with frequency. For an inter arrival time of 0.4 s, the steering angles of the follower are no longer inline with the leader's (Fig. 10). CAM sending frequency is too low to keep the follower updated with leader's steering corrections, resulting in minimal or nearly non-existent steering inputs. Upon entering the left U turn, the follower's controller struggles to keep up with the steering of the leader, while it completely fails to do so for inter arrival times of 0.4 s. Among the several runs, at different frequencies, we notice a consistent behaviour, in such way that the higher the CAM sending frequency, the stable the PID steering control. However, fixing a CAM frequency represents a sub-optimal approach for CoVP, considering that excessive CAM traffic will often be generated, which can negatively impact the throughput of the network. With this in mind, ITS-G5 proposed BSP to dynamically trigger CAMs. In the following scenarios we evaluate its performance in the same context.

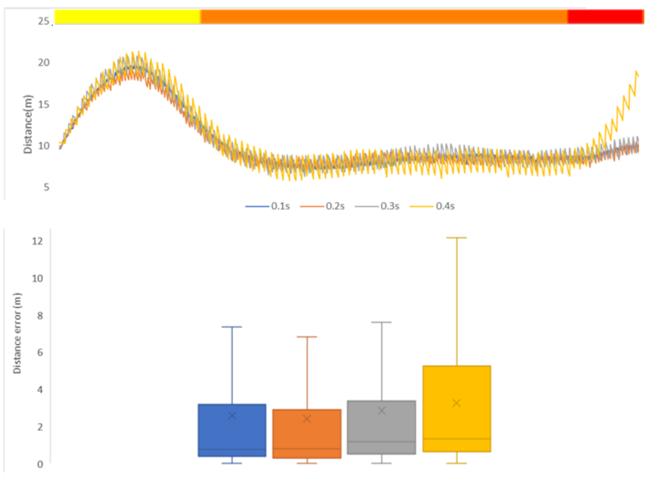


Fig. 8: Vehicle inter-distances - Scenario A

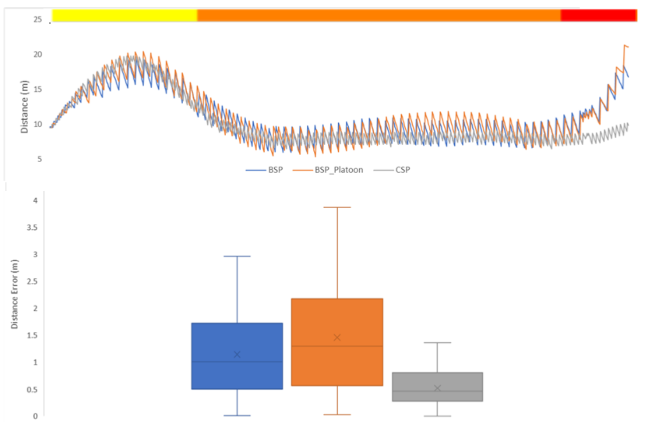


Fig. 9: Longitudinal distances analysis in different scenarios

B. *Scenario B: Basic System Profile*

For this Scenario we analyzed the BSP as standardized in ITS-G5 [14]. This profile defines the frequency at which CAMs should be triggered, considering vehicles' dynamics. BSP defines an interval of 0.1 seconds to 1 second between CAMs, except upon one of the following conditions, at which a CAM message must be immediately triggered [14]:

- the absolute difference between the current heading of the originating vehicle and the heading included in the CAM previously transmitted by the originating vehicle exceeds 4 degrees;
- the distance between the current position of the originating vehicle and the position included in the CAM previously transmitted by the originating vehicle exceeds 4 m;
- the absolute difference between the current speed of the originating vehicle and the speed included in the CAM previously transmitted by the originating vehicle exceeds 0,5 m/s.

CAM reception intervals are presented in Fig. 5. CAM sending frequencies approach 2.0 Hz, mostly due to the second triggering condition, since the CoVP speed during the orange straight part of the track is constant around 8 m/s. However, there are some high frequency triggers in the early iterations, resulting from the quick acceleration at the initial portion of the track, while trying to close the distance gap to the leader. It is also possible to observe that BSP triggers higher frequencies in response to the hard left turn (red portion of the track), that quickly shifts the leader's heading. Still, as observed in Figures 11 and 9 this increase in frequency was insufficient to maintain a stable CoVP control using this control model, and fails to follow leader's steering control. Therefore, we conclude that BSP is not well-tuned for more demanding CoVP scenarios, in which the control models exclusively rely upon cooperative support. While still trying to minimize network usage, and maintaining stable platooning control, we analyze and improve on the BSP settings in the following scenarios.

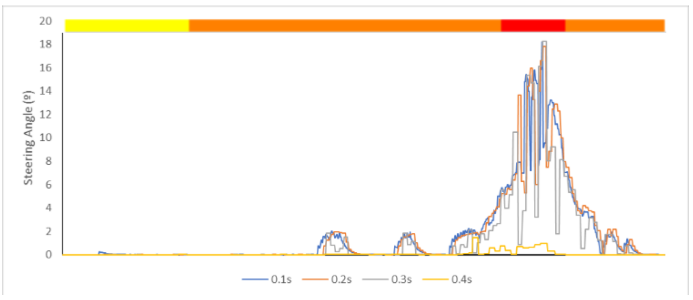


Fig. 10: Steering Angles - Scenario A

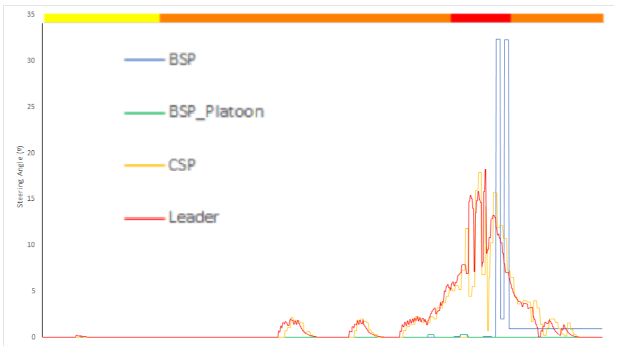


Fig. 11: Steering Analysis for different scenarios

C. *Scenario C: Basic System Profile for platooning*

In this scenario, we analyze an extension to the ITS-G5's BSP specified in [4], which recommends improved BSP settings for platooning scenarios. One of its most significant changes, was to limit the minimum frequency between CAM transmission to 2 Hz, double of the one defined for the original BSP. Test results are quite similar to the usage of the original BSP settings, as triggering conditions remain the same. As depicted in Fig. 6 and similarly to scenario B, CAM inter arrival times remain around 2Hz, in this case as a result of the minimum frequency limit set. Concerning CoVP behaviour. Figures 11 and 9 depict a similar behaviour to scenario B, which results in a failure to execute the U turn. This is a consequence of platoon instability. Concerning distance error in regards to the set point, for instance, both scenarios B and

TABLE I: Comparison between Scenarios - Number of messages and Safety Guarantee

Scenario	**Fixed Frequencies**				**BSP**	**BSP Plat.**	**CSP**
	10	5	3.3	2.5			
Message	441	227	151	113	101	101	181
Safety	OK	OK	OK	NOK	NOK	NOK	OK

C, present similar and significant errors, resulting from low CAM update frequency.

D. Scenario D: Custom System Profile for Platooning

For this scenario we setup a Custom System Profile aiming at balancing the network load originated by CAM exchanging, while guaranteeing stability. With this in mind, our approach was to adapt the second CAM triggering condition mentioned at scenario B, by changing it to 2 meters instead of 4 meters. This change impacted the CoVP behaviour considerably, both in the number of CAMs sent and its frequency, as it's possible to check at Fig. 7. As shown in Figures 5 and 6, the CSP conditions caused CAM triggering to happen much more frequently than in previous cases, resulting on a more stable CoVP control when compared to scenarios B and C (Figures 11 and 9). This also translates into a significant decrease of distance errors to the leader, leading to a smoother control. In facto, only this changed enabled the CoVP to successfully complete the hard left turn (Fig. 3).

Table I presents the number of CAM messages sent during simulation for each scenario. As shown, a fixed frequency between 3.3 Hz and 2.5 Hz should be at the threshold borderline balance to maintain CoVP control. However, fixing this frequency is not the most reasonable approach since it can cause unnecessary CAM message transmissions. With this in mind, a System profile approach as defined in ITS-G5 should be the optimal way to handle this, however, as we were able to confirm with scenarios B,C and D this kind of profiling should be adapted to the use-case and particularly to the control model. For this particular control model under test, CAM information availability is crucial to maintain a stable behaviour. This kind of profiling can be easily carried out using our framework, by fully-specifying the simulation environment and CoVP control model over ROS/Gazebo, while using OMNeT++'s capabilities to analyze the network performance and to provide new extensions to the ITS-G5 communications stack, carrying out an integrated in depth analysis of CoVP behaviours.

V. Conclusions and Future Work

This paper proposes a sub-microscopic framework for cooperative driving simulation, integrating the Gazebo simulator and ROS robotics framework, with the OMNeT++ network simulator. Using this framework, as a preliminary proof-of-concept, we implemented and validated different scenarios to evaluate the behaviour of a CoVP control model, exclusively dependant on CAM exchanging. We analyzed the impact of different CAM exchanging frequencies and ITS-G5 BSP recommendations to validate the correctness of the simulation framework.

COPADRIVe successfully enabled this analysis, within a rich and realistic simulation environment, both from the control and communications perspective. We plan to complement these analysis with relevant network performance results and to increase the complexity of the scenario and CoVP control model. At the communications level we will be including external traffic sources to evaluate this and other CoVP models in congestion scenarios.

We firmly believe this tools has the potential to support advanced realistic ITS cooperative autonomous driving scenarios, and to help reducing technology validation effort and cost.

References

[1] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen, "A Survey on Platoon-Based Vehicular Cyber-Physical Systems," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 263–284, 2016.

[2] A. Talebpour and H. S. Mahmassani, "Influence of connected and autonomous vehicles on traffic flow stability and throughput," *Transportation Research Part C: Emerging Technologies*, vol. 71, pp. 143–163, Oct. 2016.

[3] P. Fernandes and U. Nunes, "Platooning With IVC-Enabled Autonomous Vehicles: Strategies to Mitigate Communication Delays, Improve Safety and Traffic Flow," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 91–106, Mar. 2012.

[4] ETSI, "ETSI TR 102 638," ETSI, Tech. Rep. V1.1.1, 2009.

[5] O. Karoui, M. Khalgui, A. Kouba, E. Guerfala, Z. Li, and E. Tovar, "Dual mode for vehicular platoon safety: Simulation and formal verification," *Information Sciences*, vol. 402, pp. 216–232, Sep. 2017.

[6] O. Karoui, E. Guerfala, A. Koubaa, M. Khalgui, E. Tovard, N. Wu, A. Al-Ahmari, and Z. Li, "Performance evaluation of vehicular platoons using Webots," *IET Intelligent Transport Systems*, vol. 11, no. 8, pp. 441–449, Oct. 2017.

[7] Zhiwu Li, Oussama Karoui, Anis Koubaa, Mohamed Khalgui, Emna Guerfala, and Eduardo Tovar, "System and method for operating a follower vehicle in a vehicle platoon," Polytechnic Institute of Porto (ISEP-IPP), Portugal, Tech. Rep. CISTER-TR-181203, 2018.

[8] R. Riebl, H. Gnther, C. Facchi, and L. Wolf, "Artery: Extending Veins for VANET applications," in *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, Jun. 2015, pp. 450–456.

[9] M. Rondinone, J. Maneros, D. Krajzewicz, R. Bauza, P. Cataldi, F. Hrizi, J. Gozalvez, V. Kumar, M. Rckl, L. Lin, O. Lazaro, J. Leguay, J. Hrri, S. Vaz, Y. Lopez, M. Sepulcre, M. Wetterwald, R. Blokpoel, and F. Cartolano, "iTETRIS: A modular simulation platform for the large scale evaluation of cooperative ITS applications," *Simulation Modelling Practice and Theory*, vol. 34, pp. 99–125, May 2013.

[10] B. Schnemann, "V2x simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems," *Computer Networks*, vol. 55, no. 14, pp. 3189–3198, Oct. 2011.

[11] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. L. Cigno, "Plexe: A platooning extension for Veins," in *2014 IEEE Vehicular Networking Conference (VNC)*, Dec. 2014, pp. 53–60.

[12] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Fltterd, R. Hilbrich, L. Lcken, J. Rummel, P. Wagner, and E. WieBner, "Microscopic Traffic Simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 2575–2582.

[13] I. Llatser, G. Jornod, A. Festag, D. Mansolino, I. Navarro, and A. Martinoli, "Simulation of cooperative automated driving by bidirectional coupling of vehicle and network simulators," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 1881–1886.

[14] ETSI 2018, "ETSI EN 302 637-2," ETSI, Tech. Rep. V1.4.0, Aug. 2018.