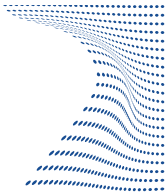
ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR



Doctoral Dissertation
Doctoral Program in Electric, Electronic and
Communication Engineering (XXXIII cycle)

Safety Applications and Measurement Tools for Connected Vehicles

Marco Malinverno
* * * * * *

Supervisors
Prof. Claudio Ettore Casetti, Supervisor
Prof. Carla Fabiana Chiasserini Co-supervisor
Prof. Nicola Amati Co-supervisor

Doctoral Examination Committee:
Prof. Claudia Campolo, Università Mediterranea di Reggio Calabria
Prof. Michele Segata, Libera Univesità di Bolzano
Prof. Pietro Manzoni, Universitat Politècnica de València
Prof. Raphael Frank, Université du Luxembourg
Prof. Paolo Giaccone, Politecnico di Torino

Politecnico di Torino
February, 2021

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Marco Malinverno
Turin, February, 2021

Summary

The automotive industry is in the middle of a technological revolution, which in the coming years will see every vehicle connected through Vehicle-to-Everything (V2X) technologies. The most important standardization bodies have done their part to usher in this slow but inexorable transition: IEEE, with the WAVE stack (Wireless Access in Vehicular Environment) and ETSI, with the ITS-G5 stack, have contributed to defining the foundations of the ITS (Intelligent Transportation System) scenario for US and Europe. The network access technology to be adopted to enable such communications is at the center of a major debate in the scientific community: IEEE proposed a protocol coming from the 802.11 wireless-LAN family, while 3GPP proposed a solution derived from the cellular networks (C-V2X).

All these technologies will enable an incredible number of applications, which will overturn the mobility experience in all of its forms. The main topics of this thesis are V2X communications, and after an initial analysis on the main solutions already developed, it presents and discusses original contributions ranging from the world of network simulation to that of V2X embedded devices.

Simulations tools plays a pivotal role in the automotive industry: because of the complexity and the high deployment costs of vehicular applications, it is usually convenient to extensively test them by simulation. This thesis introduces MS-VAN3T, an open source ETSI ITS-G5 model for the ns-3 simulator that, coupled with SUMO (Simulation of Urban Mobility), allows the reproduction of complex vehicular scenarios. MS-VAN3T can be used to develop any kind of application and comes with the possibility of transparently changing the underlying access technology.

Day-0 applications will leverage V2X technologies to improve road safety. For this reason, MS-VAN3T is used in this thesis to develop a collision avoidance system that, leveraging Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs) can warn vehicles and vulnerable users about possible future collisions. The information generated by the system is then used to build an automatic strategy, which allows vehicles to autonomously assess the collision risk and take appropriate reactions to avert the collisions.

In response to the growing demand for V2X solutions enabling the aforementioned applications, this thesis proposes an open source testing platform that can be used to assess the performance of V2X Wireless Network Interface Cards (WNICs), and that

enables a fully working 802.11p communication. The proposed platform is composed of a patched modified version of OpenWrt, an OS that is widely used for embedded devices, and it is used to test the throughput, the radio range and other important KPIs of off-the-shelf low-cost WNICs.

One of the key parameters for vehicular networks, which must be constantly measured to guarantee a reliable service, is latency. Being capable of measuring such a parameter can be very important in providing applications with acceptable performances. Although several solutions exist in literature, such as `ping` or other measurement platforms, they are typically bound to a specific protocol, such as ICMP, or require additional hardware or software capabilities other than the testing application. In this thesis, a novel lightweight, flexible, and custom latency measurement protocol is presented. The protocol, named LaMP (Latency Measurement Protocol), is completely agnostic of lower-layer protocols and enables micro-second precise latency measurements. The first open source tool leveraging LaMP, called LaTe (Latency Tester), is presented as well. LaMP and LaTe, which are constantly updated and added with new features, are used to validate and test the above-mentioned V2X platform. This thesis presents the most important results of such an analysis, and highlights the importance of open source solutions for the performance assessment of technologies for connected vehicles.

Acknowledgements

This work was fully supported by the interdepartmental center CARS@PoliTo, the Center for Automotive Research and Sustainable mobility of Politecnico di Torino.

Contents

Chapter 1

Introduction

The first heartbeat of the automotive industry is dated back in 1886, when the German engineer Karl Benz had the crazy idea of creating a vehicle for the transport of people capable of moving without the need for horse towing, paving the way to one of the most robust and resilient industry in the world. After around twenty years, in 1913, Henry Ford, a man who was determined to build *motor cars for the multitude*, installed the first assembly line for the mass production of an entire automobile: its invention pulled down drastically the time and the costs of the industry and delivered into the hands of the people the first affordable motorized vehicles. Nowadays, after the advent of the new communication techniques and with the raise of the so-called Industry 4.0, the concept of vehicle can no longer be associated with that of a merely mechanical device. The vehicles today are something more complex, devices that take advantage of a numbers of different technologies that are constantly shifting the concept of mobility toward a safer, greener and smarter paradigm.

During the last couple of decades, the academic and industrial players devoted to the automotive world developed an increasing interest on Vehicular Ad-Hoc Networks (VANETs). Because of their peculiar characteristics, it was not possible to reuse the existing communication protocols: extremely dynamic topologies, variable densities, high bandwidth demand with ultra low latency expected, and relatively high power availability, make vehicular networks something more than a derivation or a variant of Mobile Ad-Hoc Networks (MANETs). For these reasons, the main standardization bodies focused their resources in creating new solutions and new protocols able to bear with the tight specifications that the multitude of applications leveraging vehicular communications will require.

The concepts of Vehicle-to-Everything (V2X) and of connected vehicles encompass all the communications demands of the vehicular world. V2X refers to a technology (or, more precisely, to a set of technologies) enabling the wireless data exchange between the vehicle and any other entity in its surroundings. Therefore, the general definition of V2X can be separated into a multitude of sub-cases depending on the particular field

of application. The most important include: Vehicle-to-Vehicle (V2V) for the inter-vehicles communications; Vehicle-to-Pedestrian (V2P) to facilitate the interaction with pedestrians; Vehicle-to-Infrastructure (V2I) referred to the communications with the road-side facilities as well as to the communications with the telecom operator infrastructures; Vehicle-to-Network (V2N) for the communication with the network-based services and applications. The term connected vehicles, instead, identifies those vehicles equipped with V2X sensors.

The two main standardization bodies competing in this field of research are the European Telecommunication Standards Institute (ETSI) and the Institute of Electrical and Electronics Engineers (IEEE), whose efforts resulted in two standards defining as many communication stacks for vehicular networking. The solution proposed by ETSI, mainly pushed in the European market, is the so-called Intelligent Transportation System (ITS)-G5; on the other hand IEEE proposed for the US market the Wireless Access for Vehicular Environment (WAVE) protocol stack. The two standards are different implementation of the so-called Dedicated Short-Range Communications (DSRC) and are based on the IEEE 802.11p access layer, a particular Wi-Fi-based amendment thought for the vehicular world. Recently, an emerging access technology named C-V2X (Cellular-V2X) was proposed by the Third Generation Partnership Project (3GPP), promising a solution natively integrated with the raising 5G networks, able to tackle the demanding requirements of vehicular applications. The solution proposed by 3GPP has been designed to have the flexibility of replacing 802.11p in DSRC protocols.

Since the proposal of C-V2X, the scientific and industrial communities are struggling to find out which, among the two access solutions proposed, is the one that best suits the vehicular scenarios needs. Some of the studies focuses on the protocols' network performances, proposing analysis on the Packet Drop Ratio (PDR), end-to-end latency, throughput etc., while some others concentrate on the protocols' maturity, on their scalability or on their integration with the next generation networks. The vast majority of these studies, with the exception of some rare cases, are performed in laboratory using simulation tools. Using these software it is possible to recreate complex scenarios, even with thousands of connected vehicles, allowing the analysis of the network performances and the prototyping of innovative applications at extremely reduced costs.

In this thesis, a novel simulation framework for vehicular networking is developed and released as an open source project; the proposed framework, named MS-VAN3T, is then used to prototype and test some of the most interesting use cases for vehicular communication, such as a Intersection Collision Risk Warning (ICRW) application for vehicles and vulnerable users. At the same time, the need for accessible and affordable V2X solutions triggered the development of open source software and hardware platforms for the assessment and analysis of communication performances in vehicular scenarios.

1.1 Research motivation and objectives

Provided the pivotal role of vehicular communications in the future of mobility and the growing attention that the industrial and academic worlds are taking in vehicular applications, this thesis aims at developing innovative software and hardware solutions for V2X, and to provide tools and frameworks capable of speeding up the applications prototyping and testing.

The intense usage of network and mobility simulators for the analysis of vehicular applications have paved the way to the development of an open source collection of V2X frameworks for ns-3 (network simulator 3), named MS-VAN3T. Beyond the benefits that a repository including all the state-of-the art frameworks for vehicular communication can bring (above all, in terms of development and coding time), the repository comprehends a novel model for the ETSI ITS-G5 stack. By using the tool provided it is possible to test any kind of vehicular application by changing dynamically the underlying access protocol, switching from 802.11p to C-V2X or, if a centralized solution is required, to LTE. Furthermore, the ITS-G5 model produces ETSI-compliant packets that can be easily encapsulated, through the use of appropriate hardware-in-the-loop techniques, into real packets, turning the proposed system into a V2X emulator.

As the next step of this work, the aforementioned framework is used to develop a Collision Avoidance Service. This application is conceived to exploit the information present in the vehicular messages to identify and, eventually, alert vehicles and pedestrians set on a collision course. The analysis has been performed by varying the underlying access schemes, the technologies' penetration rate and other important parameters, and confirmed the importance of developing new solutions (especially in the field of vehicular safety) for V2X communications.

Alongside with the development of applications and frameworks in a simulated environment, this thesis aims to collect and present the efforts that have been made to realize affordable solutions for the implementation and assessment of vehicular communications using V2X-enabled embedded devices. Basically, a solution based on off-the-shelves available devices is presented. The proposed architecture, composed both by hardware and software components, allows the creation of standard-compliant 802.11p devices that may be used as access layer either for WAVE or ITS-G5 implementations; the software running on top of it, a custom version of a Linux distribution named OpenWrt, is open source and available for download in GitHub. Being latency one of the key parameter to be analysed in V2X networks, especially in safety-related applications, a novel application-layer protocol for latency measurements is then proposed. This protocol, named LaMP (Latency Measurement Protocol), provide timestamps with micro-seconds granularity that can be used to perform precise measurements. The first applications running on top of LaMP, named LaTe (Latency Tester), has been developed and used to assess and measure the performance, in terms of latency, of the proposed hardware platform.

1.2 Main contributions

The main contributions of this thesis can be summarized as follows:

- Study and analysis of the main solutions proposed by ETSI, IEEE and 3GPP for vehicular communications, namely WAVE, ITS-G5, 802.11p and C-V2X. This thesis gives a panoramic on the technologies and solutions that have been developed during the last couple of decades, for the world of connected vehicles and, in general, of V2X. This part includes a description of the various layers composing IEEE WAVE and ETSI ITS-G5, trying to highlight similarities and differences.

- Design, development and validation of a multi-stack framework for vehicular networks in the ns-3 simulator, called MS-VAN3T. The proposed solution provides the scientific community with a powerful simulation tool that implements the ETSI ITS-G5 stack, and that includes a collection of the state-of-the-art access layer models for V2X. MS-VAN3T includes an integration with SUMO (Simulation of Urban MObility), and comes with a couple of applications (one for V2I, the other for V2V communication), which can be used by fellow researchers as a development baseline; additionally, the project is entirely open source and published on GitHub [**github-v2i**, **github-v2v**]. At the time of writing, MS-VAN3T is the only solution for vehicular simulation that implements the ETSI ITS-G5 stack in ns-3 and that allows the user to transparently switch the underlying access technology.

- Performance evaluation of the V2X access technologies present in MS-VAN3T. The proposed framework is used to test the available models (namely, LTE, 802.11p and C-V2X) for what concerns delay, delay jitter and Packet Drop Ratio (PDR). The results are then analyzed to draw conclusions on each technology, and to highlight their advantages and weaknesses.

- Design, development and testing of a Collision Avoidance Service (CAS) based on the information present in vehicular messages. The idea behind CAS puts its basis on the Intersection Collision Risk Warning application defined by ETSI in TS 101 539-2 [34]; the service is thought to offer protection to collisions of both vehicles and vulnerable users, and is developed in a centralized and distributed version (thus, deployed in V2I, V2V and V2P scenarios). CAS leverages on the information carried by Cooperative Awareness Messages (CAMs), which are periodic messages that every ITS-G5-compliant entity must generate to advertise its information (kinetic, status, role, etc.) to the surrounding neighbors. CAS extracts the position, speed, heading and acceleration information from CAMs and use them to project the trajectory in the future. Each generated trajectory is analyzed and, in case a future collision is detected, the warning information is encoded and sent via Decentralized Environmental Notification Messages (DENMs), which are messages defined by ETSI carrying information about asynchronous events. As for now, there are no other solutions that have been developed to mitigate the collision risk

(between vehicles and between vehicles and vulnerable users) leveraging only on the information exchanged via radio interface.

- Development and testing of an automatic collision avoidance strategy that, based on the information generated by CAS, can override the control of the vehicle and change its trajectory to avert the collision. The whole architecture (CAS and evasive strategy) is implemented in MS-VAN3T, and tested through extensive simulation campaigns. The results are encouraging and highlight the importance of developing new safety applications relying on vehicular communication.

- Design, development and validation of an open source platform, based on off-the-shelves devices and on the Linux distribution OpenWrt, for the IEEE 802.11p access layer implementation and analysis. The project, available on GitHub [77], can potentially be used by anyone who is interested in building custom V2X devices and to test the performances of commercial Wireless Network Interface Cards (WNICs). In this thesis, the testing platform is built using PC Engine's APU1D embedded boards and Unex's DHXA-222 WNIC. The system is then analyzed in its network performances, including throughput, packet loss, radio range and experienced delay.

- Design and development of an application-layer protocol thought for micro-seconds precise latency measurement, named LaMP. This protocol is developed to match the need of precise and accurate measurements of the delay experienced at application layer. Other solutions, such as the widely-known ICMP protocol, have the problem of returning a latency value that is not the same that would measure an application running on top of the same system. Indeed, ICMP is implemented at layer 3 of the ISO/OSI stack, and some of the actions needed to deliver the information at the higher layers are actually time consuming. Therefore, protocols like LaMP are fundamental in the precise assessment of network performances of vehicular networks, but also in the assessment of the network performance of the myriad of applications enabled by URLLC (Ultra Reliable Low Latency Communications, one of the tenet of 5G).

- Implementation of the first application running on top of LaMP, named LaTe; the proposed software can be used to assess the latency that an application running on the device under test would experience, with a micro-second granularity. LaTe, that at the time of writing gives the possibility of testing the system on UDP over IPv4 in wired and wireless interfaces, can also be used to assess the performance of the EDCA (Enhanced Distributed Channel Access) implementation by selecting the Access Category where the LaMP messages are exchanged. In this work, LaTe is also used to validate, from the point of view of the latency, the aforementioned open source platform for V2X-enabled embedded devices. Also in this case, the entire project is open source, with the official LaMP specification and the LaTe source code that can be found in [58].

Part of the work included in this thesis was presented through several contributions in international conferences renown in the VANETs community. In particular, the project of the simulation framework MS-VAN3T was included in the paper "A Multi-stack Simulation Framework for Vehicular Applications Testing" [66] presented in the workshop DIVANet (hosted at ACM MSWiM'20). The development of the vehicular collision avoidance system is discussed and presented, using different simulation framework and in different architectures, in the paper "Performance Analysis of C-V2I-Based Automotive Collision Avoidance" [64] (WoWMoM'18) and in "Support of Safety Services through Vehicular Communications: The Intersection Collision Avoidance Use Case" [13] (AEIT Automotive'18). The extension of the service to vulnerable users, together with a study on the possible network architecture enabling the system, was firstly proposed in "Edge-Based Collision Avoidance for Vehicles and Vulnerable Users: An Architecture Based on MEC" [63], published in the peer-reviewed magazine IEEE VTM. The development of collision-averting strategies enabled by V2X-based safety services was instead proposed, although following different schemes, in "An Edge-Based Framework for Enhanced Road Safety of Connected Cars" [65], a work published in IEEE Access in 2020.

The first implementation of the platform for the evaluation and the performance analysis of V2X-enabled embedded devices was instead proposed in "Characterization and Performance Evaluation of IEEE 802.11p NICs" [82], presented in the ACM MobiHoc workshop TOP-Cars'19. The aforementioned testing platform was used for the realization of an interactive demonstration, hosted in WoWMoM'19, where the 802.11p-based testbed was used to enable the multiplayer gaming sessions running in two PCs. The demonstration attendees were able to play an online multiplayer game, with the PCs directly connected through 802.11p in OCB mode ("Demo: Open Source Platform for IEEE 802.11p NICs Evaluation" [83]). The LaMP and LaTe projects (which are actively being developed), were presented in the VTC-2019-fall conference, in the paper "A Flexible, Protocol-Agnostic Latency Measurement Platform" [81], which introduces the basic idea behind the custom protocol and the general architecture of the measurement software.

Moreover, some research contributions were supported and financed by European Projects, such as the work on the Collision Avoidance Service, supported by 5G-TRANSFORMER (ID 761536) and the work on LaMP and LaTe, financed by 5G-CARMEN (ID 825012).

1.3 Outline of the thesis

This thesis starts, in Chapter 2, with a detailed description of the protocol stacks for VANETs proposed by IEEE and by ETSI (respectively, WAVE and ITS-G5). The description of the various layers of the two solutions follows a top-down approach, starting from the application layer and by going down in the stack to the physical layer. This part gives also an overview on the two V2X access technologies, namely IEEE 802.11p

and 3GPP C-V2X, and on the current state-of-the-art on VANETs.

In Chapter 3, the multi-stack simulation framework MS-VAN3T is presented and described in detail. Two applications, namely *area speed advisor* and *emergency vehicle alert*, are used to showcase the potentiality of the V2I and V2V communication models and to provide fellow researchers with a solid development baseline.

Chapter 4 introduces and describes the CAS system in its centralized and distributed version. Moreover, it proposes a novel strategy that, based on the information generated by CAS, can take automatic actions avert the collisions.

The open source platform for V2X-enabled embedded devices, and the LaMP and LaTe projects are presented in Chapter 5. This part describes the hardware and software components used to build the platform; the LaMP protocol is developed starting from the need for micro-second precise latency measurements. It is used to develop the software LaTe, an open source tool which is used, in this thesis, to measure the latency of the V2X platform.

Finally, Chapter 6 concludes the thesis.

Chapter 2

Communication protocols for connected cars

The fundamental idea behind the concept of connected cars, and in general of VANETs, is to enable the connections among all the road players (e.g. cars, bicycles, road signs, semaphores, etc.) and to improve the current mobility experience through a myriad of applications, ranging from safety to infotainment. The pivotal element of such a vision is a strong, robust and scalable vehicular network, responsible for the fulfillment of the tight requirements imposed by the overlying applications.

The concept of a vehicular communication system leveraging the wireless technology is not new, but in the last years we have witnessed an important increase in the efforts made by the scientific and industrial communities. The main reasons for this sprint have to be found on the last decades' advances in the communication technologies, on the profuse standardization efforts made by ETSI, IEEE and 3GPP, and on the slow but inexorable shifting toward the future of mobility.

This chapter is devoted to the presentation of the protocols that have been realized for the vehicular networks. After a brief introduction on the concept of V2X, the two communication stacks proposed by IEEE and ETSI, namely WAVE and ITS-G5, are illustrated following a top-down approach. Then, the two access layer solutions, IEEE 802.11p and 3GPP C-V2X are presented.

2.1 V2X - Vehicle to Everything

As the name suggests, V2X refers to a bunch of communication technologies enabling omnidirectional connection between the vehicle and its surroundings. An non-exhaustive subset of use cases for V2X are Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P) and Vehicle-to-Network (V2N). The aim is to create links among the various players of transportation, such as vehicles, trucks, pedestrian, road

and network infrastructure, to push the creation of innovative applications in the context of Intelligent Transportation System (ITS) and, in general, to promote the development of new models and paradigms for the automotive mobility.

The applications enabled by vehicular networking can be classified into 3 main groups: 1) *Road safety applications*, 2) *Traffic efficiency and management applications* and 3) *Infotainment applications* [56]. Road safety applications, due to the socio-economic impact that car accidents and car injuries have worldwide, are the most promising in terms of market penetration. Groundbreaking services such as Intersection Collision Avoidance, Emergency Vehicle Warning and Cooperative Lane-Merging, are going to be effective even if entirely based on vehicular communication systems (i.e., without the integration or the fusion with other sensors such as cameras or LIDARs), and are designed to tackle the global plague of road accidents (that, according to WHO, kill more that 1.35 million people per year [42]). Traffic efficiency and management applications focus instead on efficiently manage the traffic flow, to reach the goals of a more sustainable and smart mobility. Some example of these applications are Green Light Optimal Speed Advisory, Cooperative Adaptive Cruise Control and Platooning. Infotainment services, which will include Social Network feature integration, gamification strategies or marketing purposes will instead boost the final user experience, enabling a multitude of innovative services that will change the user perception of mobility.

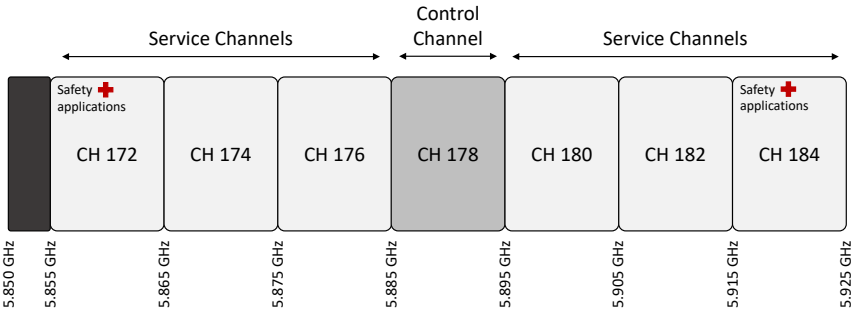
At the time of writing, the available communication protocols for V2X can be either based on Wireless Local Area Network (WLAN) or on cellular network. WLAN-based standards, such as WAVE and ITS-G5, use IEEE 802.11p as access layers. Cellular-based standards such as C-V2X, reuse instead the upper layer standardized by IEEE and ETSI, by changing the access layer and enabling a native communication with the cellular network infrastructure.

2.1.1 Current spectrum regulation for ITS

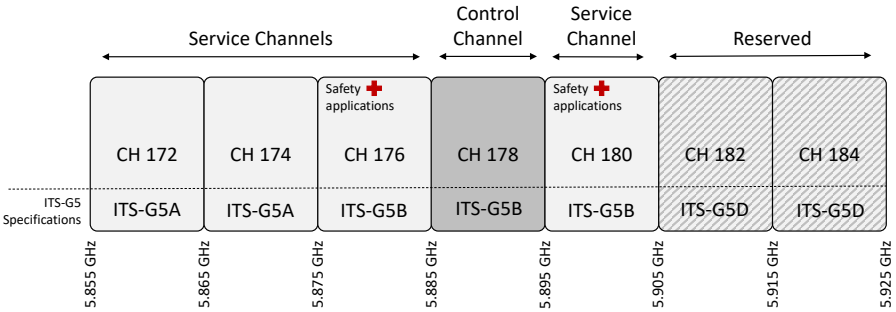
To promote and facilitate the development of vehicular technologies, the various spectrum management organizations dedicated the 5.9 GHz band for the exclusive usage of ITS services.

In US, the spectrum has been reserved by the Federal Communications Commission (FCC). FCC allocated 75 MHz for vehicular communications, from 5.850 to 5.925 GHz [39], as shown in Figure 2.1a. The band is divided into seven channels, 10 MHz each, with a 5 MHz guard band. The channels are divided into Control Channel (CCH) and Service Channel (SCH), with the first used for service advertisement and control messages, while the second is used to transmit all the applications and services data. Channels 174, 176, 180 and 182 are reserved for normal applications, while channels 172 and 184 are reserved for safety-related applications [18].

In Europe, the regulatory entity for vehicular networks is the European Commission, that with the Commission Decision 2008/671/EC [6] and with further harmonization



(a) FCC spectrum allocation for the 5.9 GHz band.



(b) EU spectrum allocation for the 5.9 GHz band.

Figure 2.1: 5.9 GHz spectrum allocation for the US and EU markets.

introduced by the European Conference of Postal and Telecommunications Administrations (CEPT), assigned the band from 5.855 to 5.925 GHz to vehicular applications [18]. The current spectrum allocation follows the recommendation in [31, 32], where the band is still divided in 7 channels as shown in Figure 2.1b. Channel 172 and 174 are used in non-safety applications, channel 178 still serves as CCH, while 176 and 180 are intended for safety applications. Channels 182 and 184 are reserved for future usage. In Europe, the de-facto standard for vehicular communication ITS-G5 introduced (through the EN 602 663 standard [28]) a fine-grained SCH assignment through the definition of 4 types of channel:

- ITS-G5A: Channels 172 and 174, for non-safety related applications.
- ITS-G5B: Channels 176, 178 and 180, for safety related applications.
- ITS-G5C: Channels in the 5.6 GHz band, for infrastructure-based broadband radio access networks.
- ITS-G5D: Channels 182 and 184 for future ITS applications.

Although in this work the focus will be on European and US standards, it is important to mention that other regulatory entities belonging to independent countries are

adopting custom policies. For example, in Korea the Telecommunication Technology Association (TTA) allocated the same spectrum with respect to Europe and US, but with different SCH and CCH division [102]. At the same time, countries like China made completely different decisions: the Bureau of Radio Regulation (BRR) stated that the only standard adoptable for ITS safety application in China is C-V2X and allocated only two channels (namely 182 and 184) [102]. In Japan, instead, safety applications communicate in a single channel allocated in the band 755.5-764.5 MHz while non-safety applications have reserved the band between 5.770 and 5.850 GHz, that is divided into fourteen channels of 5 MHz each [102].

2.2 DSRC-based protocols

As previously introduced, DSRC defines the protocol stack for the realization of V2X systems. In a DSRC system two classes of devices are defined: on-board units (OBU) and road-side units (RSU) [60]. Therefore, the type of communication enabled by DSRC are of type V2V (between OBUs) and V2I (between OBUs and RSUs).

2.2.1 WAVE

WAVE provides a communication stack that is optimized for the vehicular environment, adopting both customized and general-purpose elements. The WAVE stack is defined, in its different layers, by the set of IEEE 1609 standards, by IEEE 802.2 at Logical Link Control (LLC) layer, and by IEEE 802.11p. A data plane is defined for protocols carrying higher layer information, while a management plane is defined to support the information transfer among the different layers. Although WAVE defines also the lower layers of the communication stack (MAC - Medium Access Control - and Physical), a standard-compliant WAVE device does not preclude the device from including other radios, protocol stacks, or other communication technologies, as specified in [47]. The higher layer (Application, Presentation and Session in the ISO/OSI reference model), are not officially defined in WAVE. However, the IEEE 1609.x family of standards have been developed cooperatively with the Society of Automotive Engineers (SAE), which defined the set of messages and their utilization in the vehicular environment, especially for what concerns safety applications. The IEEE 1609.x family defines the architecture, communication model, management structure and security mechanisms for the WAVE stack, from the transport to part of the MAC layer. Finally, 802.11p describes the MAC and physical layer to be adopted. The detailed layered structure of WAVE can be seen in Figure 2.2, where the WAVE-defined architecture is enclosed in the black dashed rectangle.

This section is intended to give a high level view of the IEEE 1609.x family, reporting the main aspects and functionalities that have been standardized, without going into the details of all the procedures and interfaces. For further information, the reader is referred to the original IEEE standards [47, 48, 49, 50].

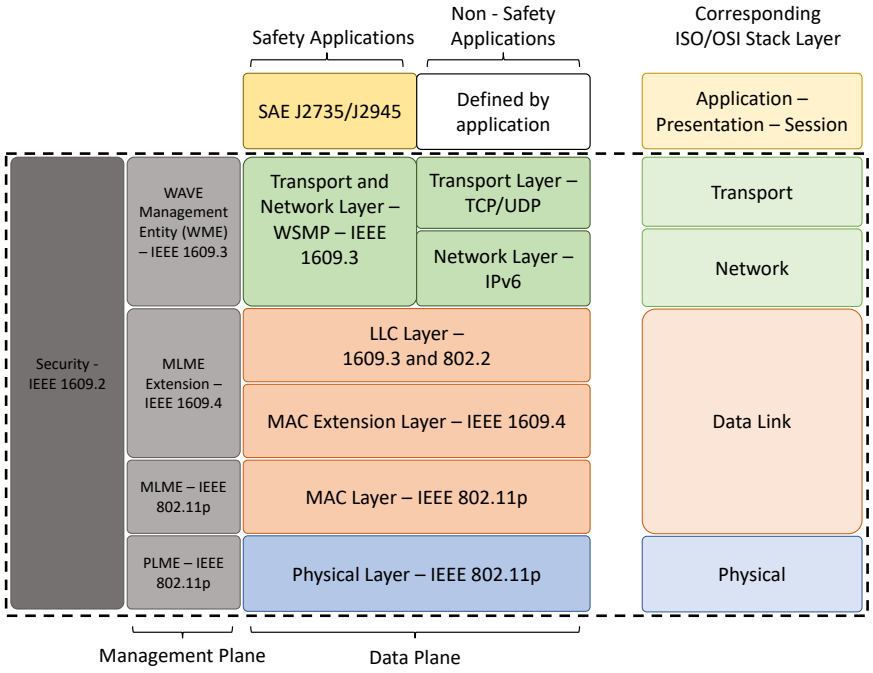


Figure 2.2: Layered structure of the WAVE protocol stack. Although the higher layers (Application, Presentation and Session of the ISO/OSI reference model) are not defined by the WAVE standards, the stack has been cooperatively designed with SAE which provided a message layer for safety-related application designed for the VANETs scenarios, namely SAE J2735 and SAE J2945.

Higher layers

Although WAVE does not specify the higher layers message format, the SAE standards J2735 and J2945 [86, 87] have been thought to meet the requirements of the DSRC environment where the resources, especially in terms of bandwidth, are limited. For these reasons, they designed a message sublayer for safety-related applications where the information are enveloped in short packets that can be frequently and efficiently broadcasted in an unacknowledged delivery mode. To maximize the overall C-ITS (Cooperative-ITS) system capacity, it is required a dense encoding of information in defining the messages, that is achieved through Abstract Syntax Notation revision One (ASN.1) encoding. In the J2735 standard, 17 message types are defined having purposes ranging from safety to GPS (Global Positioning System) correction. This work will only focus on Basic Safety Message (BSM), and will shortly introduce two alert messages, namely Emergency Vehicle Alert (EVA) and Intersection Collision Avoidance (ICA).

BSMs are used by most of the vehicular applications to exchange safety data regarding the state of the vehicle. These messages are filled with data contents required by safety and non-safety applications, and broadcasted at high frequency to surrounding

vehicles and road entities. To avoid a rapid congestion of the channel, the transmission rate is adaptive and depends on the congestion control algorithms adopted. The maximum transmission frequency is set to 10 Hz. A BSM is divided into two parts: the first shall be included in each BSM and includes information such as Latitude, Longitude, Speed, Heading, etc. The second part is optional, and includes additional safety-related data such as path history, path prediction, exterior light status etc.

EVA messages are used by emergency vehicles to broadcast warning messages notifying nearby vehicles that an emergency situation is happening in the vicinity. This type of message contains data fields to describe the event and provide advice and recommendations for the surrounding vehicles.

ICA messages are instead used to broadcast the information concerning potential collisions. They are delivered by other vehicles or by RSUs when entering a dangerous intersection without the right of way.

Transport and Network layers

The Network and Transport layer in WAVE are defined (in part) in the IEEE 1609.3 standard [49]. In particular, two different data-plane protocol stacks are defined, sharing the common Data-link and Physical layers: the first, used only by safety-related application, is the so called WAVE Short Message Protocol (WSMP), carrying WAVE Short Messages (WSM). The latter is the classic IPv6 stack, using either UDP or TCP as transport medium.

WSMP has been designed to allow vehicular applications to control the physical characteristic used for the message dissemination (e.g., transmitting power or channel number). The addressing of messages is performed using PSID (Provider Service Identifier), a unique ID identifying the source/target application. At receiving side, if the PSID value read in a received message represents a service not in the interests of the WSM recipient, then the corresponding message is not processed. The flow of WSMP message exchange can be summarized in 3 points:

1. A vehicular application generates a WSM, addressing it to the broadcast MAC address.

2. The application, depending on the particular service it is providing, selects the appropriate radio resources and parameters to be used for transmission, and invokes the appropriate request to trigger WSMP to deliver the WSM data to the lower layers.

3. At the receiver side, the message traverses the communication stack and, if the PSID is of interest, it is delivered to the appropriate application. At this point, the application can continue the message exchange using either unicast or broadcast MAC addresses.

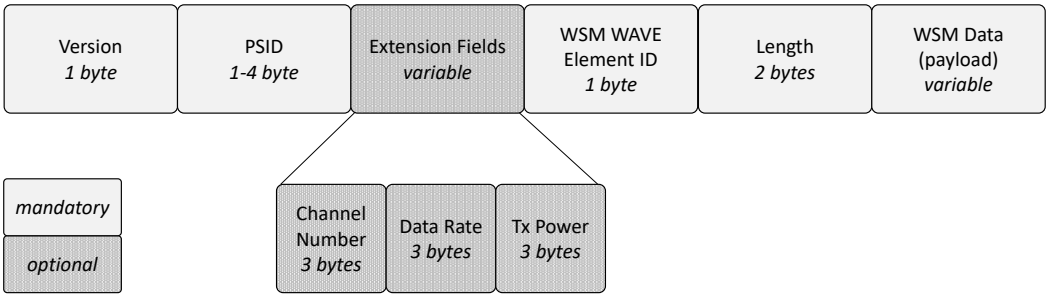


Figure 2.3: Structure of WSM header [57].

The structure of a WSM header can be seen in Figure 2.3. The protocol overhead is between 5 bytes and 20 bytes [57], with optional and mandatory fields which are described in the next paragraph.

1. Version: contains the WSMP version (4 bit) and 4 reserved bits. It can be used to assess the compatibility of received WSMs.
2. PSID: similarly to the concept of UDP or TCP ports, the PSID identifies the service (or the application) the payload is associated with.
3. Extension Field: defines the channel to be used, the data rate at which transmitting the packet and the transmission power.
4. WSM WAVE Element ID: marks the end of the optional Extension Fields and indicates the WSM Data format.
5. Length: contains the size of the payload in bytes (ranging from 0 to 4095 bytes).

Beside WSMP, WAVE supports IP version 6. IPv6 was selected over IPv4 for the known scalability issues that IPv4 has in case of large-scale addressing system with millions of devices. Normally, IPv6 is adopted when the overlying application needs IP-related features, such as routing of the packets over the Internet hosts. The WAVE standard does not give clear indications on the transport nor on the higher layer protocols that should be deployed when using IPv6. The two protocols normally adopted at transport layer are TCP or UDP, with the latter advantaged for its resilience and suitability to VANETs' highly volatile scenarios.

For what concerns the management plane at the Transport and Network layers, IEEE 1609.3 defines the WAVE Management Entity (WME). WME is in charge of facilitating the management functions at this layer: it accepts service requests from the high layer entities, and provides access to specific channels. Thus, WME is in charge of managing the channel access for the higher layer, meaning that it has to tune the radio transceivers to specific channels during different time slots, by directly interacting with the MAC layer management entity (MLME).

Another important function of WSE is the transmission and reception management of WAVE Service Advertisement (WSA) messages (intended to carry information about one or more vehicular services offered in an area) through WSMP. Example of services that may use WSA messages to advertise their presence to the neighbors are tolling, navigation, gaming, restaurant information etc. [57]. The ITS entity providing these services may be either a RSU or a vehicle, thus the only categorization possible can be made by looking at the possible roles in the WSA functionality. It is possible to distinguish between:

- Provider, assumed by a device transmitting WSAs, therefore indicating the availability for data exchange on one or more channels.
- User, assumed by a device monitoring for received WSAs, and potentially interested in starting a data exchange.

A WAVE device may assume one, both or neither role [49].

Data-link layers

In these layers several functionalities are implemented, starting from the LLC sublayer. In LLC, the EtherType (a 2-octet field of the header) is used to determine the overlying Network layer protocol (in WAVE, either WSMP or IPv6). LLC is partly defined in IEEE 802.2, while its peculiar use in WAVE is defined in IEEE 1609.3 [49]. The hexadecimal values indicating WSPM and IPv6 are respectively 0x86DC and 0x88DD. In transmission, the LLC sublayer sets the "Type" field of the LLC header to one of the two possible values. Similarly, in reception LLC reads the "Type" field and determines whether the payload should be dispatched to the IPv6 stack or to WSMP.

As already mentioned in previous sections, WAVE utilizes the 802.11p standard for its channel access layers (MAC and Physical). However, in WAVE the MAC sublayer is extended by IEEE 1609.4 [50], defining the channel coordination in support for multi-channel operations. The general purpose of a MAC sublayer is to define the rules for accessing the underlying transmission medium, so that it can be fairly shared among the upper layers [57]. In WAVE, devices communicate in the so called OCB (Outside the Context of an 802.11 Basic Service Set) mode, a transmission mode allowing direct communication. In OCB mode, as opposed to the normal 802.11 operations where all data frames are sent between STAs (802.11 stations) belonging to the same BSS, the communication is limited to STAs not belonging to any BSS. This allows the elimination of any overhead caused by the intermediary phases of data transmission, where the STAs communicates with the Access Point and vice versa, enabling a low latency and low overhead communication that is suitable for the DSRC environment.

The channel coordination feature defined in IEEE 1609.4 enables data transmission and reception involving multiple devices on multiple channels. The channel switching feature is however optional: a device is permitted to communicate through the same channel all the time. The main goal of IEEE 1609.4 standard is to find a way in

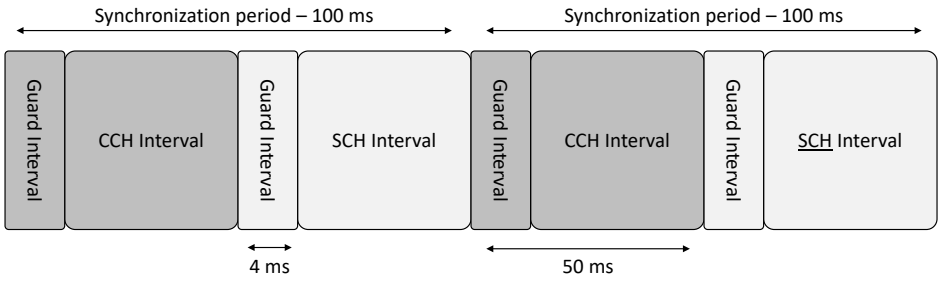


Figure 2.4: Time division in WAVE, allowing fast and efficient channel switching [57].

which multiple devices, transmitting in multiple channels, can efficiently talk to each other. The solution involves the *5.9 GHz channel partition* and the *time division* (depicted in Figure 2.4). As described in Section 2.1.1, the channel in which WAVE operates is logically divided into a Control Channel (CCH) and several Service Channels (SCH). The CCH is used as "rendezvous" channel, where all the devices tune periodically, exchanging management and control information. All the other SCHs are used for the applications-related data. The time division concept, instead, assumes that all the devices are synchronized through GPS. Thanks to this synchronization, the time is divided into alternating CCH and SCH intervals, allowing periodic rendevouz and efficient data transmission. As an example, a device with a single Physical layer is allowed to transmit priority data on a CCH during a time slot, and immediately switch to the transmission of higher layer traffic on a SCH the next time slot.

Another service standardized in IEEE 1609.4 is the channel routing for data and management frames, that allows the prioritization of data based on the combination of upper layer priority information and message type. Furthermore, this functionality permits the correct setting of the transmission power based on the message type to be transmitted.

All the mechanisms here described are coordinated by the MLME Extension present in IEEE 1609.4, that allows systems with multiple radios to effectively switch among the available channels, keeping them synchronized with the other devices. Moreover, the extended MLME maintains the Management Information Base (MIB), a register containing configuration and status information about the MAC and Physical resources available for transmission.

Access layers

The access layers of WAVE (MAC and Physical), are described in IEEE 802.11p [51] and reported in the dedicated Section 2.2.3.

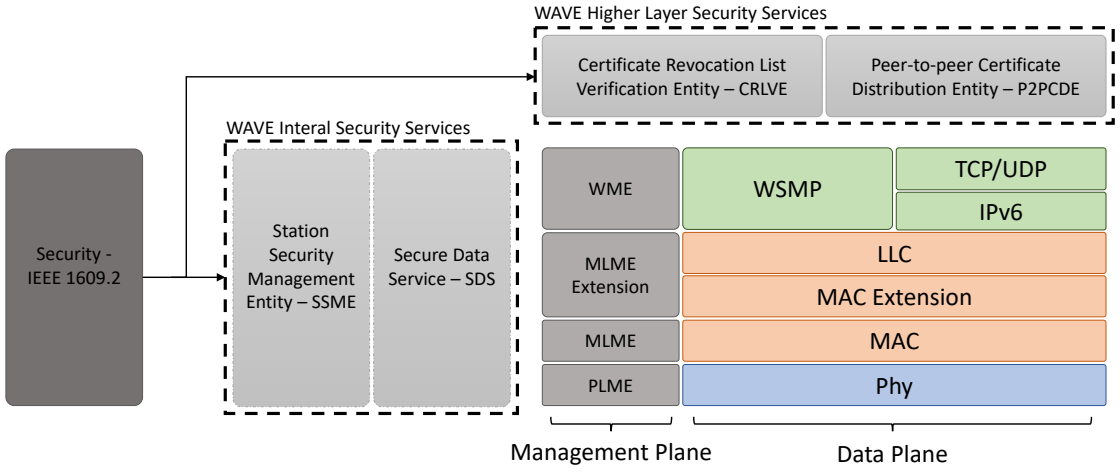


Figure 2.5: WAVE Security Services.

Security services

One of the most challenging aspect for vehicular network is surely represented by security. In WAVE, the IEEE 1609.2 standard is the one taking care of it, by defining standard mechanisms for authenticating and encrypting messages, especially for WSMs and WSAs. The security services introduced by IEEE 1609.2 consist of WAVE Internal Security Services and WAVE Higher Layer Security Services [48], as shown in Figure 2.5.

The two Internal Security Services are:

- Secure Data Service (SDS): in charge of turning normal Protocol Data Units (PDUs) into Secured Protocol Data Units (SPDUs) (and viceversa, in reception), and to manage their transmission among the various entities. The additional data overhead introduced by this service, when the PDU is turned into SPDU, is called Security Envelope. The types of SPDU that may be generated by this service are *unsecured*, *signed* and *encrypted*. An *unsecured* SPDU, as the name suggests, does not introduce any signature or cryptography mechanism to improve security. A *signed* SPDU provides instead digital signature security features, including:

  a) Authenticity - to be sure about the sender identity.

  b) Authorization - to be sure that the sender as the rights to claim the required service.

  c) Integrity - to be sure that the data has not been altered during the transmission.

  d) Non-repudiation - the ability to be able to proof authenticity, authorization, and integrity to third party authorities when requested.

An *encrypted* SPDU provides instead confidentiality, i.e., the assurance that the content of the message will be read only by the intended recipient.

SDS may implement multiple layers of cryptography on its SPDUs, so to ensure all the properties mentioned above.

- Station Security Management Entity (SSME): in charge of managing all the information about certificates. SSME manages both the information of the certificates for which the private key is internally stored in the SDS, and the information of the certificates held by other peer entities and external Certificate Authorities (CA).

The WAVE Higher Layer Security Services are:

- Certificate Revocation List Verification Entity (CRLVE): in charge of validating the incoming CRLs, and passing the related information to SSME. Therefore, when a certificate is revoked by the CA the information will pass through CRLVE to SSME. From this point on, every packet signed with the revoked signature is considered invalid and discarded.

- Peer-to-peer Certificate Distribution Entity (P2PCDE): in charge of managing the cooperation with other secured peer entities, for the distribution of certificates needed for SPDU verification. This functionality is triggered when a device receives a SPDU with an unknown certificate, and generates a request to peer devices to provide the necessary information to complete the chain. The updated list of certificates is sent through PDUs. The P2PCDE service is also designed to mitigate the risk of channel flooding, by limiting the number of responses to a request.

The security mechanisms introduced by IEEE 1609.2 have been designed to meet the DSRC capacity requirements, and to ensure a secure connection even in unacknowledged mode. The drawback of this efficient implementation is the impossibility of having one-to-one secure sessions, which often involves handshake procedures that are impossible to be implemented in vehicular scenarios.

2.2.2 ITS-G5

ITS-G5 is the set of communication protocols for the vehicular environment that ETSI thought for the European market. It has been developed since 2007, and its architecture is very similar to that of WAVE, with many technologies that are identical or derived by the American set of standards. The ITS-G5 is defined, in its different layers, by a number of ENs (European Norms) and TSs (Technical Specifications) which define higher and middle layers. As access technology, ITS-G5 relies officially on the IEEE 802.11p standard. However, with EN 303 613 [30] ETSI showed that the access layers limitations imposed by the previous standards may be overcome, paving the way for

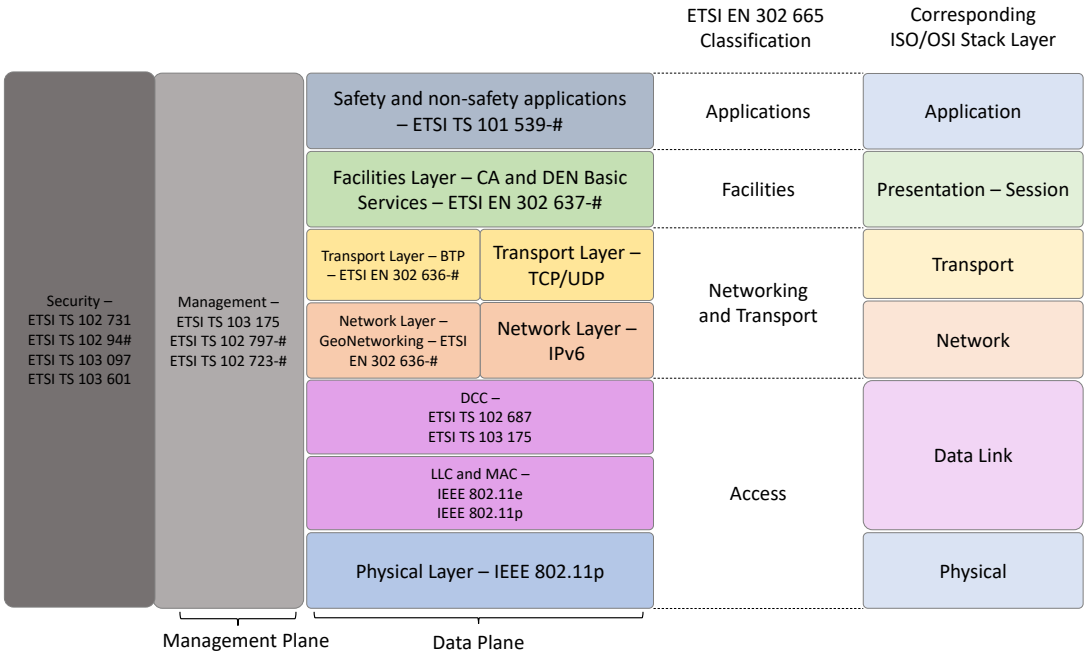


Figure 2.6: Layered structure of the ITS-G5 protocol stack. ETSI EN 302 665 [29] divides the ITS device architecture into 4 layers: applications, facilities, networking and transport and access. The "#" symbol in the image represents multiple documents with consecutive enumeration.

the interconnection among the higher layers previously standardized and the access layers defined in cellular-based technologies (such as LTE-V2X).

The general architecture of an ITS station is introduced in EN 302 665 [29], and it subdivides the communication stack into several layers:

- Applications layer, with some technical specifications on main safety-related applications (TS 101 539-1/2/3 [33, 34, 35]).
- Facilities layer, which covers the ISO/OSI layer 6 and 5 (Presentation and Session), defining the set of Basic Services and the V2X message format CAM (Cooperative Awareness Message) and DENM (Decentralized Environmental Notification Message) as described in TS 102 637-1 and EN 302 637-2/3 [36, 26, 27].
- Networking and Transport layer, which corresponds to layer 4 and 3 of the ISO/OSI stack. The reference standard is EN 302 636 [21, 22, 23, 24, 25], where the Basic transport Protocol and the Geonetworking layers are defined, and which also allows TCP/UDP on top of IPv6 for non-safety-related communications.
- Access layer, covering layer 2 and 1 of the IOS/OSI stack, where the Decentralized

Congestion Control (DCC) mechanism is defined (TS 102 687 and TS 103 175 [37, 52]), and the MAC and Physical layers are taken from 802.11p.

The resulting architecture can be seen in Figure 2.6.

As in 2.2.1, this Section gives an high level overview of each layer of ITS-G5, without delving into the details of procedures and interfaces. For further information the reader is referred to the standards present in the ETSI online library [20].

Application and Facilities layers

Differently from WAVE, where the standardization starts from the Transport layer, ITS-G5 extends its scope to the higher layers of the stack, and starts with the definition of a basic set of applications dedicated to road safety. In [33, 34, 35], some of the most important applications based on the Cooperative Awareness (CA) and on the Decentralized Environmental Notification (DEN) basic services are presented and described. In particular, [33] introduces the Road Hazard Signaling (RHS) application thought for the increased awareness of both ITS stations and drivers, through the dissemination of CAMs and DENMs signaling the presence of hazardous situations. [34] includes instead the guidelines for an Intersection Collision Risk Warning Application (ICRWA) still based on messages entirely exchanged through radio interfaces. The considered use cases includes the *turning collision warning* also called *left turn assistant*, the *merging collision risk warning*, a mode for vehicles with missing radio connectivity and several others, still related with intersections. Similarly, [35] describes another class of applications, the Longitudinal Collision Risk Warning Application (LCRWA), and the way in which the players using such application should communicate among themselves.

Beside the basic set of applications, ITS-G5 defines in the higher layers of the stack the so-called Facilities layer. The aim of this layer is to provide support to ITS applications which can share generic functions and data according to their respective operational requirements [29]. The standard introduces an additional breakdown of this part of the stack, by defining 3 different sublayers:

1. Application Support, providing functionalities to support the overlying applications. CA, DEN and SAM (Service Announcement Message - correspondent of WAVE's WSA) basic services belong to this layer.

2. Information Support, that stores and maintains the data for Local Dynamic Map (LDM), used in C-ITS for cooperative perception.

3. Communication Support, that provides the tools for lower layer independence, allowing the deployment of different access technologies (like LTE or C-V2X) and different network protocols (non-IP based).

Due to the importance that CAMs and DENMs have in the topics covered by this thesis, the two entities managing their creation will be introduced in the next paragraphs.

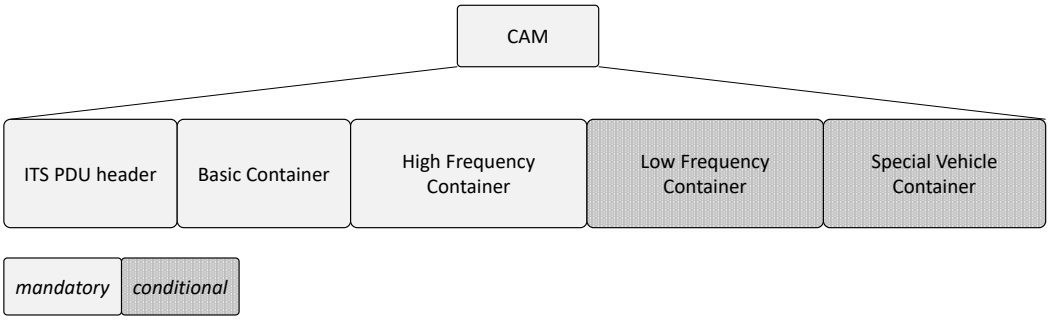


Figure 2.7: Format of a CAM message, as specified in [26]. Some of the fields are mandatory, others depend on the originating ITS-S type and dynamics.

The CA basic service, as defined in [26], is the facilities layer entity that operates the CAM protocol. It provides two important services: the CAMs transmission and the CAMs reception. For these purposes, this layer should integrate the following functionalities:

- CAM encoding, following the ASN.1 specification included in the standard.
- CAM decoding, to correctly parse the information sent by other stations.
- CAM transmission management, including the activation and termination of CAM transmission operations and the determination of the CAM frequency.
- CAM reception management, to trigger the CAM decoding, and eventually dispatching the CAM information to the application requesting it.

CAMs are always transmitted in the CCH in a single-hop fashion (i.e., it is not possible for a third-party entity to re-transmit a CAM message). The CAMs frequency is dependent on the generating vehicle dynamics, and it is thought to mitigate the channel congestion. The CAM inter-arrival time is set between 100 ms and 1 second. The general idea is to increase the frequency of fast-moving or fast-turning vehicles, and at the same time to reduce the dissemination frequency of those vehicles which are still or slowly proceeding. In this way the cooperative awareness granularity is preserved even when vehicles are moving fast (e.g., in highways), and the general position perception is less scattered.

The standard also defines the CAM format, which is represented in Figure 2.7. Beside the ITS PDU header, which includes information to uniquely identify the generating ITS station, 4 different containers (2 mandatory and 2 conditionally present) are defined.

1. Basic Container, it provides the basic information of the originating ITS station such as type and last known reference position.
2. High Frequency Container, that contains all the fast-changing status information of the generating ITS station such as speed, position, heading, acceleration etc.

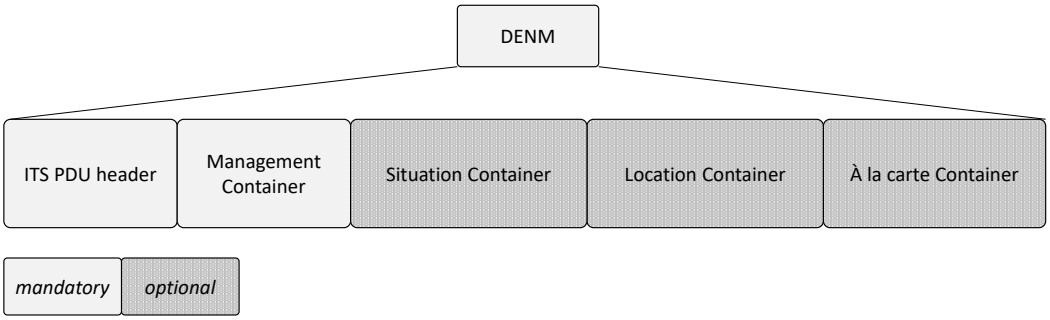


Figure 2.8: Format of a DENM message, as specified in [27]. The Management Container is mandatory, while all the others are optional.

3. Low Frequency Container, including static or slow-changing vehicle data, like the vehicle's role or the exterior light status. Due to the low volatility of such information, this container is not always included.

4. Special Vehicle Container, a conditional set of information included only by determined classes of vehicle (special transport, emergency and rescue vehicles etc.).

All the fields present in the CAM messages are defined in a special common data dictionary [38].

The DEN basic service, as defined in [27], is an application support module provided by the Facilities layer that operates the DENM protocol. Differently from the CA basic service, which periodically generates CAMs without the interaction of the higher layers, the DEN basic service is triggered by the applications and provides an asynchronous tool to notify information to other ITS stations. In transmission, the DEN basic service should provide the primitives to generate, update or terminate the DENM. In reception, it should dispatch the received information to the higher layers, when necessary. The information carried in a DENM are related to events that have potential impact on road safety or on traffic condition. For the nature of these events, it may happen that DENM are not associated with the generating ITS station, but with the event itself. For these reasons, it has been necessary to define multiple types of DENM: *new DENM*, when the event is new; *update DENM*, used to refresh the information of a certain event; *cancellation DENM*, to signal the termination of the situation that triggered the initial DENM.

Differently from CAMs, a DENM can be forwarded by a third-party entity. The DENM format can be seen in Figure 2.8. Beside the ITS PDU, already introduced for CAMs messages, it is possible to identify the following Containers:

1. Management Container, providing information related to the DENM management and DENM protocol, such as the action ID, the detection time, the event position etc. This is the only mandatory Container present in a DENM.

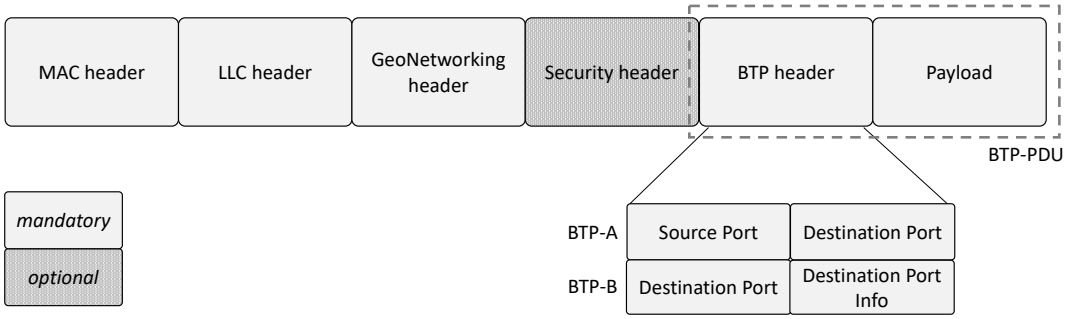


Figure 2.9: Structure of a BTP-PDU encapsulated into the full ITS-G5 frame [25].

2. Situation Container, which includes information describing the detected event through special *causeCode*, and through qualitative scale.

3. Location Container, describing in detail the location of the detected event.

4. À la carte container, which contains additional information not included in other containers, such as the event lane position, the external temperature, road works presence etc.

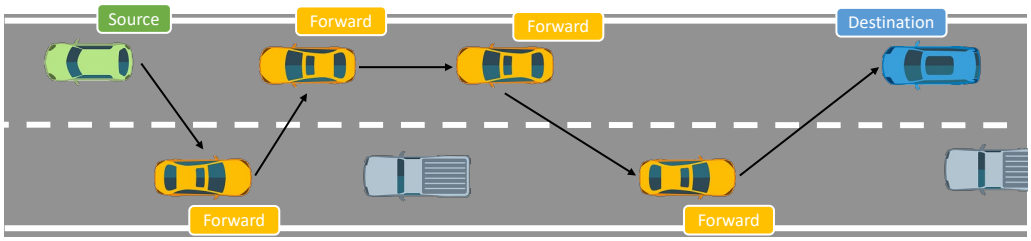
As for CAMs, all the field present in DENMs are defined and described in the common data dictionary [38].

Transport and Network layers

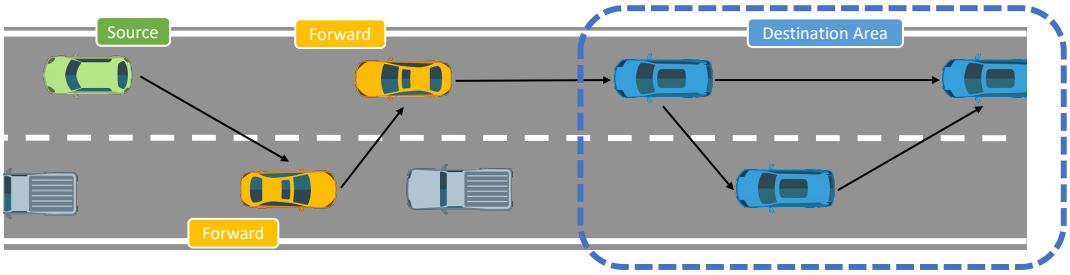
The Network and Transport layers in ITS-G5 are mainly standardized in EN 302 636-1/5 [21, 22, 23, 24, 25]. ITS-G5 uses Basic Transport Protocol (BTP) at Transport layer, and GeoNetworking (GN) at the Network layer. In case it is required, it is possible for the services to rely on the classic TCP/UDP over IPv6 or even TCP/UDP IPv6 over GN.

BTP is a Transport layer protocol that, similarly to UDP, provides a connectionless end-to-end transport service for ITS ad-hoc networks. The delivery of PDUs among BTP entities is not guaranteed, thus no reception acknoledgement nor reordering functionalities are provided. It allows the DEN and CA basic services to access the lower layer of the stack, i.e., GN.

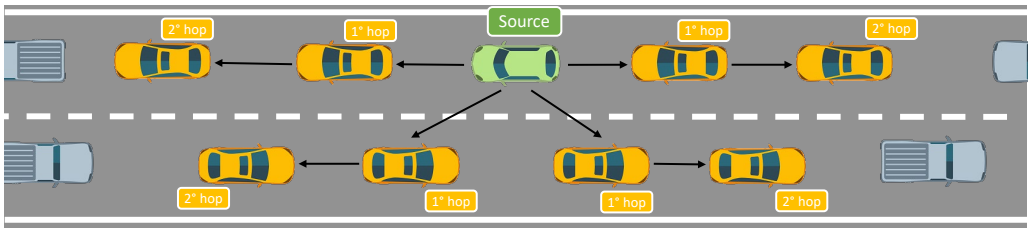
The BTP-PDU is shown in Figure 2.9, when encapsulated in a frame comprising the lower layer headers (MAC, LLC and GN). The BTP header can be of two types: BTP-A and BTP-B, depending on the type of service. If there is the need for an interactive session (i.e., if the intended recipient is required to reply), then the BTP-A header should be used, including source and destination ports. If instead the session is non-interactive the BTP-B header should be used, including only destination port and information. CA and DEN basic services make use of BTP-B header, since the receiver is not needed to reply to CAM and DENM messages.



(a) GeoUnicast addressing scheme, to send unicast packets.



(b) GeoBroadcast addressing scheme, to send packets to all the nodes in a destination area.



(c) Topologically-scoped Broadcast, to broadcast packets to all the n-hop neighbors. In this example, a 2-hop forwarding is implemented.

Figure 2.10: GN addressing schemes for unicast and broadcast transmissions.

Due to the volatile nature of VANETs, the network protocol adopted in ITS-G5 should be able to provide communication even without the help of a coordinating entity. GN is a family of protocols (initially proposed for MANETs, then adapted to VANETs) that utilizes geographical positions to route the information and, therefore, to disseminate data packets. In ITS-G5, GN provides a connectionless fully distributed protocol that can bear with rapid modifications of the network topology, and that can work with intermittent or even without infrastructure access. GN has been thought to supports the periodic transmission of safety messages, to provide multi-hop forwarding functionalities and normal unicast transmission for classic IP-based services. Unlike conventional networks, where addressing and routing are purely based on addresses represented by numbers, string or hexadecimals, in GN every node has a partial view of the surrounding area and every packet sent carries the geographical address of the intended destination (unicast or broadcast). Whenever a node receives a packet, it reads

the geographical information and decides whether it is necessary to forward the packet or not. By using GN it is possible, for example, to forward a packet to a specific area (e.g., an area affected by a flooding), and alert only the vehicles present in that specific location.

GN mainly provides three types of addressing schemes, as shown in Figure 2.10:

1. GeoUnicast, to send unicast packets. When a vehicle wants to send a unicast packet, it determines the destination's position and includes it in the GN header. Thus, the packet is sent and continuously forwarded until it reaches the intended destination.

2. GeoBroadcast, to broadcast a packet in a certain area. As in GeoUnicast, the generating node should include the destination area in the GN header, and the packet is forwarded until it reaches the intended area.

3. Topologically-scoped broadcast, to broadcast a packet from the source to all nodes in the n-hop neighborhood. CA basic service, with its CAM protocol, is an example of topologically-scoped broadcast (more precisely, of a single-hop broadcast since the CAMs cannot be re-forwarded).

Access layers

One of the core functionalities of ITS-G5 is the so called Decentralized Congestion Control. DCC is a mechanism that prevents C-ITS stations to flood the channel and that guarantees a stable and reliable service even in highly dense scenarios. DCC works by tuning the MAC and Physical layer parameters depending on channel quality indicators such as the Channel Busy Ratio (CBR). Differently from WAVE, in ITS-G5 there is no alternating access scheme.

In ITS-G5 the MAC-layer reference protocol is 802.11p, that introduces an important feature allowing a fine grained Quality of Service selection for the overlying applications, namely the Enhanced Distributed Channel Access (EDCA) protocol. In particular, with EDCA data are assigned to 4 different MAC queues, depending on the Access Category (AC) and that represent different levels of priority. The 4 ACs are Background (BK), Best Effort (BE), Video (VI) and Voice (VO).

Each queue that wants to transmit data, should contend the access to the physical layer. To do so, the protocol requires the queue to wait for the channel to be idle for at least AIFS (Arbitration Inter-Frame Spacing), a time period that depends on the AC. In case of virtual collisions, the queue should backoff for a period of time still depending on its AC (the higher the AC, the lower the backoff), and that is extracted as a random variable in an interval called Contention Window (CW). More details on EDCA can be found in Section 2.2.3.

Thus, the channel access scheme will be directly controlled by DCC, which will set the parameters of the different ACs, but also other parameters such as transmission power, packets intervals, and sensitivity of the radio, by taking ad-hoc decisions.

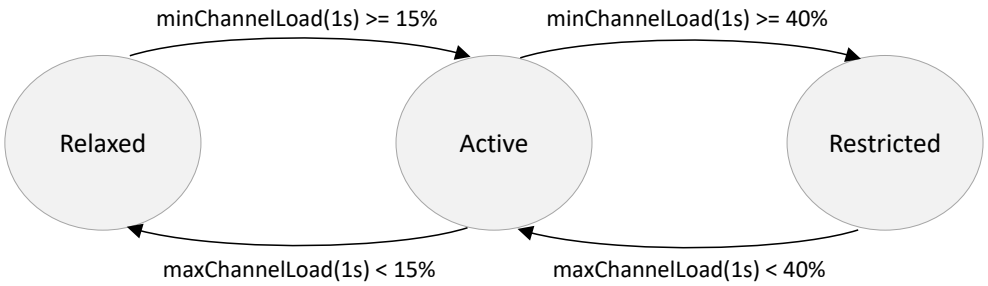


Figure 2.11: State machine of DCC based on CBR sensing.

The DCC algorithms may implement one or several among the following techniques to reduce the channel load:

1. Transmit Power Control (TPC), in which the output power is altered to adjust the current channel load. This allows, for example, to reduce the interference range achieved by an ITS station transmitting V2X packets.
2. Transmit Rate Control (TRC), in which the transmitting ITS station regulates the time between consecutive packets, thereby reducing the transmission frequency.
3. Transmit Datarate Control (TDC), in which the transmission datarate is changed, to allow for example to reduce the time in which the station is transmitting (by increasing the datarate).
4. DCC Sensitivity Control (DSC), in which the channel is sensed to determine (based on the list of sensitivity present in [37]) whether the transmitter is clear to send or not.
5. Transmit Access Control (TAC), in which, in case of high channel occupancy, the transmission policies become more restrictive for ITS stations transmitting many packets.

By monitoring the CBR, ITS-G5 DCC classifies the channel into 3 different states: relaxed, active or restricted. The resulting state machine can be seen in Figure 2.11, where each state has its associated parameters (EDCA, tranmit power, carrier sense threshold, modulating and coding scheme) and where the transitions among states depend on the minimum or maximum channel load computed in predefined time intervals.

Below DCC ITS-G5 implements 802.11p, described in Section 2.2.3. Differently from WAVE, which does not admit other access technologies rather than 802.11p, ITS-G5 adopts a more relaxed strategy. In fact, during the last few years there have been several amendments released by ETSI to integrate 3GPP's solutions in the ITS-G5 stack, as detailed in ETSI EN 303 613 [30] and related documents.

Security services

The approach taken by ETSI to define the security profile of ITS-G5 is different with respect to that taken by IEEE for WAVE: ETSI defines, in [53], a range of security services which should be supported by an ITS entity to guarantee secure connection between itself and other entities, that are then detailed in their interfaces and protocol in other ad-hoc documents. Each of the services defined operate within multiple layers of the stack, and is coordinated through a security management entity. A non exhaustive list of services defined is reported below:

- Security associations management. This functionality gives the opportunity to establish a secure communication between ITS stations, so that they can exchange secured V2X packet.
- Single message services. This set of services are in charge of securing the transmission or reception of a single message (e.g., CAM and DENM) by providing functions to authorize, validate, encrypt and decrypt messages.
- Integrity services. In charge of computing and checking messages' checksum.
- Enrollment and Authorization. These services should guarantee the ITS entity to request credentials to Enrollment Authorities and authorization tickets to Authorization Authorities. The credentials are needed to be trusted by other entities, while authorization tickets are needed to access the various ITS services.

2.2.3 802.11p

IEEE 802.11p is the standard originally adopted for vehicular communications both in WAVE and ITS-G5. Its MAC and Physical layers are largely based on two other 802.11 amendments: 802.11a, from which it derived the 5 GHz band and the OFDM (Orthogonal Frequency-Division Multiplexing), and 802.11e, from which it inherited some important Quality of Service (QoS) features. At-a-glance, 802.11p defines a method to exchange data packets that does not requires the STAs to belong to any BSS. The so called OCB mode introduced in 802.11p enables a distributed access scheme that is suitable for the volatile environment generally present in VANETs. Although 802.11p is an adjustment of other Wi-Fi-based protocols, due to the different nature of the problems addressed it can be considered as a standalone architecture.

MAC layer

The major revisions and adjustments that allow 802.11p to be suited for the vehicular environment, are to be found at MAC layer. Rapid changes in the network topology, coupled with the need of an efficient and reliable connection, forced IEEE to reconfigure the channel access technique. In normal 802.11 network, STAs communicate in

AC	CW_{min}	CW_{max}	AIFSN
AC_BK	CW_{min}	CW_{max}	9
AC_BE	CW_{min}	CW_{max}	6
AC_VI	$(CW_{min}+1)/2 - 1$	CW_{min}	3
AC_VO	$(CW_{min}+1)/4 - 1$	$(CW_{min}+1)/2 - 1$	2

Table 2.1: Default EDCA parameter, as specified in [51].

Parameter	Value
SlotTime	13 µs
SIFSTime	32 µs
CW_{min}	15
CW_{max}	1023

Table 2.2: Default values for *SlotTime*, *SIFSTime*, CW_{min} and CW_{max} as specified in [51].

the context of the BSS of belonging. All the communications are directed to and from the Access Point, which is identified and reached through its MAC address (called, in 802.11, BSSID). In the "p" amendment it is possible, regardless the BSS, to use BSSID wildcards, and all the communications happens directly among STAs, without the need for a coordinating Access Point. This way of accessing the channel is called OCB mode (Outside the Context of a BSS).

Another important feature enabled at the MAC layer of 802.11p (and partially introduced in previous sections) is EDCA, that allows the prioritization of data according to 4 different access categories (AC):

1. AC_BK: Background;
2. AC_BE: Best Effort;
3. AC_VI: Video;
4. AC_VO: Voice;

802.11p implements 4 transmit queues, each corresponding to a different AC. Each queue should contend the access to the channel by waiting for it to be idle for AIFSN (AIFS Number) periods; AIFSN is calculated as:

$$AIFSN[AC] = AIFSN[N] \times SlotTime + SIFSTime \tag{2.1}$$

Each AC is associated to a different AIFSN, the higher the priority, the lower AIFS. In case of virtual collision the queue should implement a backoff strategy, by extracting a random value between CW_{min} and CW_{max} (Contention Window min or max). The standard set the default values for CW and AIFS, as shown in Table 2.1. These strategies, combined together, allow the system to respect the queues' priority, and at the same

Parameter	Value
Frequency band	5.9 GHz
Channel size	10 MHz
Number of used data subcarriers	52
Number of used pilot subcarriers	4
OFDM symbol period	8 µs
Cyclic prefix	1.6 µs
FFT symbol period	6.4 µs
MCS	BPSK, QPSK, 16QAM, 64QAM
Available data rate	3, 4.5, 6, 9, 12, 18, 24, 27 Mbps

Table 2.3: Physical layer parameters for 802.11p, as standardized in [51].

time (through the CW extraction randomness) prevent the higher priority queues to monopolize the channel. This 4-layer priorities scheme is enhanced by the so called Alternate EDCA, which introduces 2 more layers of priority (namely, Video alternate and Voice alternate) offering a more granular differentiation among ACs.

The standard values reported in [51] for CW_{min}, CW_{max}, *SlotTime*, *SIFSTime* that allows the computation of the parameter in Table 2.1, can be seen in Table 2.2.

Physical layer

The Physical layer in 802.11p derives from that of 802.11a, with some modifications needed to tackle the peculiar characteristics of VANETs. In particular, to counteract the effects of multi-path fading and Doppler effect (which may cause inter symbol interference), the channel used are reduced from 20 MHz to 10 MHz. OFDM, in 802.11p, is based on 64 orthogonal subcarriers and it supports datarates from 3 Mb/s to 27 Mb/s (depending on the Modulation and Coding Scheme - MCS). The MCS determines also the minimum sensitivity required to correctly demodulate the received signal. The possible modulations are BPSK, QPSK, 16QAM and 64QAM. The main characteristics of the 802.11p Physical layer are resumed in Table 2.3.

2.2.4 Toward the next generation of 802.11 vehicular protocols

The fact that IEEE 802.11p has more than 10 years of history has its pros and cons. The pros are mainly linked to the wide testing and proof of concepts that have been made during these years. The cons are instead relative to the fast obsolescence to which this type of technologies incur in a relatively short time. As already introduced, 802.11p derived Physical and MAC layer from 802.11a and 802.11e. In the meantime, the 802.11 family was added with new protocols (802.11n/ac/ax) that extremely improved the performance at the access layers. For that reason, in 2018 IEEE formed a new Working Group which primary objective is to create the new generation of 802.11-based V2X

protocol, namely 802.11bd. The standardization is still ongoing and the main features required for the new protocol are summarized as follows [45, 72]:

- Increase the throughput, by providing at least a mode that doubles the 802.11p throughput.
- Increase the communication range, by providing at least a mode that doubles the 802.11p communication range.
- Re-design of OFDM numerology, by optimizing the tone spacing and guard duration interval, to efficiently meet the vehicular mobility requirements.
- Multiple Input Multiple Output (MIMO) and Low Density Parity Check (LDPC) support.
- Implementation of midambles, i.e., control sequences in the middle of the packet to improve the contrast to Doppler effect.
- Retro-compatibility with legacy 802.11p, with guaranteed fairness in the channel access.

2.3 Cellular-based protocols

Cellular-based solutions for V2X communications are getting more and more attention from the industry and academic world. Although standardization started almost ten years later with respect to IEEE 802.11p, the V2X solution proposed by 3GPP is promising under multiple aspects. Improved radio range, possibility to use Uu interface to connect to the cellular infrastructure, higher scalability, and broader possibilities in terms of involved players, are among the most important advantages introduced by cellular V2X (C-V2X). The approach is different with respect to WAVE and ITS-G5: the scope of 3GPP is to integrate the V2X communication inside the cellular infrastructure; for this reason, C-V2X does not cover any of the higher layers of the communication stack, and focuses only on the channel access protocols (Data Link and Physical layer).

3GPP embedded the concept of V2X for cellular networks in Release 14 [1], where it introduced two modes of operation:

a) through PC5 interface, which supports one-to-many communications and enables direct connection among UEs (User Equipments);

b) through Uu interface, which enables the communication with the network infrastructure, namely, with the eNB (E-UTRAN NodeB), which plays the role of base station in LTE network.

Communication through PC5 interface is supported over the sidelink, and can be either coordinated by a eNB (under network coverage, also called transmission mode 3), or in

a complete independent way (out of network coverage, also called transmission mode 4). Therefore, through the PC5 interface it is possible to enable V2V-based applications. Communication through Uu is only supported under network coverage, with the UE receiving V2X messages (unicast or broadcast) through the downlink, and sending V2X messages via uplink. Through the Uu interface, it is possible to have access to the solid edge computing capabilities, to cloud applications, and in general to V2N-based services.

In Release 14, 3GPP mostly provides data transport for the messages used by the basic road safety services already introduced in previous sections (CAMs, DENMs, BSMs etc.) and specifies the reference architecture for LTE-based V2X systems.

Release 15 [2] provides instead the service requirements to enhance the support for a wider set of V2X scenarios, like vehicle platooning, advanced driving, extended sensor and remote driving. Among the most important functionalities introduced by Release 15, there is the support for Carrier Aggregation (CA) for transmission mode 4, the support for 64-QAM modulation, improved message generation frequency, and other improvements including the initial integration of V2X services in 5G networks.

Finally, Release 16 [3] (that, at the time of writing, is at its draft stage) introduces requirements related to vehicle QoS support, allowing the V2X applications to be timely notified of expected change in QoS (predictive QoS), and specifies the advanced V2X services enabled by 5G networks.

In the next sections, the integration of V2X services in 4G and 5G networks are presented.

2.3.1 LTE-V2X

The first definition of LTE-V2X appeared in Release 14, as a derivation of Release 12's Device-to-Device communication (D2D). LTE-V2X access layer consists of NAS (Non-Access Stratum), RRC (Radio Resource Control), PDCP (Packet Data Convergence Protocol), RLC (Radio Link Control), MAC and Physical layers. This section will mainly focus on MAC and Physical layer.

The channel where direct messages are exchanged in LTE-V2X is called sidelink (as opposed to uplink and downlink). It can be provided in any LTE band within the ITS spectrum, and it is the medium where the nodes communicate with or without the coordination of the eNB.

Sidelink relies on single carrier frequency division multiple access (SC-FDMA), the same scheme used in the LTE uplink, and supports 10 and 20 MHz-wide channels. Subcarrier spacing is 15 kHz, and grouped into 180 kHz-wide blocks (12 subcarriers each block). In time domain, there is a division into subframe of 1 ms (called transmission time intervals TTIs). Each TTI is composed by 14 symbols: 9 data symbols, 4 demodulation reference signals (DMRS) symbols and one empty symbol for transmission/reception switching. This configuration allows LTE-V2X to tackle the Doppler effect and to cope with the problems carried by working in the 5.9 GHz spectrum. LTE-V2X supports

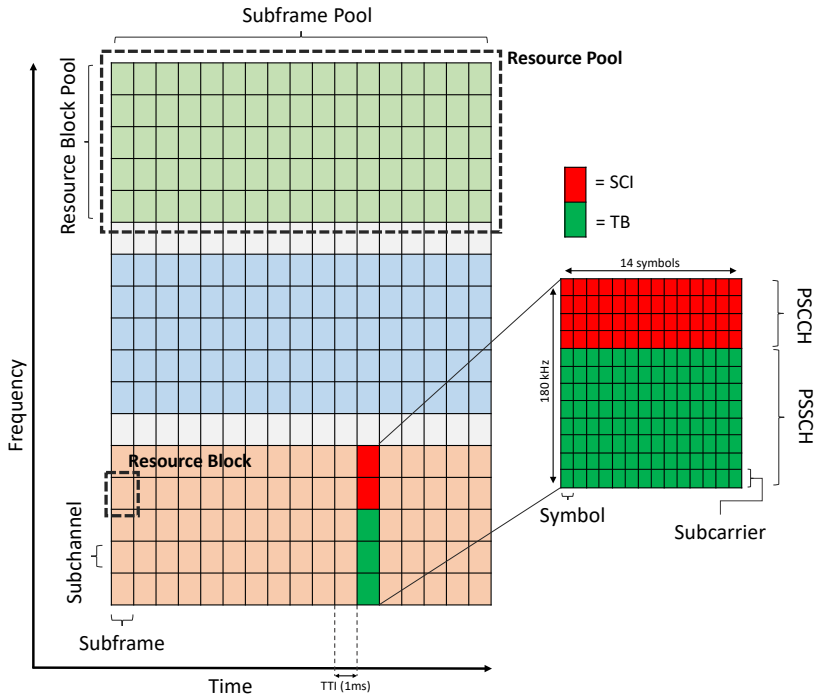


Figure 2.12: Time and frequency division in C-V2X sidelink.

several combinations of MCSs, based on QPSK and 16-QAM.

The minimum resource allocation slot, in sidelink, corresponds to a subchannel in the frequency domain, and to the TTI in the time domain. A normal packet may occupy one or more subchannels in a single TTI. For the sidelink communications 3GPP introduces the concept of Resource Pools (RPs), determining the resources dedicated to each specific UE-to-UE channel. In the frequency domain, the RP is divided into resource block pools, while in time it is divided into subframe pools. Depending on the type of data carried, it is possible to further divide the frequency domain into Physical Sidelink Shared Channel (PSSCH) and Physical Sidelink Control Channel (PSCCH). PSSCH carries data information inside Transport Blocks (TBs) with a TB containing the full packet to be transmitted; the control operations are instead performed by Sidelink Control Information (SCI) messages, describing the transmission property of the next PSSCH, and carried inside the PSCCH. Depending on the suchannelization scheme, it is possible to have adjacent or non-adjacent PSSCH and PSCCH. In Figure 2.12, the adjacent scheme is shown, with all the different time and frequency resources clearly identified.

Due to the time-frequency structure of the physical medium, where resources are orthogonally allocated, the allocation scheme plays a crucial role in providing an efficient service. Depending on the transmission mode, 3GPP envisioned different scheduling algorithms. In mode 3, the computational effort is outsourced to network controllers; in mode 4, it is possible instead to take advantage of the predictability of V2X messages

(especially of those associated with cooperative awareness services), and make use of Semi-Persistent Scheduling (SPS) mechanisms.

Transmission mode 3

In transmission mode 3, where the UEs are under network coverage, the scheduling is performed directly from the network by V2X control functions. This transmission mode requires the vehicles to establish a communication with a central controller, reachable through the Uu interface communicating with the eNB. A big advantage of transmission mode 3 is that having a controller with a wide view of the network status may lead to extremely efficient resources usage. Due to the variety of network configuration and proprietary solutions existing, 3GPP decided to leave transmission mode 3 algorithm specifications to Mobile Network Operators (MNOs). In particular, it is possible for each MNO to choose whether to adopt a dynamic strategy, in which the vehicles request the resources to the eNB, or a static strategy, in which the eNB fixes the resources to be assigned to V2X communications.

Transmission mode 4

In transmission mode 4, where the UEs are outside of the network coverage, each node should autonomously select the resource to use and the scheduling algorithm should be deployed in a distributed environment. 3GPP provided full specification of the SPS algorithm in charge of resource allocation, that can be summarized as follows.

A node that wants to transmit data, randomly selects a resource by choosing among those resources that were free in previous observations (decision that is taken also based on the information present in SCIs). The resource is then kept for a period within 5 and 15 times the packet generation interval. This means that if the UE transmits a CAMs every 100ms, for 5 to 15 times it will use the same subchannels (every 100 TTIs) to transmit its packets. At the end of that time, the same resource is kept for an additional time interval with probability within 0 and 0.8 (decided by the operator). The channel is monitored within 1 second-long windows, and the available resources are determined by comparing the received signal power with specific sensitivity thresholds and by reading at SCI information (advertising future reservations). Among the selected free resources, the 20% less interfered are tagged as available. If the resources selected are not enough, the sensitivity thresholds are re-configured and the process repeated again.

Mode 4 also supports congestion control through algorithms that, however, are not in the scope of 3GPP standards. 3GPP only defines the metrics and the possible mechanisms to be adopted to reduce the channel congestion: as in DCC the reference value is CBR, defined in this case as the amount of subchannels that experienced an average RSSI (Received Signal Strength Indicator) higher than a certain threshold, during the last 100 subframes [70].

2.3.2 5G-V2X

The natural network evolution toward the 5th generation involves also the vehicular world, and 3GPP is currently working to properly define which are the new features that C-V2X should have to be transparently and efficiently integrated into 5G. The first step toward this integration was made in Release 15; after several discussions regarding the role of 5G in V2X services, it has been decided that while the integration with 5G New Radio (NR) will enable a whole new set of advanced applications, the basic safety services (comprising the cooperative awareness) will remain based on Release 14 standard (i.e., on LTE-V2X). The idea is not to provide backward compatibility, but rather to add an optional interface toward NR connecting the UE to Next Generation NodeB (gNB), enabling a link with improved performances.

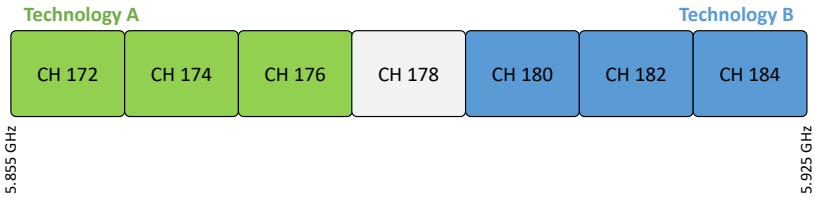
5G-V2X also introduces some modifications to enhance the PC5 interface (communicating through sidelink), listed below:

- Carrier aggregation techniques for both transmission mode 3 and mode 4, supporting up to eight bands.
- Wider set of numerology combinations, and increased subcarrier spacing (from 15 kHz to 30 or 60 kHz).
- MCS extended to 64-QAM.
- Possibility to use frequencies above 6 GHz.
- Establishment of a sidelink feedback channel to improve reliability and decrease latency.
- Use of MIMO antennas, to enable spatial diversity and contrast multipath fading.
- Reduction of the TTI to 0.5 ms.

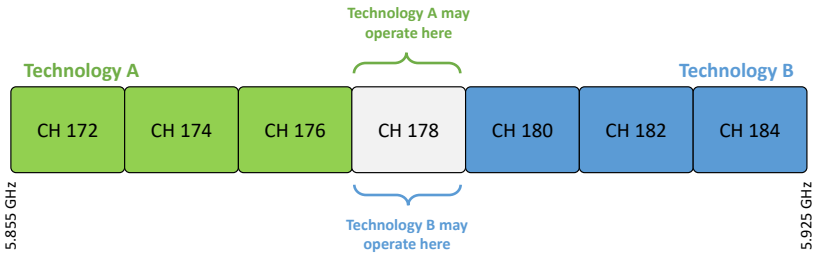
2.4 Interoperability study

The issue of channel co-existence between Wi-Fi and cellular-based V2X technologies is of critical importance for the whole ITS industry. To this aim, an interesting study carried on by 5GAA [5] proposed some spectrum sharing frameworks, reported below. The first solution foresees an a-priori agreement among all the involved stakeholders, to define preferred channel to be used for each technology (called, in this case, Technology A and Technology B). This solution is naturally in line with the 10 MHz subchannelization of the ITS spectrum, and avoids any co-channel interference between the two V2X technologies as shown in Figure 2.13a.

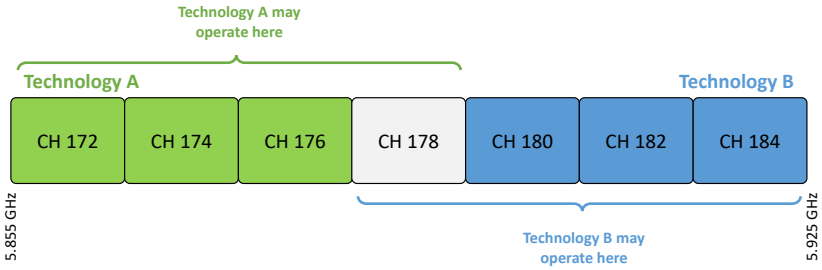
The second solution is instead based on a detect-and-vacate procedure: in this case, the two technologies may operate in their reserved spectrum and, in case they need to transmit in the middle band, they need to monitor the activity in the relevant channel



(a) Sharing of 5.9 GHz via preferred channels.



(b) Sharing of 5.9 GHz, via preferred channels complemented by mutual detect-and vacate in the middle 10 MHz channel.



(c) Sharing of 5.9 GHz, via preferred channels complemented by mutual detect-and vacate extended to the lower and upper 10 MHz channels.

Figure 2.13: Possible solutions for technologies coexistence in the 5.9 GHz band, from [5]

and proceed with the transmission only if the channel is sensed as idle, as shown in Figure 2.13b.

The third option is the natural evolution of the detect-and-vacate solution: the two technologies may normally operate in their reserved spectrum, and in case they need more bandwidth, they can additionally use the middle band, by prior sensing the channel as idle, as shown in Figure 2.13c.

Those strategies, if properly adopted and tuned, lead to a full sharing of the 5.9 GHz spectrum and represents a great benefit for the ITS players willing to converge on an implementable solution for the co-channel interference.

2.5 Related work and comparison studies

The two access technologies presented in this chapter, namely IEEE 802.11p and 3GPP C-V2X, are currently at the center of the discussion in the VANETs world. There have been a number of studies which have discussed the two technologies from various perspectives, by performing performance comparisons in both peer reviewed and white papers.

The vast majority of the works tend to agree on the higher potentiality of the physical layer of C-V2X, which thanks to its channel access scheme can accommodate a higher number of nodes and reach longer distances. There are, however, several scenarios suggesting that 802.11p provides similar, or even better results. Moreover, the large majority of the papers do not take into account all the possible technologies and mechanisms (such as channel switching, dynamic MCS etc.) that a 802.11p device may implement, nor the possible advantages that the QoS introduced in EDCA may play when it comes to service provisioning.

Starting from the moment at which 3GPP defined Release 14, all the subsequent works can be subdivided into those having a direct interest in promoting one or the other technology, and those that do not have any interest in doing so; most of the times, stakeholders' white papers belong to the first group, while peer reviewed papers belongs to the second group.

Within white papers, some of the publications claim that 802.11p is a highly mature technology which has been thoroughly validated with thousands of field tests. Filippi et al. in [40] (a white paper from NXP Semiconductors), assert that 802.11p is the only technology that underwent a remarkably high number of trials, and whose scalability has been extensively tested. Other works, mainly carried on by telecommunication companies or by consortium with strong interests in cellular-based technologies, such as [95, 12, 94], push C-V2X as a more efficient technology, supported by a solid ecosystem and which naturally integrates with 5G. Some of these works carry on direct comparisons among the two technologies, by reporting real measurements showing the advantages that the C-V2X technology brings. Those studies triggered an immediate response from the counterparts, and in [69] the aforementioned results demonstrating the superiority of C-V2X are contested and accused of being biased by several methodological errors.

Fortunately, most of the peer reviewed papers respect the neutrality that is required for a scientific approach to the issue. In general, these kind of analysis are carried on in simulation environment, which has the advantage of drastically reduce the costs of development and testing. The simulation approach is also at the base of some of the works presented in this thesis, such as the MS-VAN3T simulation framework, used to develop and validate the proposed collision avoidance system.

Some of the works present in literature, like [74, 100, 70, 15], highlight the advantages of the cellular based solution in terms of radio range and in connection stability.

However, the study on the performance of the various modulation and coding schemes

(MCS) performed in [15] and [70], pointed out the strong correlation between communication distance and MCS adopted in 802.11p. Some of the results presented in [70] suggest also that the scheme 16QAM-3/4, in 802.11p, provides better results in congested scenarios if compared to C-V2X in transmission mode 4.

The authors of [100] study and compare the performances of C-V2X in transmission mode 3 and transmission mode 4; their results highlight the advantages of the centralized solution from the point of view of the resource allocation efficiency and reliability. Furthermore, the benefits of adopting transmission mode 3 are confirmed in the results presented by Vukadinovic et al. in [99].

One of the parameters that is less recurrently studied in literature is the latency experienced when using one of the two access technologies. Mannoni et al., propose in [67] one of the few works performing such analysis. Their results, which are coherent to those presented in Chapter 2 and 5, point out that the channel access scheme adopted in 802.11p (CSMA/CA) is able to reach a latency that is one order of magnitude lower with respect to the resource allocation scheme of C-V2X (SPS). Indeed, with 802.11p it is possible to communicate with latency below the millisecond, while in C-V2X the minimum latency is never lower than 15 ms.

These results are verified for low densities and at reasonable distances (in [67], less than 350 m). As the density and the distance increase, instead, C-V2X is able to keep stable results, while the latency in 802.11p tend to explode, reaching very high values.

Chapter 3

MS-VAN3T: a multi-stack simulation framework for VANET applications testing in ns-3

3.1 Introduction

In the process of developing and testing a V2X application, a crucial role is played by network and mobility simulators: due to the intrinsic limitations arising from working with real vehicles (both from the economic, logistic, and security point of view), the scientific community developed a multitude of solutions enabling vehicular communications in simulation environment.

Among the V2X-dedicated network simulators implementing VANET solutions, the Veins framework [90] is surely one of the most established. It features the bidirectional coupling between the urban mobility simulator SUMO [61] and the network simulator OMNeT++ [76], via the so-called TraCI (Traffic Control Interface) interface. Beside its native WAVE/802.11p implementation, other projects, such as SimuLTE [96], have been merged with Veins to extend its capabilities. SimuLTE enhances Veins by adding in the simulation loop an accurate model of LTE. Veins was also extended by the Artery and Vanetza projects [85, 10], introducing a fully functional ETSI ITS-G5 model and a flexible platform for VANET applications prototyping.

Another simulation engine enabling the creation of complex network scenarios, and which leverages on a wide community of developers and tens of different frameworks, is ns-3 (network-simulator 3) [75]. With ns-3 it is possible to leverage on the following models to simulate VANETs scenarios:

- A model for 802.11p, including an accurate implementation of MAC and Physical

layers;

- A model for LTE communications (called LENA, [14]), comprehending the radio access part (both for UEs and eNBs), as well as an implementation of the control and data plane of the EPC (Evolved Packet Core), including the Mobility Management Entity (MME), the Serving Gateway (SGW) and the Packet Data Network Gateway (PGW);
- A model for C-V2X in transmission mode 4, using the framework proposed by Eckerman *et al.* [19], in which a C-V2X network unassisted communication is modeled.

We used the aforementioned models, along with an implementation of the TraCI interface (inspired by the project in [7]) for the coupling of SUMO and ns-3, to develop the first solution for vehicular application testing based on the ITS-G5 stack for ns-3. The name of the framework, that is presented and discussed in this thesis, is **MS-VAN3T** i.e., **M**ulti-**S**tack Framework for **VANET** applications testing in **ns-3**.

Unlike the other simulation frameworks for vehicular networks, where the users are mostly limited to the usage of a single predefined communication stack as access technology, MS-VAN3T gathers under a single open source repository all the state-of-the-art VANETs access layer frameworks available in ns-3, and allows any kind of application to be developed by easily and transparently switching the underlying channel access technology.

Therefore, in line with the relaxed approach undertaken by ETSI in ITS-G5 for the choice of the protocol to be used at the access layers, MS-VAN3T enables the development of V2X applications with the possibility of deploying them over LTE, C-V2X or 802.11p (thus, enabling V2V, V2I and V2N scenarios). MS-VAN3T comes with a full implementation of the ETSI ITS-G5 stack, comprising the model for Applications, Facilities, BTP and GeoNetworking. This thesis focuses on the higher layer of ITS-G5 implemented in MS-VAN3T, namely Applications and Facilities.

Beside the testing of applications using a multitude of different access technologies models, MS-VAN3T comes with the possibility of being easily turned into a V2X message emulator. Thanks to particular objects present in ns-3 (named `fd-net-devices`), it is possible to redirect all the messages generated in simulation to a specific interface of the host PC. The emulated V2X messages are encapsulated inside BTP and GeoNetworking headers and can be used to feed any kind of external applications.

The BTP and GeoNetworking PDUs generated in emulation mode, can be further encapsulated inside UPD and IPv4 so that the V2X messages generated by MS-VAN3T can be sent to any host of the network.

These kind of emulation functionalities, which to the best of the author's knowledge are not yet present in any of the existing open source VANET solutions, simplify the integration with hardware-in-the-loop scenarios, and enable the users to generate real V2X messages starting from the simple definition of SUMO simulations. Virtually, all

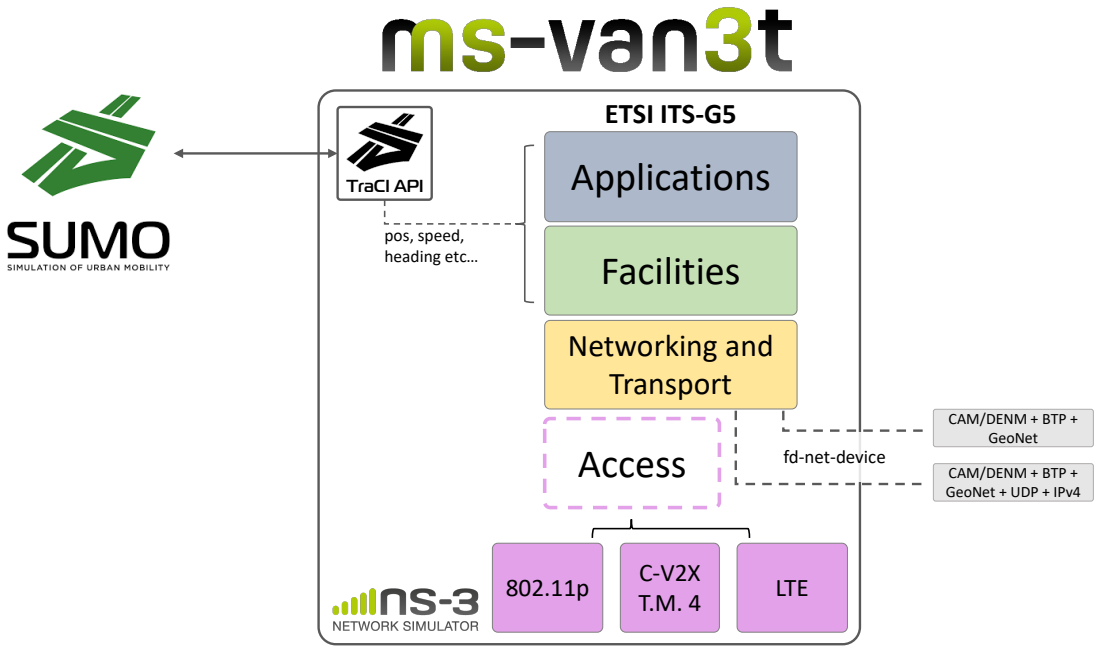


Figure 3.1: Main components of MS-VAN3T: on the left SUMO, allowing the simulation of complex mobility scenarios (including vehicles, pedestrian and other entities). On the right MS-VAN3T implemented in ns-3, communicating with SUMO through the TraCI interface, and implementing the ITS-G5 stack over 802.11p, C-V2X in transmission mode 4 or LTE, as well as an emulation mode, to redirect V2X messages outside the simulation environment.

the vehicular applications developers may leverage this kind of tool in the testing phase; this will tackle all the security, logistic and economic limitations imposed by working in the automotive context.

Therefore, the aim of this project is to support fellow researchers by providing them with an open source framework [71] that contains the tools needed to recreate centralized V2I/V2N and distributed V2V solutions, and that speeds up the development and testing phase, also thanks to a couple of applications included in the repository that can be used as a development baseline. Additionally, the presence of multiple access technologies under the same repository, may foster the investigation of technology coexistence issues, that nowadays are being faced by all the VANETs industrial and academic players.

3.2 MS-VAN3T framework architecture

MS-VAN3T is composed by two different pieces of software, interacting among each other through the TraCI interface, and that provide together a flexible platform for

application testing:

- SUMO (v1.8.0, at the time of writing), which allows users to dynamically model the physical positions of the nodes in the network (i.e., in our case, any road player) following realistic mobility patterns and dynamics. It enables the creation of arbitrarily complex scenarios involving vehicles, pedestrians, bicycles etc., and the interaction with the simulation elements using the TraCI API [101]. SUMO also provides the user with a GUI (Graphical User Interface), making it possible to visualize and interact with the simulated entities in a much easier way than just launching them and retrieving the final results.

- ns-3 [75] (v3.33 at the time of writing), an open source discrete-event simulator, which allows users to model all the aspects of the communication among the various entities, including the involved network stacks.

The elements of the simulation framework are schematized in Figure 3.1, in which the SUMO GUI is shown during the execution of a simulation, as well as the different ns-3 modules which can be selected for the V2X application testing and evaluation.

MS-VAN3T generates standardized V2X messages following the ETSI specifications, already introduced in Section 2.2.2. A model for the Facilities layers CA and DEN basic service is provided: it is possible, using MS-VAN3T, to generate, encode and decode standard compliant CAMs and DENMs messages. The CAMs generation follows the specification of EN 302 637-2 [26], where the transmission periodicity is dynamically set based on the generating entity's kinetics properties. The information about position, speed, heading, acceleration and exterior lights status, needed to populate the CAMs, are retrieved thanks to the integration with TraCI.

In a nutshell, TraCI establishes a TCP connection between SUMO and ns-3, which can be used to perform on-demand queries about each simulated entity. From the point of view of the vehicles, the TraCI interface acts as Vehicle Data Provider (VDP), and provides the Facilities layers with all the information required for messages creation. The TCP connection established by TraCI can be used also in the opposite direction to control the entities dynamics from ns-3; this can be useful, for example, if it is needed to simulate the case in which the driver takes some action after the reception of a particular V2X message.

The DENMs generation follows instead the specification of EN 302 637-3 [27]: DENMs are event-based messages, generated on-demand by the applications which use them to disseminate the information related to road hazards, collision risks and of generic events which are of potential interest for the receivers.

3.2.1 Facilities layers model

In its ITS architecture, ETSI defines the Facilities layers to support the distribution and processing of V2X messages, needed from the overlying applications to run their logic.

The implementation of ETSI-compliant features of the Facilities layers in ns-3 adds an important aspect to MS-VAN3T, namely the possibility to generate CAMs and DENMs messages that can be decoded also outside the simulation context: the messages generated can be received and processed by real V2X devices, which in turn can interact with the simulated entities, turning ns-3 into an emulator of CAM and DENM messages.

The model proposed in MS-VAN3T comprises both CA and DEN basic service, needed to manage the transmission and reception of CAMs and DENMs. All the messages are encoded using ASN.1, a syntax notation standard which allows the representation of complex data structures that can be read by any platform. This notation is completely programming language-agnostic, and allows different platforms, with different architectures, to transparently exchange information.

Within the CA and DEN basic services, two modules for ASN.1 CAMs and DENMs encoding and decoding were created. These modules are in charge of receiving ASN.1-encoded messages, extracting the relevant information, and providing the ITS-S applications with the requested data. Moreover, they should also provide the overlying applications with functions to encode and send ASN.1 messages to the network interface used for communication. The open source tool *asn1c* [91] is used to generate the encoding and decoding functions of the ASN.1 messages.

In Figure 3.2, the logic implemented by the CA and DEN basic service is depicted. The CA basic service encodes CAM messages independently from the overlying applications, that have no control over the CAM dissemination logic. This logic, that is non-trivial and that depends on the vehicle's dynamics, is internally managed by the CAM Transmission Management. The information needed to populate CAMs comes from a generic Vehicle Data Provider (VDP); in a real situation, this VDP will collect the information coming from the multitude of sensor on-board of the vehicle and will use the in-vehicle network (e.g., the CAN bus) to deliver the information to the CA basic service. In MS-VAN3T the VDP is linked with TraCI, from which the information on vehicle's dynamic is parsed and returned to the CA basic service. The modularity of MS-VAN3T enables any kind of VDP to be plugged to the CA basic service, with low effort.

Once a CAM is ready to be sent, the CA basic service delivers it to the lower layers of the communication stack. At reception side, the CA basic service should manage the CAMs received from the underlying layers, and decode the ASN.1-encoded messages. Finally, the information included in the CAM should be delivered to the requesting ITS-S applications, through an appropriate interface. The DEN basic service provides similar features to the overlying applications, with the difference that, being the DENMs event-based messages, they are triggered directly by the ITS-S applications.

The complete list of features included in the two models of CA and DEN basic service in MS-VAN3T can be found in Appendix A.

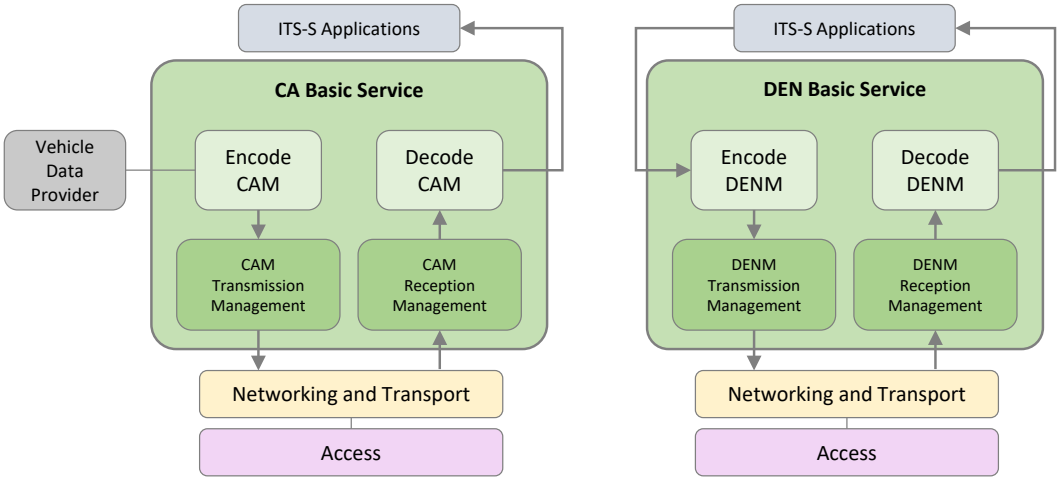


Figure 3.2: CA and DEN basic services implemented in MS-VAN3T.

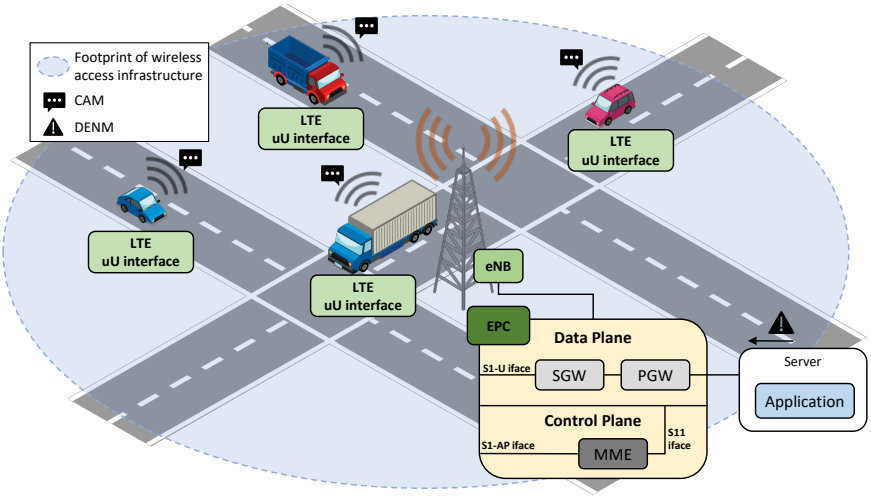
3.2.2 V2I/V2N scenarios

MS-VAN3T enables the possibility of modeling V2I and V2N scenarios, where the road players send their CAMs toward a centralized entity (e.g. a server), which collects them, runs the application logic, and replies back with DENMs when needed. Depending on the communication technology chosen at access layer, a V2I or a V2N-like scenario is simulated.

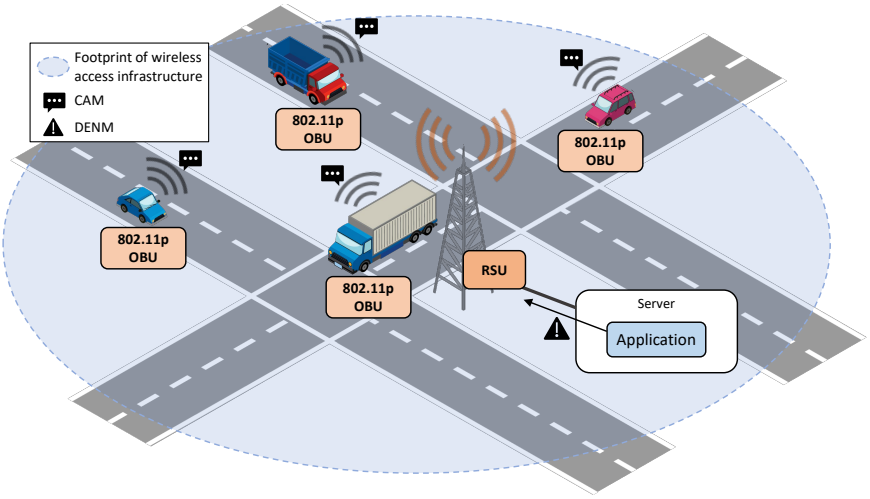
Two communication models can be used to enable the connectivity of the vehicles towards the centralized service. Depending on which one is selected, the resulting scenario can be seen as V2I or V2N. The two models are:

1. 802.11p, using the model available in ns-3. In this case, vehicles are equipped with 802.11p-compliant OBUs (On Board Units), and broadcast their CAM messages, which are received by a remote host that is connected to a RSU (Road Side Unit). The access layer communication model focuses both on MAC and Physical layers of 802.11p and does not implement any kind of Congestion Control algorithm. Due to the proximity between the clients and the server, and due to the fact that the simulated entities belong to the same subnet, it is possible to tag the resulting scenario as V2I. The 802.11p-based configuration is depicted in Figure 3.3b.

2. LTE, using LENA as simulation framework, in which a standard LTE network is established, with the vehicles acting as UEs (User Equipments) connected (through their Uu interface) to the eNB (eNodeB), that is in turn connected to the EPC (Evolved Packet Core) through the S1-U interface. The EPC implements the SGW (Serving Gateway) and PGW (Packet Data Network Gateway) blocks. The PGW is connected to a remote host that runs the application server logic. This scenario falls into the case of V2N communication, since the UE is connected to a service



(a) Centralized scenario with LTE as communication technology. The simulated entities send CAMs through their Uu interfaces toward the server, which is connected to the EPC via the PGW.



(b) Centralized scenario with 802.11p as communication technology. The simulated entities send CAMs through their 802.11p OBU interfaces toward the server, which is directly connected to the 802.11p RSU.

Figure 3.3: Centralized scenarios based on LTE and 802.11p.

provider that can be ideally anywhere in the Internet. As far as the control-plane is concerned, LENA implements a model for the MME (Mobility Management Entity) node, that is connected to the eNB via S1-AP interface, and to the SGW-PGW block through the S11 interface. The main functions of the MME include the session initialization and bearer management. Differently from the 802.11p scenario, in which the messages are broadcasted directly using GeoNetworking, in this case

the messages should traverse the IP-based network generated by the LTE framework, therefore should be encapsulated into UDP and IPv4. Due to the absence of a MBMS (Multimedia Broadcast Multicast Service) functionality in the LTE model, all the messages are sent as unicast. The resulting configuration can be seen in Figure 3.3a.

3.2.3 V2V scenarios

With MS-VAN3T it is also possible to model distributed scenarios, where vehicles are configured to broadcast CAMs and DENMs directly among themselves, using V2V-based protocols. In this case, the application logic can be deployed directly inside the vehicles, in a purely distributed fashion.

The aforementioned configuration is enabled by the integration of two communication models, one of which has already been introduced in the previous section:

1. 802.11p, using the 802.11p model available in ns-3. As in the V2I scenario, vehicles are equipped with OBUs, but in this case they broadcast CAMs and DENMs among themselves.

2. C-V2X, using the access layer model proposed by Eckerman *et al.* [19], in which a C-V2X network, with PC-5 interfaces configured in transmission mode 4, is established among the road players. In this model, the vehicles communicate through the direct communication interfaces and exchange messages with their peers through the sidelink, without relying on the eNB. The resulting model represents an out-of-coverage communication scenario.

The V2V scenario, using either C-V2X transmission mode 4 or 802.11p, is depicted in Figure 3.4. In this case, each entity transmits and receives packets within its radio-range, represented by the green circle around the vehicles.

3.3 Building applications on top of MS-VAN3T

The simulation framework presented in this thesis comes with two sample applications, one for the V2I/V2N case, developed with a centralized client/server architecture, and one for the V2V case, developed in a distributed fashion. Each of them has been developed with the aim of showcasing the potential of the framework, and to provide significant example of the featured ETSI-compliant message encoding and decoding.

3.3.1 V2I/V2N application: Area Speed Advisory

The application proposed for the centralized V2I/V2N communication models is based on a client/server architecture. As far as the LTE communication is concerned, the

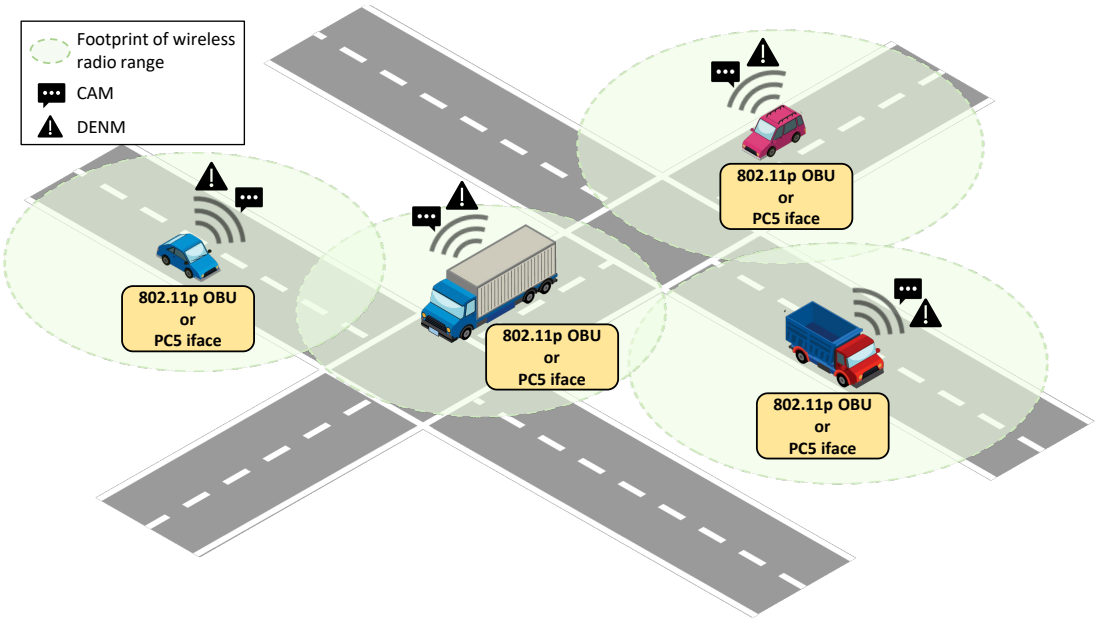


Figure 3.4: Distributed scenario with 802.11p or C-V2X as communication technology. The simulated entities send V2X messages through their direct communication interfaces. Messages are received and processed by the other vehicles.

server is placed in a remote host connected to the EPC through the PGW, while it is located in a host directly connected to the RSU when 802.11p is deployed.

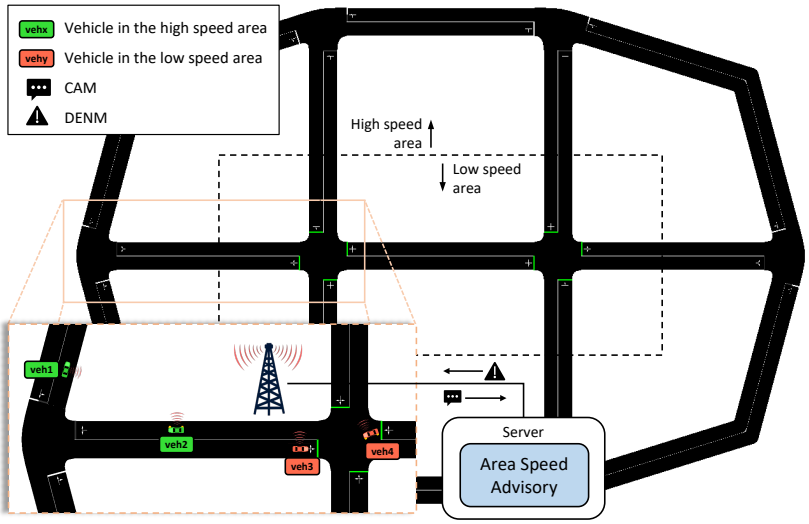
The main difference between the two communication models, from the application point of view, resides in the higher end-to-end latency experienced when using LTE: in this case, indeed, both CAMs and DENMs need to pass through the EPC before they are delivered, while the usage of 802.11p allows the server to be deployed directly behind the RSU, just one hop away from the clients.

The map used in this context includes two road crossings connected through a central two-way street, and is depicted in Figure 3.5a. The network access point (eNB for LTE and RSU for 802.11p) is placed at the center of the map, with sufficient coverage to ensure connectivity to all the vehicles traveling in the simulated area.

The core logic of the application is to divide the map in two different zones: the first zone is the central one, comprising the two crossings, in which the maximum allowed speed is 25 km/h. The second zone is instead the outer one, where the vehicles are allowed to reach a speed up to 100 km/h.

Due to the absence of MBMS functionalities in the LTE model (that makes it unsuitable to generate and manage broadcast packets), the application follows a different logic, depending if it is deployed in the LTE or in the 802.11p scenario.

For what concerns the LTE scenario, the server monitors the position of the vehicles, by reading the CAMs, and warns them if they enter an area with different speed restrictions. The vehicles are configured to periodically send their CAM messages in



(a) Two screenshots from SUMO-GUI, showing the map and the sample application in action for the V2I/V2N communication models. The middle zone, comprehending the two crossings, has speed restrictions. The image also shows the network access point, where the server is connected. CAMs are transmitted from the vehicles to the server, and DENMs are transmitted from the server to the vehicles.

(b) An example of a real implementation of the Area Speed Advisory application, with the server taking care of slowing down the vehicles approaching a school.

Figure 3.5: The Area Speed Advisory application.

unicast to the server. The server, aware of the boundaries of the two areas, analyzes the position of the vehicles and when it realizes that a vehicle is moving from a zone to another, it generates and sends a unicast DENM message, warning the driver about the

necessity of reducing his/her speed or about possibility to increase it.

The 802.11p scenario instead can completely benefit of the GeoNetworking capabilities: the server generates and broadcasts DENMs using GeoBroadcast (see Section 2.2.2). The destination area set in GeoBroadcast is the central zone, so that each vehicle entering it will receive the DENM informing about the different speed limitations. In this case, the server will not monitor the position of each vehicle, but will simply generate DENMs with fixed frequency that, thanks to GeoNetworking, will be correctly disseminated in the relevant zone.

This kind of application can be used to signal the presence of a road hazard (e.g., ice on road), or to delimit certain sensitive zones (e.g., school areas, dangerous crossings, etc.), as shown in Figure 3.5b.

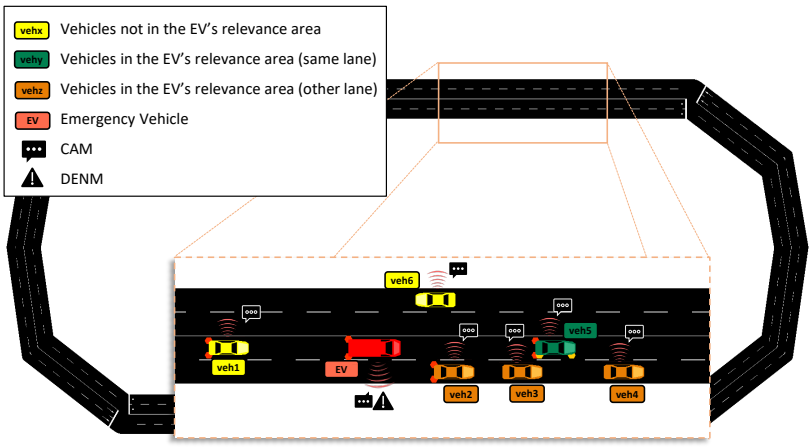
Figure 3.5a shows the V2I/V2N sample application running in SUMO, with ns-3 managing the communication and application logic. Four vehicles are depicted, two in the high-speed area (*veh1*, *veh2*) and other two in the low-speed area (*veh3* and *veh4*). In this case, once a vehicle receives a DENM informing the driver to increase the speed, ns-3 tells SUMO (through TraCI) to change the vehicle color to green (just for visualization and debugging purposes) and increase its speed. Conversely, whenever a vehicle moves from the high-speed to the low-speed area, its color is changed to red and its speed is reduced.

When using LTE, CAMs are sent as unicast messages from the vehicles to the server running the application logic. At the same time, the server generates and sends unicast DENMs to notify the vehicles changing speed area. Al the messages are encapsulated into BTP, GeoNetworking, UDP and IPv4. Conversely, when using 802.11p, all the messages (both CAMs and DENMs) are broadcasted among the entities, and are encapsulated into BTP and GeoNetworking headers.

3.3.2 V2V application: Emergency Vehicle Alert

The application proposed to showcase the potentiality of the MS-VAN3T's V2V models is called Emergency Vehicle Alert (EVA). In this case, the supported access technologies are 802.11p and C-V2X in transmission mode 4. Similarly to the V2I/V2N scenario, the application relies on ETSI-compliant CAM and DENM messages (in this case, directly exchanged between road players) to actuate its logic. As per-standard, all vehicles exchange CAM messages to inform nearby nodes about their current status. The scenario also includes, however, the presence of Emergency Vehicles (EVs), which should be able to travel without being slowed down by the other vehicles.

An EV could be an ambulance, a police motorcycle, or a firefighter vehicle, as shown in Figure 3.6b. In particular, we modeled a scenario in which two EVs periodically send DENM messages to all nearby vehicles. Upon reception of a relevant message (i.e., a DENM message from an approaching EV), a normal vehicle will try to limit its hindrance to the EV. If it is traveling on the same lane as the EV, it will try to change lane. If this is not possible (for instance due to the other lane being occupied), the vehicle will speed



(a) Two screenshots from SUMO-GUI, showing the map and the sample application in action for the V2V communication model. All the vehicles exchange CAMs through direct communication. In addition, the EV (in red) broadcast DENMs, which ar in turn received by the surrounding vehicles, processed and used to take evasive action.

(b) An example of a real implementation of the Emergency Vehicle Alert application, with an ambulance transmitting DENMs to inform vehicles about its presence.

Figure 3.6: The Emergency Vehicle Alert application.

up and try to perform a lane merge maneuver as soon as possible. If on a different lane, it will slow down. In this way, the EV will be able to overtake a normal vehicle without being forced to reduce its speed.

The map used in this case represents a urban scenario, where normal vehicles have

a maximum speed varying between 30 km/h and 60 km/h and EVs can, instead, travel up to 75 km/h; a circular road segment is involved, with two lanes for each direction of travel. The configuration chosen (two lanes roads and vehicles' maximum speed) seem to be fairly realistic, as this is a common case which is often encountered in large cities all over the world.

A screenshot of SUMO GUI while running the application is shown in Figure 3.6a. The red vehicle is the EV broadcasting DENMs, while the orange vehicles (like *veh2*) are nodes that have correctly received and parsed a DENM from the EV and are slowing down to be safely overtaken. The green vehicle (*veh5*) is instead a car traveling on the same lane as the EV. It is thus trying to accelerate to change lane as soon as possible and let the EV go by. Finally, the yellow vehicles (such as *veh1 and veh6*) are cars which received the DENMs sent by the EV, but do not need to react, as they may be traveling in the opposite direction, they might not be directly interested by the EV trajectory or they may be still too far away.

3.4 Simulating with MS-VAN3T

The main objective of the MS-VAN3T project is to provide fellow researchers with a tool that can be used to easily develop any kind of vehicular application, and to deploy it over any access technology. MS-VAN3T can be used to measure application-related KPIs, to test the V2X application features, but also to evaluate the performances of the underlying access technology. In this section, the main simulation results both related to the applications metrics and to the access technology performances are presented.

3.4.1 V2I/V2N application performances

As previously mentioned, the main target of the proposed V2I/V2N applications is road safety. A couple of traffic lights are deployed to manage the two intersections at the center of the map; the two traffic lights, however, were modified to show always green to all incoming vehicles. In this way the Area Speed Advisory application can be evaluated in the worst case scenario: the vehicles approach the intersections at maximum speed, and realize about the other vehicles' presence when they already entered the intersections area. In some cases, they will be unable to stop and avoid a collision. This configuration mimics the case of a distracted driver, of a blind unregulated intersection or of a traffic lights system failure.

The simulations are then evaluated by observing the average number of collisions with and without the Area Speed Advisory application. The tests involved an increasing number of vehicles, starting from 10, to 40 vehicles in total. The results plotted show the total number of collision occurred at the two road crossings as a function of the total number of vehicles present in the scenario. The vehicles could reach a maximum speed of 100 km/h, and for each density we ran 100 simulations, each lasting 200 s, by changing the mobility trace.

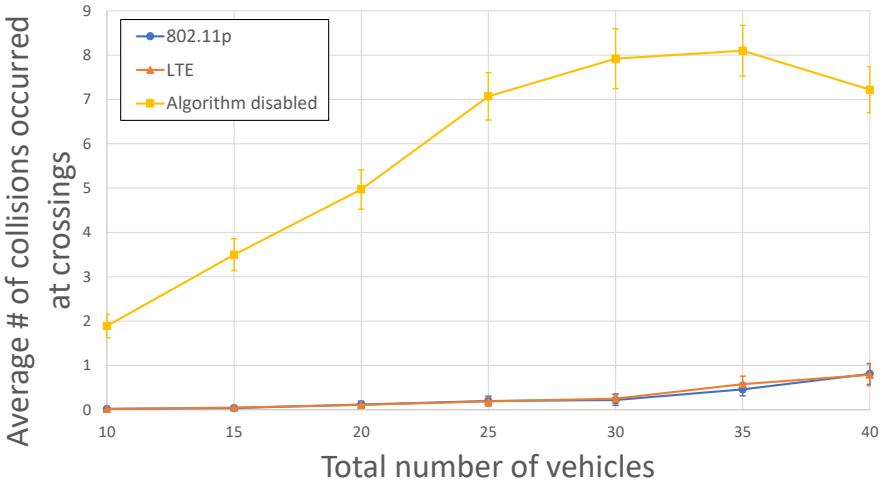


Figure 3.7: Results of the V2I/V2N simulations. The average number of collisions at road crossings as a function of the total number of vehicles present in the scenario is reported.

The results are plotted in Figure 3.7, where the increasing number of vehicles are reported on the x axis and the average number of collisions at road crossings on the y axis. For each value, the 95% confidence interval on all the simulations is plotted. The results show the benefits of the Area Speed Advisory application on the total number of road crossing collisions. The sample application developed on top of MS-VAN3T, which basically slows down the vehicles approaching the intersections, was able to avoid a large number of collisions. The average number of collisions reported, when the algorithm is disabled, increases up to 8 collisions per simulation, when the total number of vehicle is between 30 and 35. Then, due to the increased traffic, and to the consequent drop in the average vehicles speed, the total number of collisions starts to decrease. When the algorithm is instead enabled, the number of collisions is always low, with an average of less than 1 per simulation, even when the density is very high.

3.4.2 V2V application performances

As the main metric for the assessment of the V2V-based application, the focus is on the average speed of the EV, under different traffic conditions, comparing the case in which the alerting system is enabled (i.e., DENMs are sent by the EVs), either through 802.11p or C-V2X, to the case in which it is disabled.

The application is evaluated by increasing the total number of vehicles, ranging from 10 to 40. The map used is the one described in Section 3.3.2. For each case 100 simulations are run, each lasting 200 seconds. The mobility traces are different at every run and include two EVs, one per travel direction. The speed of the two EVs has been averaged over all the simulations.

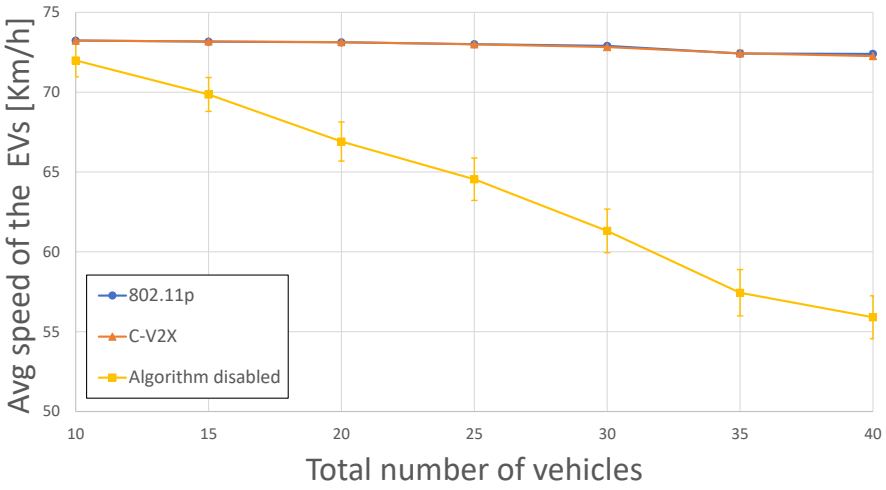


Figure 3.8: Results of the V2V simulations. The average speed kept by EVs as a function of the total number of vehicles present in the scenario is reported.

The results are depicted in Figure 3.8, in which the increasing number of vehicles is reported on the x axis and the measured average speed of EVs on the y axis. For each value, 95% confidence intervals over the 100 simulations are reported.

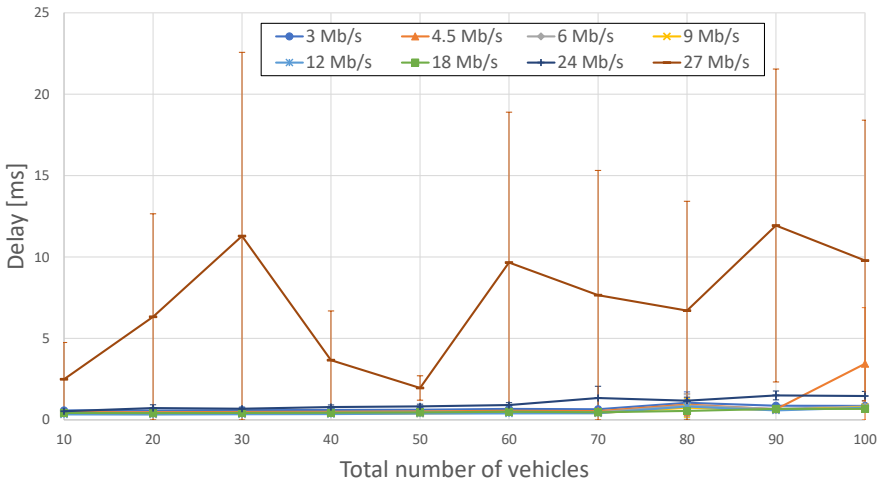
The results show how the proposed solution can be an effective, yet simple, algorithm for an EV use case, and how MS-VAN3T can be effectively used to assess safety applications performances even in V2V scenarios. Indeed, the EVs are almost always able to keep their maximum speed when they are sending DENMs to nearby vehicles, no matter the traffic density or pattern. Instead, when no DENMs are sent and the vehicle density starts increasing, a quite consistent speed drop (sometimes more than 10 km/h) can be observed. Importantly, the selected V2V technology does not make a significant difference when it comes to the algorithm effectiveness, at least as long as all the needed DENMs are correctly received by the involved vehicles.

All the analyzed data, depicted in Figures 3.7 and 3.8, has been obtained either through SUMO output files or by coding a CSV logging mechanism into ns-3 (available in the source code of our sample applications).

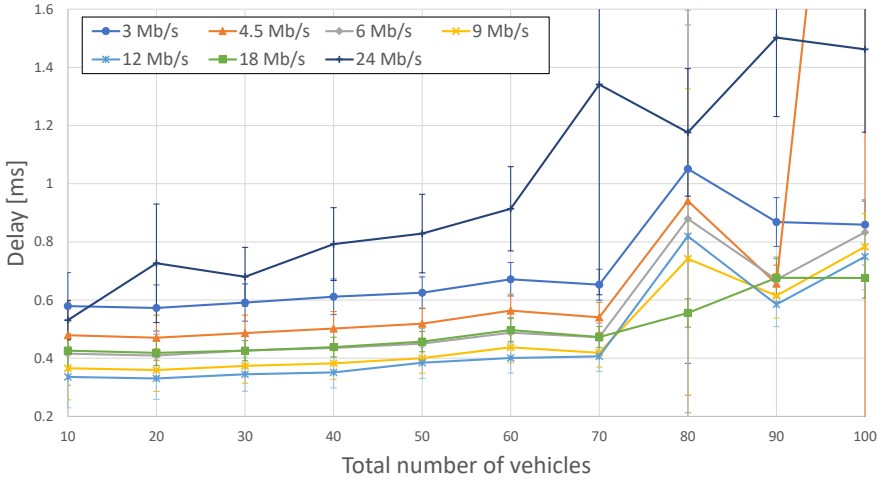
3.4.3 Access technologies performances

Being the various access layers models developed by different persons/teams, it is important to stress that the results here presented are strictly dependent on the specific implementation of the communication technology. Therefore, the same tests performed on a different models of the same technology may lead to slightly different results.

To analyze the access protocols performances, a tool named Flow Monitor was used when possible [17]. With Flow Monitor it is possible to track each IP flow in the ns-3 simulation, thanks to probes that are installed at IP layer. Flow Monitor was used



(a) Delay computed with Flow Monitor when communicating through 802.11p. Each line corresponds to a different datarate, with the plot reported as function of the total number of vehicles present in the scenario.



(b) Zoom of the plot in the region between 0.2 and 1.6 ms.

Figure 3.9: 802.11p delay (one-way).

to compute the end-to-end delay, the delay jitter and the Packet Drop Ratio (PDR) in 802.11p and LTE. Unfortunately, due to the specific network implementation, it is not possible to attach the Flow Monitor probes in C-V2X-based nodes; therefore in C-V2X the protocol performances are inferred using information available at application layer.

All the values reported have been obtained by running 10 simulations, each 1000

seconds long. The map is the same used for the V2I/V2N application, with a centralized entity (either connected through 802.11p, LTE or C-V2X) which gathers the CAMs messages and sends DENM messages. The results are focused on delay, on delay jitter and on Packet Drop Ratio (PDR).

802.11p

When using 802.11p as communication protocol, in MS-VAN3T vehicles normally transmit CAMs in broadcast using BTP and GeoNetworking. Since Flow Monitor install probes at the IP layer of the transmitting/receiving entities, in this case the model has been slightly modified and added with UDP/IPv4 capabilities. Therefore, CAMs are sent in unicast to the central server which is, in turn, connected to the RSU. The server replies back with DENMs, according to the logic implemented in the V2I/V2N application mentioned earlier.

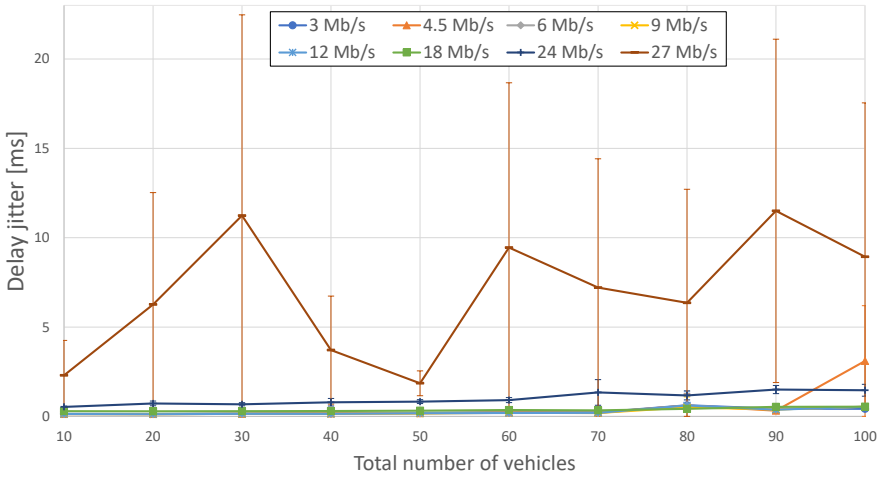
Therefore, each vehicle's OBU (and the RSU connected to the server) contends for the shared channel and packet collisions are likely to happen. The simulations, in this case, have been carried on by configuring the OBUs to transmit using one of the possible datarate of 802.11p. The plots, shown in Figure 3.9, 3.10, and 3.11 are all reported as a function of the number of vehicles present in the scenario (from 10 to 100).

Figure 3.9 shows the one-way delay, i.e., the delay experienced when transmitting a packet from one node to another, and the corresponding 95% confidence interval. The results suggest that 802.11p is able to keep a very low delay when transmitting at datarates up to 24 Mb/s. When transmitting at 27 Mb/s instead, due to the MCS parameters adopted, the connection becomes highly unstable: the increased number of retransmissions cause the nodes to perform multiple backoffs for every transmission. For this reasons, the delay experienced is higher with respect to the case in which lower datarates are adopted.

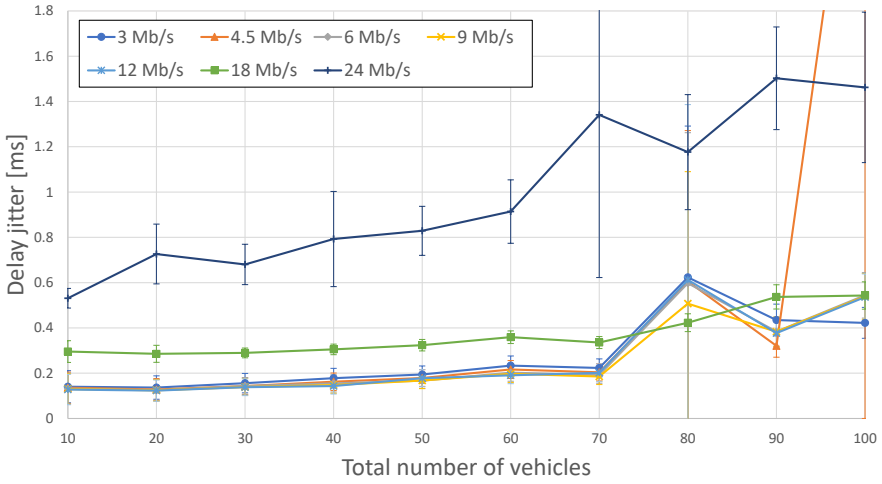
In Figure 3.9b, a zoom on the region between 0.2 and 1.6 ms is depicted. Here, all the different datarates are clearly visible, and some considerations can be made: from 3 Mb/s up to 12 Mb/s the delay decreases, and this phenomenon is mostly due to the reduction in transmission time. Then, starting from 18 Mb/s, due to the aforementioned reasons linked to the MCS parameters, the delay starts to increase and the communication becomes unstable.

All the previous deductions are supported by what is shown in Figure 3.10. The delay jitter (computed as the standard deviation in the measured delay) is very low for the lower datarates, while is remarkably high when transmitting at 27 MB/s, denoting a highly unstable and unpredictable communication.

The plots related to the PDR are shown in Figure 3.11. At high datarates, also in this case, the protocol performances seem to suffer the coding scheme settings: when transmitting at 27 Mb/s, a very high number of packets are lost (up to 50%); at 24 Mb/s the situation is improved, but still around 30% of packets are lost. The value decreases only when adopting a transmission rate equal or below 18 Mb/s, as shown in the zoomed



(a) Delay jitter computed with Flow Monitor when communicating through 802.11p. Each line corresponds to a different datarate, with the plot reported as function of the total number of vehicles present in the scenario.

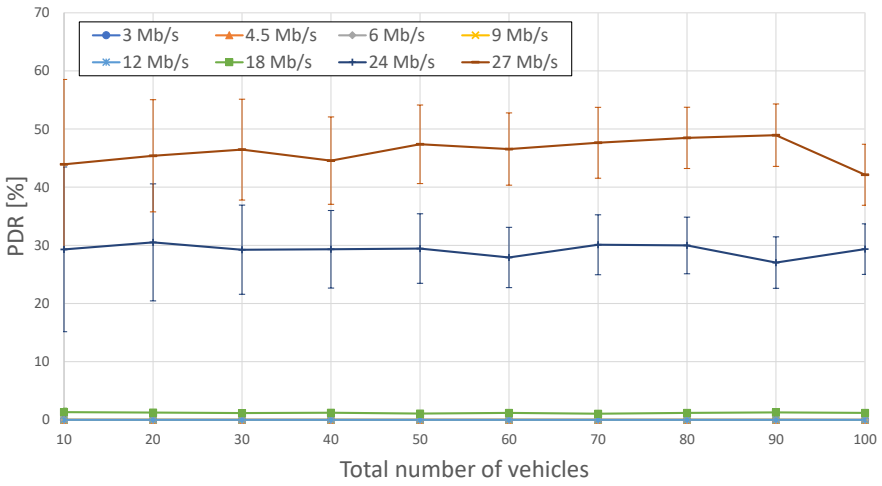


(b) Zoom of the plot in the region between 0 and 1.8 ms.

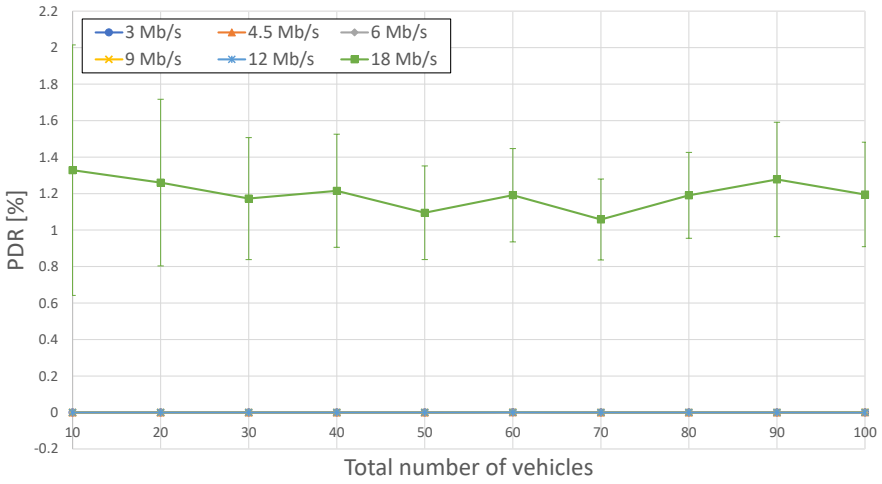
Figure 3.10: 802.11p delay jitter.

plot in Figure 3.11b.

It is worth to mention that the PDR values here reported are computed at IP layer: 802.11p does not implement any kind of low level retransmission technique, therefore when two or more stations access the channel at the same time, the corresponding packets are lost.



(a) PDR computed with Flow Monitor when communicating through 802.11p. Each line corresponds to a different datarate, with the plot reported as function of the total number of vehicles present in the scenario.



(b) Zoom of the plot in the region between 0 and 2.2 ms.

Figure 3.11: 802.11p delay (one way).

LTE

The scenario is configured to characterize the LTE performances is similar to the previous one: the vehicles are equipped with UEs that are used to transmit unicast CAMs to the central server using the Uu interface. The central server is connected, thanks to the PGW, to the EPC. The point-to-point link used to connect PGW and server introduces a fixed delay of 5 ms. The LTE parameters chosen are the default set in LENA: 25 RBs for

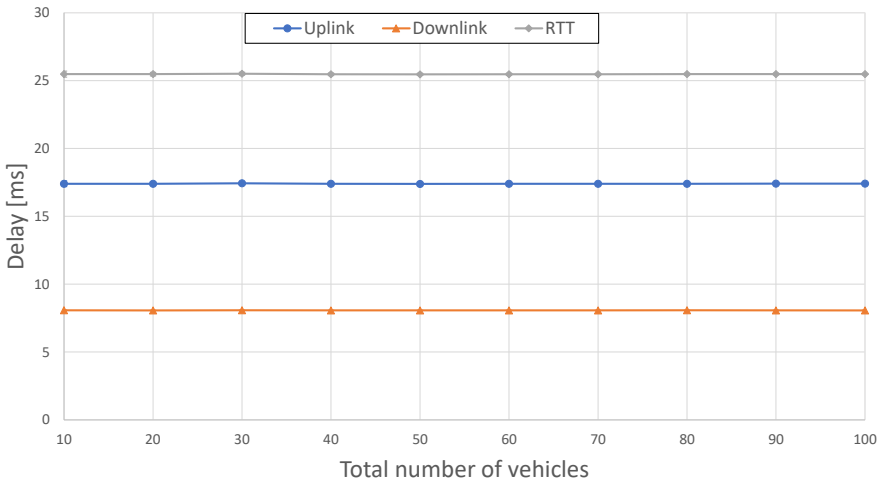


Figure 3.12: Delay computed with Flow Monitor when communicating through LTE. Values corresponding to Downlink, Uplink and RTT measurements are reported.

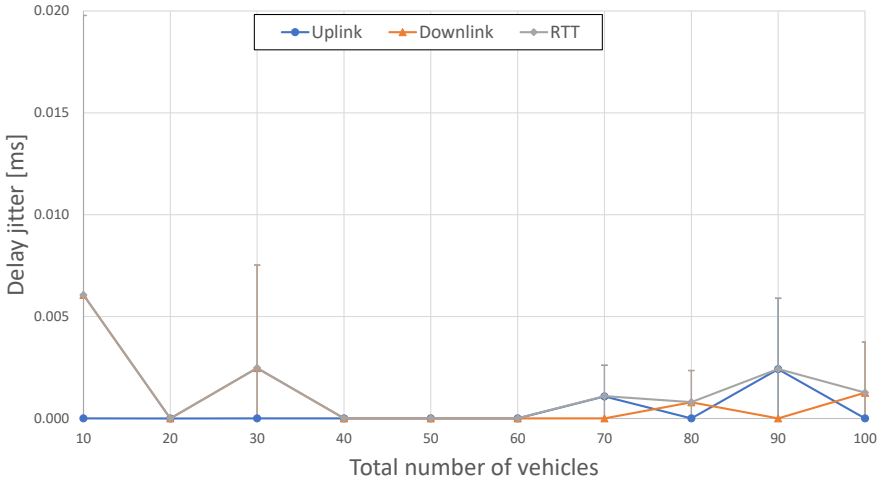


Figure 3.13: Delay jitter computed with Flow Monitor when communicating through LTE. Values corresponding to Downlink, Uplink and RTT measurements are reported.

the Uplink and Downlink channel, with EARFCN (E-UTRA Absolute Radio Frequency Channel Number) set to 100 for Downlink channel and to 18100 for the Uplink channel. Due to the presence of the HARQ retransmission system, the results are expected to show an improved PDR, and a communication with higher (but more stable) delays. Also in this case, the results are plotted as a function of the number of vehicles present in the scenario.

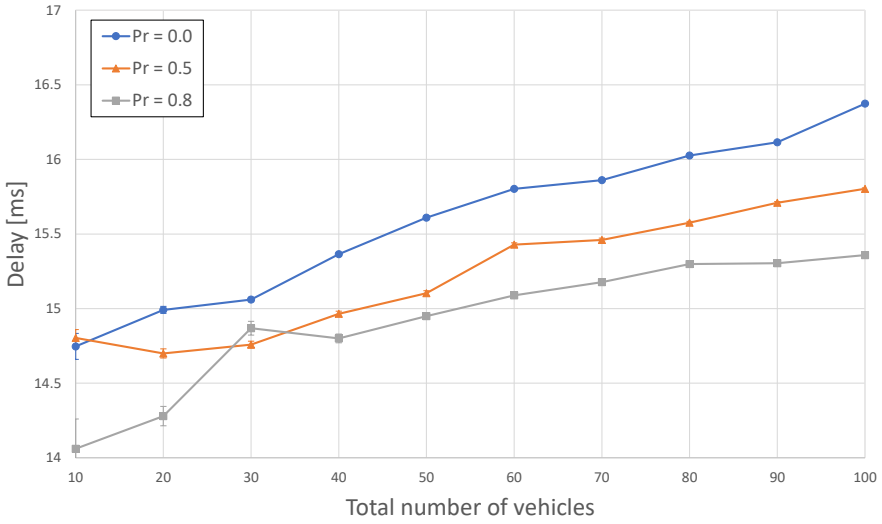


Figure 3.14: Delay computed at application layer when communicating through C-V2X. Plots corresponding to 3 different values of the resource reselection probability are reported.

The plot in Figure 3.12 confirms the behavior expected: the delay is 17 ms in uplink (where the CAMs are transmitted), 8 ms in Downlink (where the DENMs are transmitted), totaling a RTT of around 25 ms. The LTE network established can serve, without any problem, 100 UEs. Differently with respect to 802.11p, the results are stable, with variance close to 0, as depicted in Figure 3.13.

The PDR is not reported because its values is constantly 0: the presence of low-layer retransmission mechanisms (such as HARQ, in the case of LTE), makes the communication very reliable, with packet loss equal to 0 even when the number of transmitting nodes is high.

C-V2X

To analyze the C-V2X performances, the delay, delay jitter and PDR of the model is studied by varying the resource reselection probability P_r. In transmission mode 4, a vehicle reserves the selected subchannel for a number of consecutive transmissions randomly extracted between 5 and 15. This value is included in the SCI information and, as soon as all the retransmissions are done, a new resource is reserved with probability (1 - P_r), with P_r between 0 and 0.8. In the set of simulation here analyzed, the MCS 20 is used, the resource pool are composed by 3 subchannels (each formed by 10 RBs) and the resource reservation interval is set to 20 ms. All these parameters are tunable in the MS-VAN3T framework.

This scenario foresees vehicles broadcasting CAMs through the dedicated PC5 interface. As previously stated, in this case it is not possible to attach IP probes since

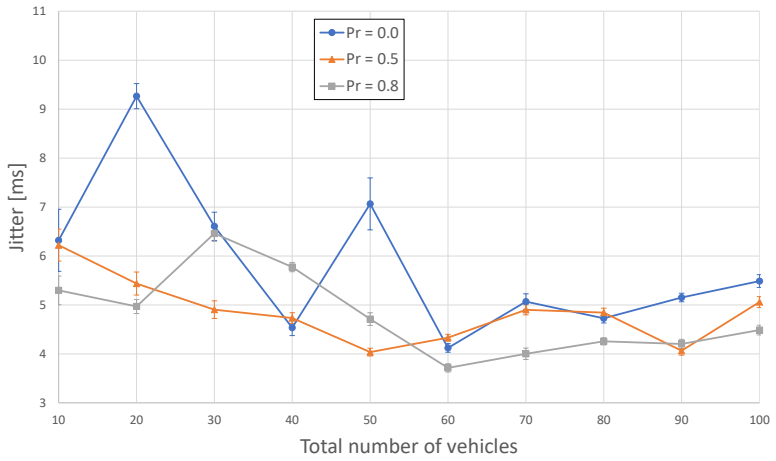


Figure 3.15: Delay jitter computed at application layer when communicating through C-V2X. Plots corresponding to 3 different values of the resource reselection probability are reported

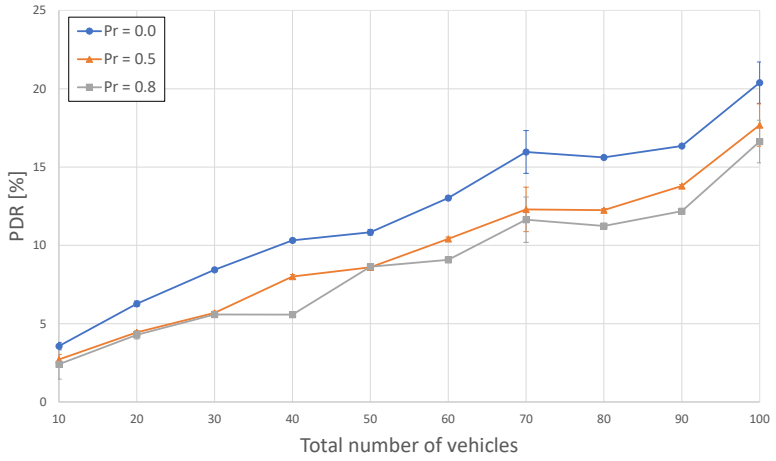


Figure 3.16: PDR computed at application layer when communicating through C-V2X. Plots corresponding to 3 different values of the resource reselection probability are reported

the C-V2X model included in MS-VAN3T makes use of raw sockets to implement the direct communication. For this reason, all the results here reported are collected at application layer: as soon as the CA basic service receives the message, it forwards it to the upper layer, where the application collects and log the received information. The performances of the system were studied by selecting 3 values for P_r, namely 0, 0.5 and 0.8.

The delay computed in this case shows a direct correlation with the number of vehicles present in the scenario. Moreover, it is clear that by increasing the probability

of reselecting a resource, the final delay decreases (around 0.5 ms each step). The results also suggest that, in general, C-V2X offers higher delays with respect to 802.11p. However, the communication seems to be way more stable: either by looking a the very small confidence intervals in Figure 3.14, or by looking at the small jitter (on average, around 5 ms), the cellular-based solution offers a reliable technology that can be used to enable performances demanding applications.

For what concerns the Packet Drop Ratio, plotted in Figure 3.16, a direct relation with the total number of vehicles present in the scenario is still highlighted. However, the model implemented in MS-VAN3T, at the time of writing, still does not implement any HARQ system. Therefore, in a real scenario, or in a situation in which HARQ is implemented to mitigate the packet loss, the results are expected to improve.

3.5 Emulating with MS-VAN3T

One of the core functionalities of MS-VAN3T is represented by the possibility of redirecting all the V2X messages created in simulations toward any of the physical or virtual network interface of the host PC. This important feature, implemented using the so-called `fd-net-devices` can be very useful in the testing of vehicular applications based on V2X messages: the CAMs and DENMs messages generated in simulation by MS-VAN3T can easily be relayed by the host PC to, for example, a 802.11p radio. Therefore, those messages can be received by real vehicles equipped with 802.11p sensors, that will decode the information as if the messages were created by other real vehicles in the surroundings. This artificial communication can be used to test V2X-based safety applications, without having to equip an entire fleet with V2X sensors.

To help the user in the implementation of such configuration, MS-VAN3T comes with an example showing how to redirect all the V2X messages to a specific interface in the host PC. When using the emulation capabilities of MS-VAN3T, each vehicle transmitting CAMs and DENMs will have to spoof a MAC address, which will then be assigned to the dedicated `fd-net-device`. Therefore, the user will be required to set the target interface of the host PC in promiscuous mode. Each vehicle is configured to generate CAMs according to the CAMs dissemination rules standardized by ETSI, while DENMs are generated with frequency 1 Hz.

Additionally, the aforementioned example comes with the possibility of redirecting the messages using UDP toward a specific IP address, by simply specifying the address and the port to be used in transmission.

Figure 3.17 shows the dissection of the V2X packets generated by MS-VAN3T on the host PC, sniffed with Wireshark on the target interface (`wlp63s0`). The fact that the packet dissection is successful validates the entire MS-VAN3T stack, since those messages, if received by an external entity implementing a standard compliant ITS-G5 stack, can be correctly received and decoded.

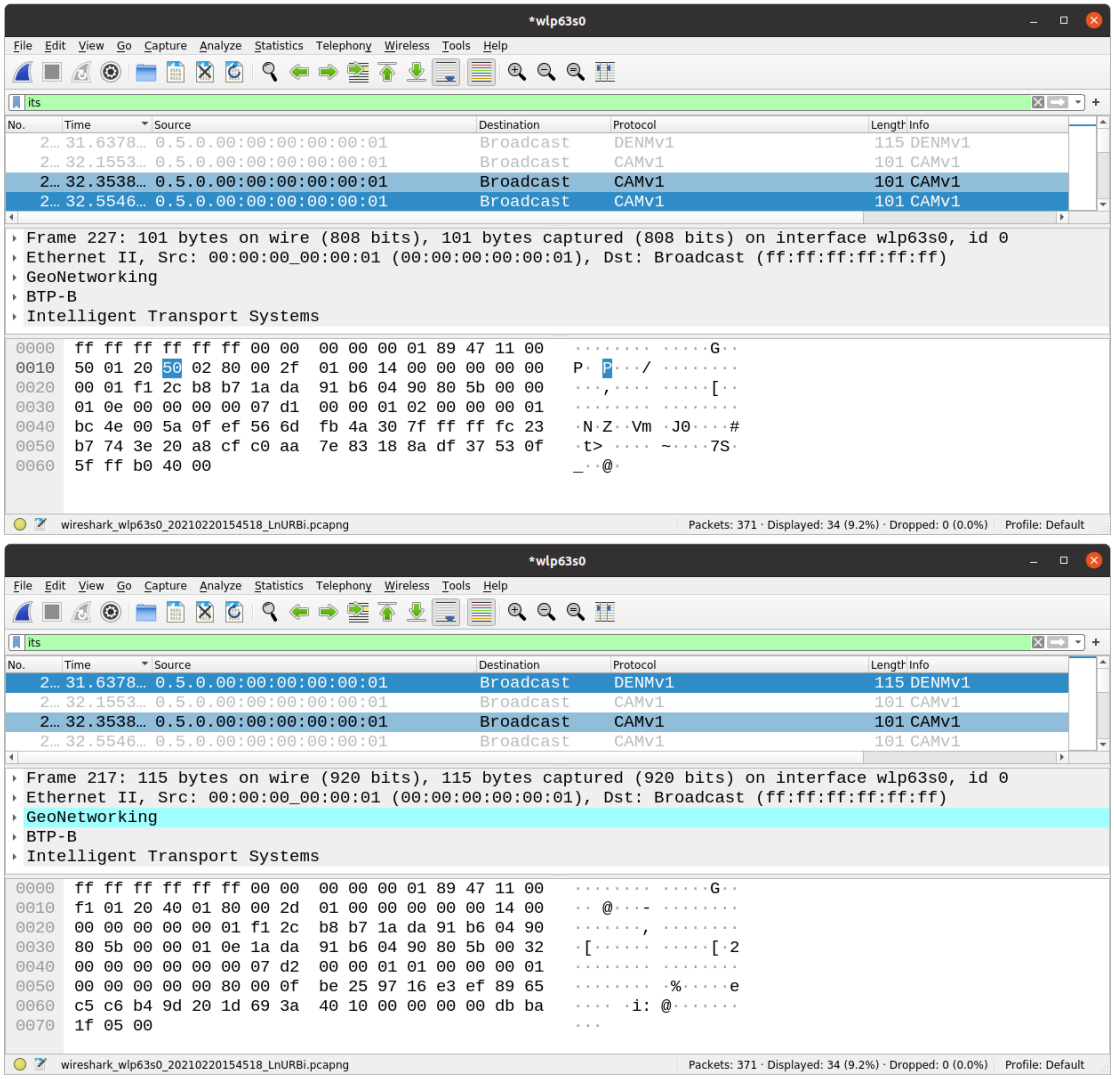


Figure 3.17: CAMs and DENMs generated by MS-VAN3T and transmitted to the physical interface `wlp63s0` of the host PC. The messages are sniffed using Wireshark, which is able to correctly dissect them, as it they were generated by real V2X entities.

3.6 Future work

MS-VAN3T is actively improved and new features are tested and implemented on a daily basis. At the time of writing, the core functionalities of Facilities, BTP, and GeoNetworking are being improved.

As a future step, an important feature that will be added is the possibility of using alternative mobility trace generators with respect to SUMO. In particular, an idea could

be of adding the possibility of injecting real GPS traces. In that case, each simulated node inside ns-3 will link its Vehicle Data Provider to the dedicated GPS trace, so that the generated CAMs will be filled with positioning information coming from real devices.

Another important feature that will be developed in the future, is the integration with other access technologies: the first 5G model for ns-3 is currently in development phase [4], and it will be integrated in MS-VAN3T to give the possibility of developing applications relying on the 5G technology. As soon as other models of potential interest for the vehicular world will be available in ns-3, like 802.11bd (the evolution of 802.11p) or 5G-V2X, they will be included in MS-VAN3T.

Chapter 4

V2X-supported collision avoidance systems

The number of fatalities on roads remains unacceptably high. With a peak of 1.35 million, as reported in the "Global Status Report on Road Safety 2018", traffic accidents represent the 8th leading cause of death for people of all ages and the 1st for children and young adults from 5 to 29 years of age [42]. The severity of this problem, in terms of deaths and injuries, is often under-estimated by governments and local authorities. The under-reporting of traffic accidents fatalities is still common in many regions of the world, and this often results in a low priority being given to road safety with respect to other public health concerns that are less deadly, but have higher media coverage. Motivated by these ghastly figures, safety has emerged as a prominent application of vehicular networks.

As mentioned in Section 2.2.2, ETSI dedicated an entire set of Technical Specifications to road safety applications [36, 33, 34, 35], and included Road Hazard Signaling (RHS), Intersection and Longitudinal Collision Risk Warning (ICRW and LCRW) among the prominent use cases of Basic Set of Application (BSA) for ITSs. Those applications are considered as *primary road safety applications*: as introduced in Clause 4 of [35], primary road safety applications are those ITS applications whose aim is to reduce the risk of collision and improve road safety.

In this thesis, a novel application for detecting potential collisions at the intersections (thus, a ICRW application) is presented, and adapted to a number of different scenarios. The study focuses on the possible network architectures where the ICRW application may be deployed, as well as on the number of technical challenges imposed by this kind of application. The proposed solutions are then implemented and validated through simulations in MS-VAN3T, the vehicular network framework presented in Chapter 3.

4.1 ICRW according to ETSI: an overview

ETSI defines ICRW as an application in charge of providing collision risk warning to drivers, by delivering punctual information in case of risk detected. ICRW is based on the processing of basic V2X messages, CAMs and DENMs for ITS-G5-based communications, and BSMs and ICAs for WAVE-based applications. Since ICRW has been specified by ETSI, from now on the services is considered as deployed in an ITS-G5 scenario, with vehicles and vulnerable users broadcasting CAMs, and with collision alerts coded into DENM messages. Additional messages that can be used to assess the collision risk can be generated by other services such as Traffic Light Maneuver (TLM), Road and Lane Topology (RLT) and Infrastructure to Vehicle Information (IVI). In addition to the potential collision detection functionality, the integration with TLM, RLT and IVI enables the possibility of detecting traffic sign violations at intersections (functionality that, however, is not implemented or discussed in this work).

An ICRW compliant entity should be able to generate, receive and process V2X messages. In particular, two functional modes are specified: ICRW originating mode and ICRW receiving mode. The former requires the compliant ITS-S to be able to generate CAMs, with the additional capability of tweaking the transmission rate when approaching sensitive locations. Furthermore, it requires the possibility of triggering DENMs to inform other entities of the detected risk. The latter requires instead the ability of receiving V2X messages, process them and take appropriate actions. As an example, in case of a vehicle ITS-S receiving a DENM related to a collision risk detected, ICRW should issue a warning to the driver, and eventually it should trigger a DENM transmission to inform the other entities of the potential threat.

ETSI defines tight requirements in terms of end-to-end latency: the standard specifies that from the moment at which the relevant information is available at source, to the moment at which the warning is presented to the driver, the measured latency cannot exceed 300 ms.

4.1.1 Functional requirements

The ICRW application relies on the performances of both originating and receiving stations, and the basic functional requirements are provided in Clause 6 of [34]. To summarize, an ICRW-compliant application is required to specify the conditions under which a possible collision is detected. Such an estimation is performed by looking at the position and movement information coming from the surrounding entities. The risk of collision can be inferred by computing the Time-to-Collision (T2C) i.e., the estimated time at which the actual collision will occur.

By monitoring the dynamics of involved entities, the ICRW application may stop issuing the collision alert: this usually happens when the ICRW user (either it can be the driver, the automatic driving assistance systems implemented in the vehicle, a cyclist or a pedestrian) takes actions to avert the collision. When the ICRW application detects a

potential collision, either inferred from received CAMs or notified through DENMs, it is required to trigger a proper warning to the user, or to activate dedicated vehicle-based collision avoidance systems.

4.1.2 Operational requirements

The operational requirements specified by ETSI for ICRW are related to security and performance aspects. In a nutshell, when designing ICRW some considerations on the possible failures should be made, so to develop an application that is completely fault tolerant. Among the possible disruptive events, there is the silent interruption of the system (caused either in transmission, in reception or in the data processing phase), the generation of DENMs not corresponding to actual collisions, or other system-level failures. In case one or more faulty behaviors are identified, the ICRW application should immediately stop its normal routines and notify the driver about the abnormal behavior.

From the performances point of view, the standard sets some minimum requirements both in position accuracy, in transmission range and in end-to-end latency. The confidence level for the position estimation should never be less than 95%, with a minimum accuracy equal or better than 2 meter. The ITS-S entities are required to implement transmission systems with a radio range, in line of sight scenarios, of at least 300 m. The standard also sets some constraints in terms of latency, by specifying the maximum processing time (i.e., the time that it takes from the moment at which a packet is received to when the alert or the automatic action are issued) to 80 ms. Moreover, whenever ICRW triggers a DENM, the DENM shall be updated with frequency 10 Hz, as long as the collision risk persists.

Finally, an ITS-S can be considered standard-compliant only if it can process at least 1000 CAMs and DENMs per second, value that corresponds to the realistic maximum amount of messages that can be transmitted in the CCH of ITS-G5 every second.

4.2 ICRW application: the Collision Avoidance Service

Assessed the importance of developing new and innovative solutions targeting the high demand for safer mobility systems, in this thesis a novel application to avoid collisions at intersections is proposed. The solution provided draws on the ETSI's ICRW specification, is called Collision Avoidance Service (CAS) and can be deployed under multiple scenarios, as well as using a number of different wireless access technologies. In particular, a couple of architectures has been designed and developed, each one possibly enabled by different communication technologies.

The reference architecture of CAS is depicted in Figure 4.1. The idea behind the system is straightforward: the road entities (vehicles, cyclists, pedestrians, etc.) broadcast

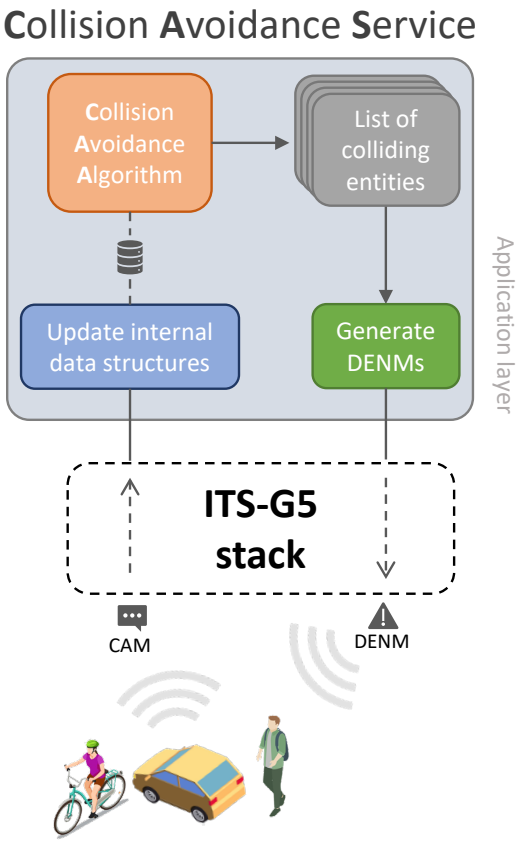


Figure 4.1: Reference architecture of the Collision Avoidance Service.

their information through Cooperative Awareness-based systems. The messages are received and processed by the entity running CAS; depending on the case, it can be a dedicated server placed behind the RSU, in a MEC server or in the cloud. Furthermore, as better explained in the next sections, the receiving entity may be directly one of the road players running an instance of the distributed version of CAS.

As soon as a CAM is received, it is processed by the lower layer of the stack, and it is finally delivered to CAS: as a first step, the system extracts the meaningful information from the message (position, speed, acceleration, heading) and updates the internal data structures holding the status of all the entities subscribed to the service. Then, the Collision Avoidance Algorithm (CAA) is run: the algorithm checks if the triggering entity is set on a collision course with any of the entities stored in the internal data structures. At the end, CAA returns the list of colliding pairs, and generates the DENMs accordingly.

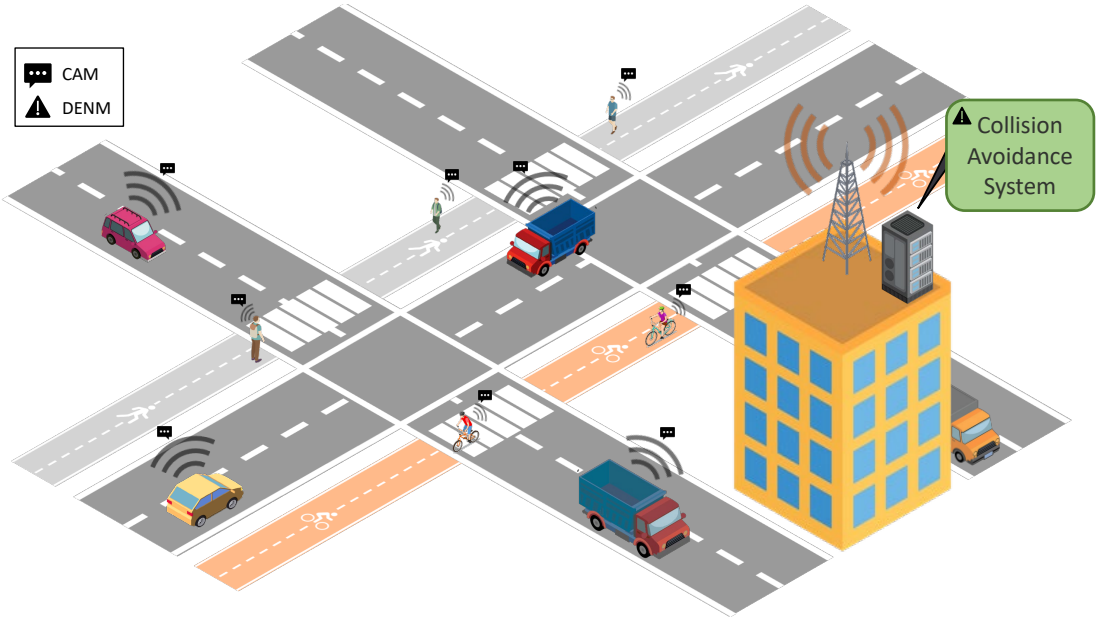


Figure 4.2: Centralized version of CAS. Vehicles and vulnerable users are connected to the network infrastructure through their devices, and the service runs in a central unit placed either in the proximity of the base station (e.g., in a MEC server) or in a cloud node.

4.2.1 Centralized solution

The first architecture proposed, depicted in Figure 4.2, is centralized; the communication is based on a client-server paradigm and requires the users (either vehicular and non-vehicular) to be connected to a central unit which runs CAS. The ITS-S collects the CAMs from the surrounding entities and pass them to the application layer, where CAS processes it and runs its internal logic.

As soon as a possible collision is evinced, the involved entities are notified through DENMs, and the receiving system is then in charge of implementing one of the following actions: *(i)* to notify the driver (or the pedestrian, the cyclist etc.) of the upcoming threat by implementing a warning system (visual, acoustic, haptic, or a combination of the three); *(ii)* implement an automatic evasive action (when possible).

The centralized solution falls into the category of V2I, V2N and V2P applications (depending on where CAS is deployed and on the type of road players involved), and enables the protection against vehicle-to-vehicle and vehicle-to-vulnerable-user collisions, with the vulnerable users that can be connected to the network infrastructure either through normal smartphones or through dedicated devices (e.g., wearable, smart bands, smart watches etc.).

The access technologies enabling this solution are both wireless-LAN-based and cellular-based: the central unit running CAS can be either deployed in a RSU station,

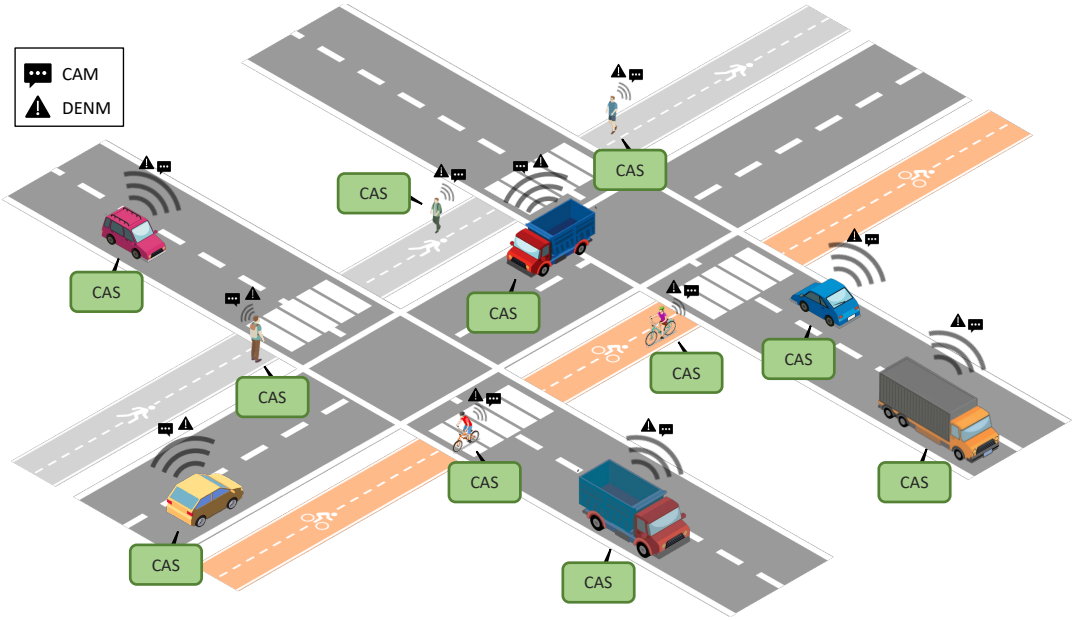


Figure 4.3: Distributed version of CAS. Vehicles and vulnerable users exchange CAM messages among themselves using a direct communication protocol. Each entity runs its own CAS service and, based on the configuration adopted, DENMs are generated and eventually transmitted to the involved entities.

thus communicating with the users through IEEE 802.11p, as well as in the cellular network (i.e., reachable using LTE or 5G devices), by adopting edge-based solutions (using Multi-access Edge Computing - MEC - systems) or cloud-based solutions, with the latter disadvantaged for end-to-end delay issues.

4.2.2 Distributed solution

The second architecture proposed, depicted in Figure 4.3, is distributed; the CAS system, in this case, is spread among the users (either vehicular and non-vehicular). The users is not connected to a central unit: each entity is in charge of receiving and processing CAMs, of running the CAA logic, and of independently assessing the collision risk.

As soon as a possible collision is found, the strategy can be either to notify the involved user through appropriate DENMs, or (assuming that all the players in the scenario are running an instance of the CAS system) to only notify the current user of the possible threat (thus, by adopting a so-called *silent strategy*).

The distributed architecture comes with its pros and cons: as an advantage, being the system directly developed at the user side, it can rely on a general reduced end-to-end delay, with the system that can therefore gain something in terms of responsiveness. On the other hand, it requires non-trivial communication, computational and power capabilities to be implemented directly into the end-user devices. This problem can

be negligible in a fully-vehicular environment, where the aforementioned capabilities can be easily integrated into the vehicles' control unit. Problems may arise when the system is instead developed into smartphones, or dedicated devices, that may lack in computational capabilities or may suffer of battery drains, due to the resource demanding operation of both communication and computational requests of CAS.

The distributed solution falls into the category of V2V and V2P applications, enables the protection against vehicle-to-vehicle and vehicle-to-vulnerable-user collisions, with the vulnerable users that can be connected to the distributed network through dedicated devices. Even in this case, the enabling access technology can either be wireless-LAN-based or cellular-based: the users can exchange V2X messages using C-V2X sidelink channel (both in transmission mode 3 or mode 4) or using 802.11p OBUs.

4.2.3 The Collision Avoidance Algorithm

The core of CAS is the Collision Avoidance Algorithm (CAA), which aim is to determine whether two entities are set on a collision course or not. Each time the service receives a CAM from a user (being it a vehicle, or a vulnerable user), it generates the future trajectory by relying on the position, heading, speed and acceleration information included in the current CAM. Each generated trajectory is then analyzed and compared with the others (based on the previously received CAMs), in order to determine if any pair of vehicles or vehicle-vulnerable user is likely collide in the future. To this end, two main parameters are computed:

1. Space-to-collision (S2C), i.e., the minimum distance that will be reached between the entities under test, assuming that they will follow the projected trajectory.

2. Time-to-Collision (T2C), i.e., the time that it takes to reach the above S2C between the entities under test.

The system detects a possible collision between two vehicles when both S2C and T2C parameters are below some thresholds. The value of such thresholds can be dynamically calculated: the S2C threshold depends on the size of the involved entities, while the T2C threshold depends on their speed, deceleration capabilities, mass and also on external factors such as the road surface status. Once all pairs of entities have been analyzed, the algorithm returns the list of pairs identified to be on a collision course.

If one or more of such pairs are identified, the underlying system generates the needed DENMs, each including the type of hazard, the detection time, and the point of collision. DENMs are then forwarded to the vehicles involved in the detected collision or, in case of the aforementioned V2V *silent strategy*, directly forwarded to the upper layers where the HMI will show the corresponding alert. The CAA algorithm and the selection of T2C and S2C thresholds are detailed in Appendix B.

The structure of a DENM generated by CAS is reported in Table 4.1. Beside the basic ITS PDU header, present in every V2X message standardized by ETSI, a DENM generated by CAS contains a Management and a Situation Container. The first is used by CAS

Section	Field	Value - Info
ITS PDU header	Protocol version	0x01 - Version of the protocol
	Message ID	0x01 - Message ID for DENM
	Station ID	variable - ID of the server (in centralized scenarios) or ID of the generating entity (in distributed scenarios)
Management Container	Action ID	variable - ID of the collision event
	Detection time	variable - Time at which the collision was first identified
	Reference time	variable - Time at which the DENM was generated
	Event position	variable - Position where the collision will take place
	Station type	variable - Type of the generating entity
Situation Container	Information quality	informationQualityHighest - Accuracy of the prediction (based on positioning errors)
	Cause Code Type	collisionRisk or collisionWithCyclist or CollisionWithPedestrian - Type of collision
	Sub Cause Code Type	variable - ID of the other entity (with respect to the message's recipient)

Table 4.1: Structure of a DENM generated by CAS

to include the general information regarding the collision event: detection time, event position, and type of the generating entity. The latter is used to specify that the DENM corresponds to an event of type *collision*, and to specify which is the other entity involved (with respect to the message's recipient). In case the message is broadcasted, this field contains the ID of the recipient, while an additional field (namely, the *linkedCause* field) contains the ID of the other entity involved.

4.3 System validation through simulations

CAS has been developed on top of MS-VAN3T, the vehicular network simulation framework based on ns-3 and SUMO introduced in Chapter 3. Therefore, the system was tested and validated in a scenario where the information needed to run the algorithm is delivered through standard-compliant V2X messages. Depending on the communication technology established, CAMs and DENMs are either forwarded to a central server, or directly distributed among the road players. In the next sections, the CAS system is evaluated by extracting some KPIs from the simulations campaigns. In particular, the analysis focuses on:

1. The percentage of collision detected by the system.

2. The T2C at the first DENM, namely the time that separates the first DENM reception and the actual collision.
3. The distance between the vehicle and the collision event when the first DENM is received.
4. The delay from the moment at which a DENM is created to the moment at which it is received.

As previously mentioned, CAS was originally designed to support purely vehicular scenarios; with little changes, however, it is possible to extend its coverage to support also vulnerable users such as pedestrian and cyclist. Therefore, the validation of CAS has been made by first evaluating a scenario where only vehicles are present, then by including vulnerable users in the loop.

All the simulation are performed by trying to stress the CAS system as much as possible, and by trying to put it in the worst possible scenario. The speed of the vehicles approaching the intersection is voluntarily kept very high (up to 130 km/h): although this kind of behavior is unrealistic, especially in unregulated urban intersections, it must be considered that if the CAS system is able to identify and prevent collisions in such extreme conditions, even more so it will be able to protect the road users in normal situations, where the speeds are lower.

4.3.1 Purely vehicular scenario

The map used to test the system represents a common urban scenario, with a central road crossed by two secondary roads; Figure 4.4 shows the layout of the scenario, taken

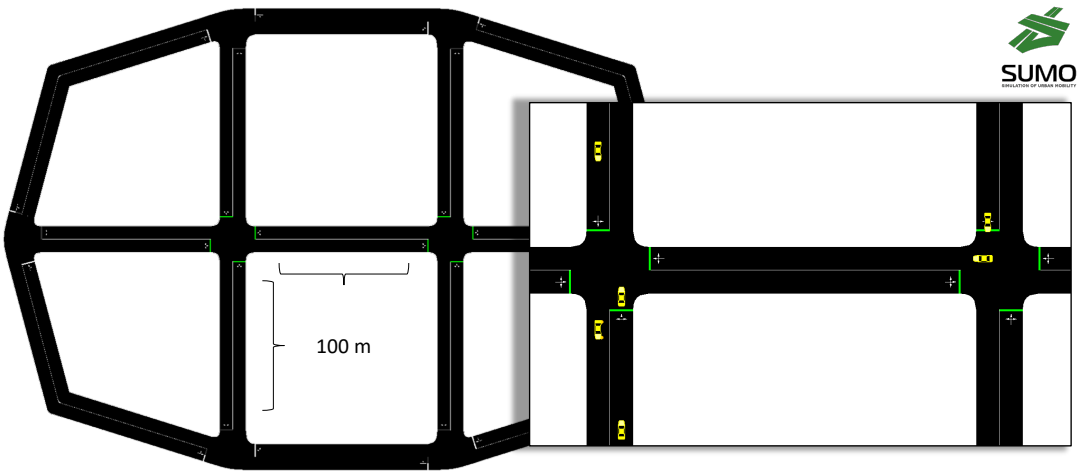


Figure 4.4: Map used to test and validate CAS: an horizontal segment crossed by two vertical roads. Externally, each dead end is connected through a ring, so that the vehicles can drive endlessly.

from the SUMO GUI. In total, the horizontal axis is 300 m long, while the vertical is 200 m long. Each vehicle is generated randomly in one of the segments, and is configured to drive endlessly in the map. Whenever it reaches one of the external intersections, each vehicle is configured to randomly pick one of the possible directions. Whenever it reaches one of the internal intersections, instead, each vehicle is configured to go straight.

In principle, SUMO is built to avoid collisions: this means that in a normal configuration the vehicles approaching at the central intersections will avoid each other, by implementing the basic precedence rules. Thus, without any further modification it is be impossible to validate CAS, since no actual collision ever occurs. For this reason, the two central intersection are regulated by a modified traffic light system, configured to show at the same time the green light to all incoming directions. With this simple, yet effective trick, SUMO lets the vehicles approach the crossings at the maximum speed, so that they will not have the time to avoid each other once inside the intersection.

Simulation results - centralized solution

The simulations analyzed in this section show the performance of the centralized version of CAS. The system is evaluated by increasing the total number of vehicles present in the scenario, from 10 to 30; additionally, the maximum speed of the vehicles is gradually increased, from 25 km/h to 130 km/h. All the aforementioned configurations are tested by changing the underlying access technology: in one case, a 802.11p-like communication is established among the vehicles, thus all the vehicles are equipped with 802.11p OBUs and the CAS system runs in a server connected to the RSU. In the other case a LTE network is established, where the vehicles are equipped with UEs and CAS runs in a server connected to the EPC. In this case, the server hosting the centralized CAS logic will be connected to the PGW and SGW block, which is in turn connected to the eNB through the S1-U interface. Although the implementation does not exactly match that of a Multi-Access Edge Computing (MEC) node placed at the edge of the network, in this case the low traffic in the EPC, added with the low latency artificially configured in the point-to-point interface between the server and the EPC, creates an environment that closely mimics a network with a MEC node running the CAS system.

Each simulation lasts 3600 seconds of simulated time.

For all the results presented in this part, the system was configured to deliver DENMs for collisions happening no more than 10 seconds in the future; therefore, the T2C threshold is fixed to 10 seconds. By using this large gap it is possible, at reception side, to filter out all the messages received for low collision risks. Moreover, since each DENM contains the position of the collision event (that is computed by CAA), a vehicle receiving a DENM may internally implement its own evasive strategy, that can be based on its specific measurements (about speed, vehicle's mass, road surface conditions, etc.) and on the perceived neighbors status. For what concerns the S2C threshold (i.e., the distance limit that two entities should reach under which a DENM is triggered), the

calculation is dynamic with a direct dependence on the entities' size (as detailed in Appendix B). The value of γ (the safety margin added to the S2C threshold to tackle eventual positioning errors) is set to 1 meter.

Technology	Parameter	Value
	Total number of collisions	6971
802.11p	Collision detected	6971 (100%)
LTE	Collision detected	6971 (100%)

Table 4.2: Collision detected by the system in the centralized solution

The first glimpse on the system effectiveness can be seen in Table 4.2, which shows the total number of collisions happened during all the simulations: CAS was able to detect in advance 100% of the dangerous situations, no matter the total number of vehicles or their maximum speed. As a matter of fact, after the analysis on the system performances, it was possible to notice that the increase of vehicles has no effect on any of the studied parameters. For this reason, the results shown in the next plots are all shown as a function of the maximum speed reachable by the vehicles.

The plot in Figure 4.5 shows the time gap between the reception of the first DENM and the corresponding collision, as a function of the maximum speed set on the vehicles. The access technologies tested (802.11p and LTE) seem to perform similarly, with no significant differences in the measured values; these results suggest that both the

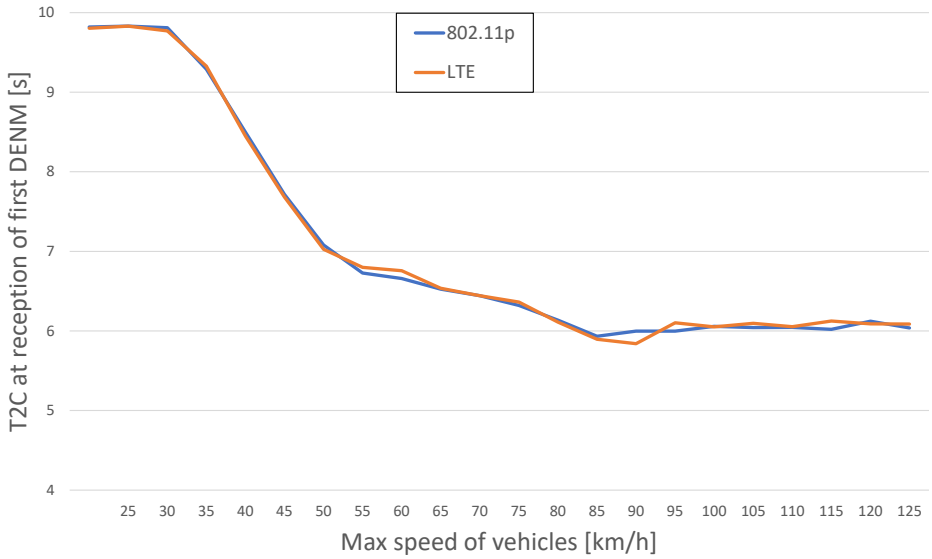


Figure 4.5: Time that separates the first DENM reception and the corresponding collision, as a function of the max speed of the vehicles. The results when deploying a 802-11p-like communication are in blue, while in orange the results with a LTE network in place.

technologies are able to guarantee the basic delay requirements to enable this kind of service.

At the same time, it is possible to notice that with the increase of the vehicles' maximum speed, the measured T2C tends to decrease rapidly, until 50 km/h; then, the value seems to stabilize around 6 seconds. The reason for that behavior are multiple: first of all, by increasing the speed, the distance traveled between two CAMs increases accordingly. This phenomenon is actually mitigated by the dynamic CAM generation implemented in the CA basic service but, however, at low speed still produces visible effects. At higher speeds, instead, the road topology plays an important role: since the vehicles are configured to drive following realistic behaviors and patterns, it is very unlikely that they reach their maximum speed, especially considering that the longest straight road in the urban scenario proposed is 300 meters. For this reason, the average speed kept by the vehicles stabilizes around a certain limit and, consequently, the time that separates the first DENM reception and the corresponding collision stabilizes accordingly.

Figure 4.6 shows the distance between the vehicles and the collision event at the reception of the first DENM, as a function of the maximum reachable speed; also in this case the results comprehend the case in which a 802.11p and LTE communication are deployed. In the same plot, the red dashed line (referring to the red vertical axis on the right) shows the speed of the vehicles at first DENM reception. As in the previous case, the plot suggests no relevant difference when adopting one or the other access

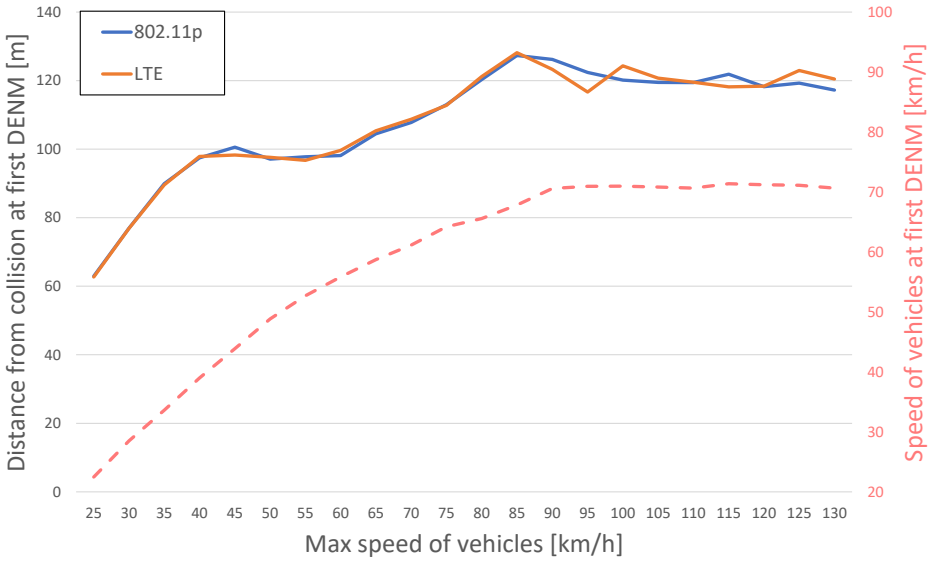


Figure 4.6: Distance that separates the vehicle and the collision position identified by CAS at the reception of the first DENM, as a function of the max speed of the vehicles. In the same plot, referred to the vertical axis on the right, the speed of the vehicles at reception of the first DENM is plotted using the red dashed line.

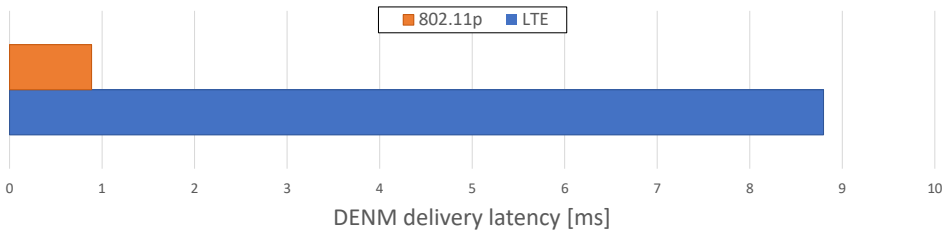


Figure 4.7: Delay computed from when a DENM is generated, to when it is received.

technology. The distance between the vehicle and the collision event is computed using the Haversine formula [44]: one point is the vehicle latitude and longitude (generally returned by the Vehicle Data Provider, which in this case is the TraCI interface), while the other point is the *Event Position* field in the *Management Container* of the received DENM. The studied parameter increases coherently with the speed measured at first DENM reception; the increase is constant for low speed, and is flattened as the speed of the vehicles reaches the limit imposed by the road topology.

The last results for this set of simulations are shown in Figure 4.7 and give some additional indications on the access technology performances from the point of view of the delay. The two bars represent the time gap between the moment in which a DENM is generated to when it is received at the recipient's application layer. As predictable, given the results shown in Chapter 3, 802.11p outperforms LTE in terms of delivery delay: it reaches an average delay below 1 ms, while with LTE (considering the 5 ms of fixed delay between PGW and the server running CAS) the delay is around 9 ms. Interestingly, this result does not seem to be reflected in any of the application-related performances, with the CAS system that works properly even when a slower access technology, such as LTE, is used to communicate.

Simulation results - distributed solution

The simulations analyzed in this section show the performances of the distributed version of CAS. Also in this case, the system is evaluated by increasing the maximum speed reachable by the vehicles, by increasing the total number of vehicles in the scenario, and by changing the underlying access technology. The two options tested are: *(i)* 802.11p in V2V configuration (therefore, the vehicles directly exchange CAMs and leverage on the enclosed information to run the CAS logic) and *(ii)* C-V2X, with all vehicles equipped with PC5-enabled devices, using transmission mode 4 to broadcast CAMs and internally run the CAS logic.

When using 802.11p, the OBUs are configured to transmit at 6 Mb/s; in C-V2X the devices are configured to reselect the resource with probability P_r = 0.8 (to minimize the delay, as shown in Section 3.4.3), with the resource reservation interval fixed to 20 ms.

Tecnology	Version	Parameter	Value
		Total number of collisions	6971
802.11p	Normal	Collision detected	6971 (100%)
		First DENM type [internal / external]	35% / 65%
	Silent	Collision detected	6971 (100%)
C-V2X	Normal	Collision detected	6971 (100%)
		First DENM type [internal / external]	39% / 61%
	Silent	Collision detected	6971 (100%)

Table 4.3: Collision detected by the system in the distributed solution. Additionally, the table reports an analysis on the type of the first DENM received for each collision adopting the normal strategy, showing the percentage of internal and external DENMs.

Additionally, the distributed version of CAS was tested in two different configurations: the normal configuration, where as soon as a possible collision event is identified, a DENM is created and transmitted to the involved entities, and the so called *silent* configuration, where every vehicle is in charge of identifying only the collision events involving itself. In the second version, as soon as a collision is identified, the entity does not generate a DENM to be transmitted in the channel, but it only issues an internal alert. This information can be used by the vehicle to simply show a warning in the in-vehicle HMI, or to possibly trigger some automatic collision avoidance system.

The internal CAS parameters are set as in the centralized version: T2C threshold fixed to 10 seconds and the S2C dynamically calculated using the dimensions of the vehicles included in each CAM.

Table 4.3 shows that also in the distributed version CAS has a 100% of precision in detecting future collisions, no matter the underlying communication technology. The mobility traces used are the same as in the centralized version, therefore the final number of collisions are exactly the same. The Table reports the percentage of collision detected for 802.11p and C-V2X simulations, as well as an analysis on the first DENM received for each collision event when adopting the normal version of the system (i.e., the one in which the vehicles generate and send DENMs when an external collision is identified). The same value is not reported for the *silent* version, since in that case all the collisions trigger only *internal* DENMs.

Interestingly, the results suggest that by adopting a strategy where every entity checks for itself *and* for the others, it is possible to increase the system's responsiveness. Indeed, by adopting the normal strategy it has been computed that most of the times the first DENM referring to a collision is not internal, but is generated by some external entity which, based on the received information, identifies the dangerous situation and triggers a DENM. This phenomenon is more evident with access technologies guaranteeing a low delivery delay (such as 802.11p), because a lower delay is always translated in a fresher information, and in a more accurate trajectory predictions. Therefore, the first conclusion is that by adopting the *silent* version, the system performances degrade and this degradation is correlated to the delivery delay of the adopted access technology.

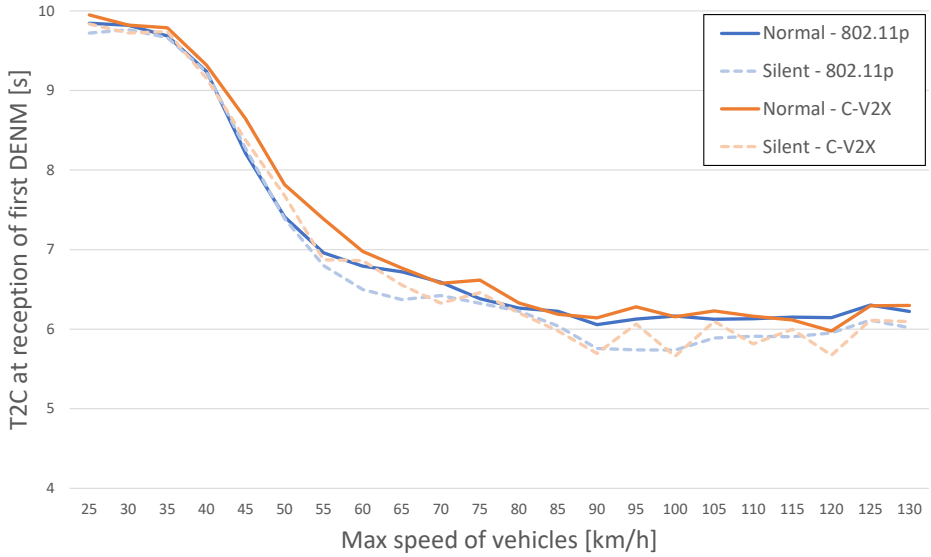


Figure 4.8: Time that separates the first DENM reception and the corresponding collision, as a function of the max speed of the vehicles. The results when deploying a 802-11p-like communication are in blue, while in orange are the results with a C-V2X communication in place. The dashed lines refer to the corresponding results with CAS in *silent* mode.

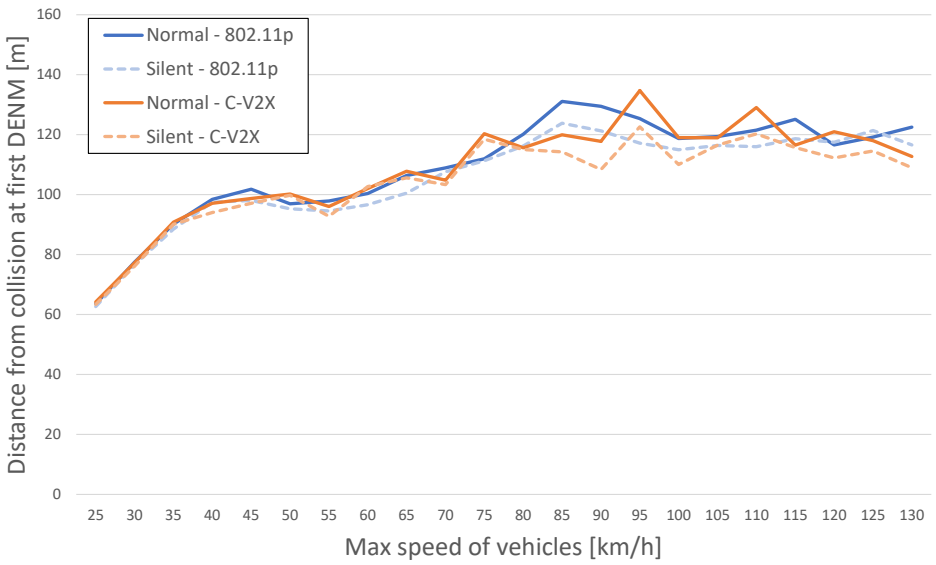
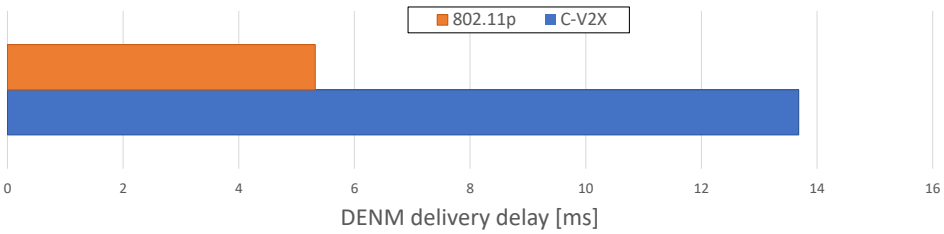
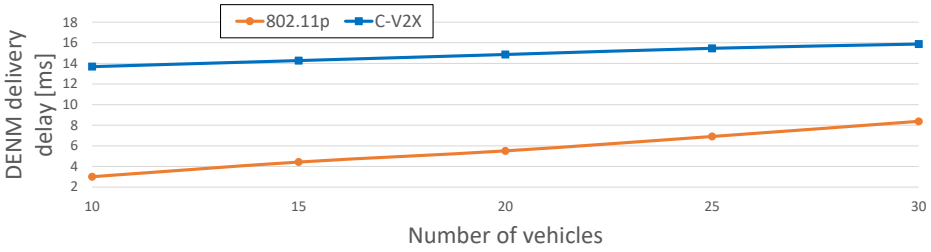


Figure 4.9: Distance that separates the vehicle and the collision position identified by CAS at the reception of the first DENM, as a function of the max speed of the vehicles. The dashed lines refer to the corresponding results with CAS in *silent* mode.



(a) Delay computed from when a DENM is generated, to when it is received.



(b) DENM delivery delay when increasing the number of vehicles.

Figure 4.10: DENM deliver delay analysis for distributed version of CAS.

The above considerations are confirmed by looking at Figure 4.8. The plot, referred to the time gap between the reception of the first DENM and the corresponding collision, shows that in general the normal strategy guarantees a T2C higher with respect to the *silent* strategy. Also in this case it is possible to confirm that having an external entity monitoring the vehicles' status makes the system faster in reacting to dangerous situation; having a higher T2C, indeed, allows the driver (or the autonomous system) to increase the time available to implement an eventual evasive maneuver.

In Figure 4.9, the distance between the vehicles receiving a DENM and the corresponding collision event is plotted. As in the centralized solution, these results suggest that the studied distance increases with the maximum speed, following a trend that seems to stabilize for speeds greater than 90 km/h. Also in this case the *silent* version is less performative, and in general it provides a system that informs the involved entities when they are closer to the collision, with respect to the normal version.

Finally, Figure 4.10a depicts the delivery delay when using 802.11p and when using C-V2X (only in normal mode, since in *silent* mode no DENM are generated and transmitted). It is interesting to notice that, using the same setting as the centralized simulations, now 802.11p experiences a greater delay (more than 5x): this is because the traffic in the channel, especially at the moment in which DENMs are generated, is increased massively when adopting the distributed version of CAS. In the centralized version, DENMs are generated only by one entity, namely the server running CAS. In the distributed service, as soon as two vehicles are set on a collision course, every surrounding entity running an instance of CAS generate and send the corresponding

Figure 4.11: Typical situation in which the *silent* version of CAS does not identify a collision: vehicles A and B are going to collide, but the building in the middle attenuate their radio connection. Vehicle C does not help, because DENM transmission is disabled.

DENM. This phenomenon is less visible in C-V2X, due to the fact that the SPS mechanism can manage a high number of contending nodes and to keep a stable delay. It is instead more visible with 802.11p, which adopts CSMA-CA as distributed channel access scheme. In particular, Figure 4.10b shows this effect, as a function of the total number of vehicles present in the scenario. In 802.11p, the DENM delivery delay is around 3 ms with 10 vehicles in the scenario, and increases up to 8.5 ms when 30 vehicles are present. In C-V2X instead, the increase in the number of vehicles is only translated in an increase of 2 ms in the experienced delay (from 14 to 16 ms).

To conclude, even though the results presented in this section suggest that the normal version of CAS is preferable to its *silent* version, it is worth mentioning that the latter brings several advantages from the point of view of the load in the channel, since no DENM is actually transmitted.

At the same time, however, there still remain some scenarios in which the normal version may have fundamental advantages. An example is shown in Figure 4.11, where two vehicles (A and B) are approaching an intersection. A and B are in a non-line-of-sight situation and the buildings in the middle attenuate their communication so that the cannot "hear" each other. Additionally, there is a third vehicle (C), which is in line-of-sight with both of them: this vehicle can receive CAMs from both A and B, and can

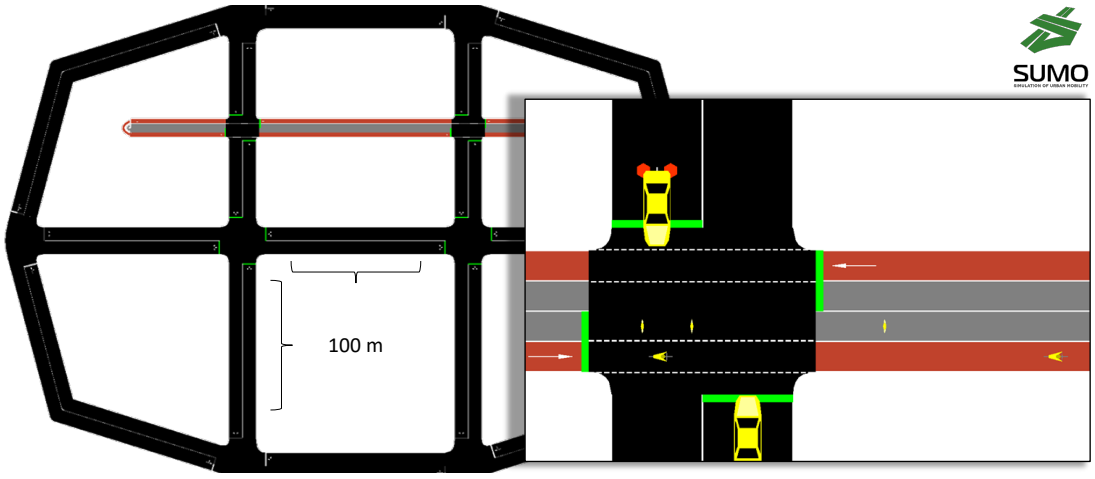


Figure 4.12: Map used to test and validate CAS with vulnerable users: the two vertical roads are crossed by a pedestrian and a bike lane, represented respectively in grey and in orange.

warn them in case of collision. If the *silent* version of CAS is deployed, C does not send them a DENM to signal the collision, with the result that the two vehicles will crash at the intersection.

Other considerations can be made by comparing the system performances when deploying the centralized or the distributed solution. From the point of view of the time that separates the first DENM and the actual collision, it can be seen that the distributed solution (due to the lower delay introduced by the direct communication among the vehicles) increases the system performances when deployed in the normal version; also the distance from the collision at the reception of the first DENM is slightly higher when using the distributed solution. From a practical point of view, instead, the centralized solution comes with several advantages. First of all, it is much easier to be implemented since it requires only to instantiate a single central service that collects all the CAMs coming from certain area (for example from a crossing); at the same time, the distributed version requires all vehicles to embed an instance of CAS, which, due to the heterogeneity of brands and types of vehicles, is not something easy to be achieved. Moreover, the distributed solution, when DENM broadcasting is enabled, introduces a significant overhead in the channel load, due to the flood of messages that follows every possible collision detected.

4.3.2 Vehicles and vulnerable users

The map used to test and validate CAS in case of vulnerable users is shown in Figure 4.12. The road topology is the same as in the purely vehicular case, but in this extended

configuration the two vertical roads are crossed by a pedestrian and a bike lane, represented in SUMO respectively with grey and orange lanes. Bikes and pedestrians follow the same rules of vehicles, with the exception that they can only go back and forth in their respective lanes; as the vehicles, they are randomly generated in one of the dedicated segments and configured to endlessly traverse the map. In this way, the system is stressed every time a vehicle and a vulnerable user cross their paths.

The two additional crossings are configured to force collisions as previously explained: also in this case, the traffic lights are configured to show green to all incoming directions, so that the entities involved do not have the time to avoid the collision once entered the crossing.

Simulation results

In this case, the system is analyzed by increasing the total number of vehicles and their maximum speed (as in the purely vehicular scenario), and by deploying it in the centralized and distributed configuration, respectively using LTE and 802.11p as communication technologies. Note that, although the integration of 802.11p-based radios on wearables or smartphones seems a bit unrealistic, for the sake of completeness this work will compare all the technologies available for V2X communication, including 802.11p for pedestrians and/or cyclists.

The total number of vulnerable user is fixed to 10 pedestrians and 10 cyclists. When the distributed version is in place, the entities are configured to broadcast DENMs as soon as they identify possible collisions (i.e., the *silent* version is disabled). The analysis focuses only on the DENMs exchanged for collisions between vehicles and vulnerable users (i.e., the analysis on the collisions between vehicles is not reported).

For all the results presented in this part, the system was configured to deliver DENMs for collisions happening no more than 10 seconds in the future. Also in this case, having a large gap (in terms both of time and space) from the first DENM to the actual collision, allows the receiving system to filter out all those DENMs received for low risk, and to eventually perform an evasive maneuver. The S2C threshold is instead dynamic and computed as detailed in Appendix B, with the safety margin γ set to 1 m.

By looking at the results in Table 4.4, it is possible to evince that the number of collisions involving bicycles is 10 times higher with respect to the number of collisions involving pedestrians. This is mainly due to the different speed at which pedestrians

Version	Technology	Parameter	Value
		Total number of collisions (Pedestrians)	561
		Total number of collisions (Bicycles)	5398
Centralized	**LTE**	Collision detected (Pedestrians)	561 (100%)
		Collision detected (Bicycles)	5398 (100%)
Distributed	**802.11p**	Collision detected (Pedestrians)	561 (100%)
		Collision detected (Bicycles)	5398 (100%)

Table 4.4: Collision detected by the system with vulnerable users in the loop.

and bicycles move around the map: the firsts are configured to reach a maximum speed of 2 m/s, the latter of 7 m/s (corresponding to ~25 km/h). Therefore, also due to how the crossings are configured, it will be easier for a pedestrian to stop and avoid a collision with a vehicle. At the same time, the results in Table 4.4 confirm the effectiveness of CAS: in both configurations (centralized and distributed), the system was able to detect all the collision events.

In Figure 4.13, the results for the T2C at first DENM reception are shown. It is evident that the lower the speed at which the entity is moving, the higher will be the time that separates the first DENM and the actual collision. It is also possible to notice that the lower number of samples collected for vehicle-to-pedestrian collisions makes the results way less smooth with respect to the vehicle-to-bicycles or vehicle-to-vehicle case, in which thousands of situations are analyzed.

By looking at the DENMs received by bicycles (the orange and red lines in Figure 4.13), the increase of the vehicles' speed is translated to a decrease in the measured T2C at first DENM. This is verified for speeds up to 70 km/h, then the value stabilizes around 5 seconds. The plot confirms also that, on average, the distributed version of the algorithm brings advantages in term of T2C at first DENM, as already analyzed in previous simulations.

For what concerns the distance between the entities and the actual collision at the

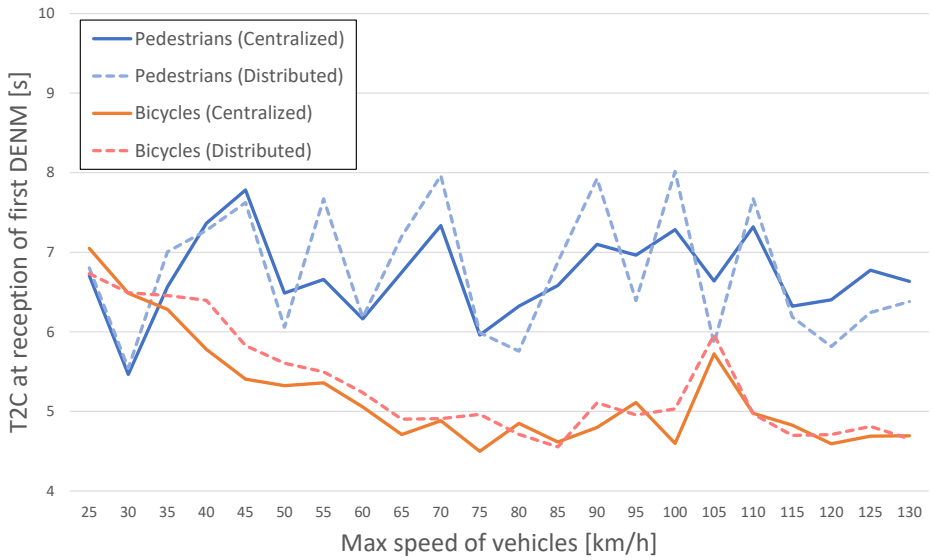


Figure 4.13: Time that separates the first DENM reception and the corresponding collision, as a function of the max speed of the vehicles. The red and orange lines show the analysis of the DENMs received by pedestrians, in blue the analysis of DENMs received by pedestrians. The solid lines correspond to the cases in which the centralized version of CAS is deployed, the dashed to the cases in which the distributed version of CAS is deployed.

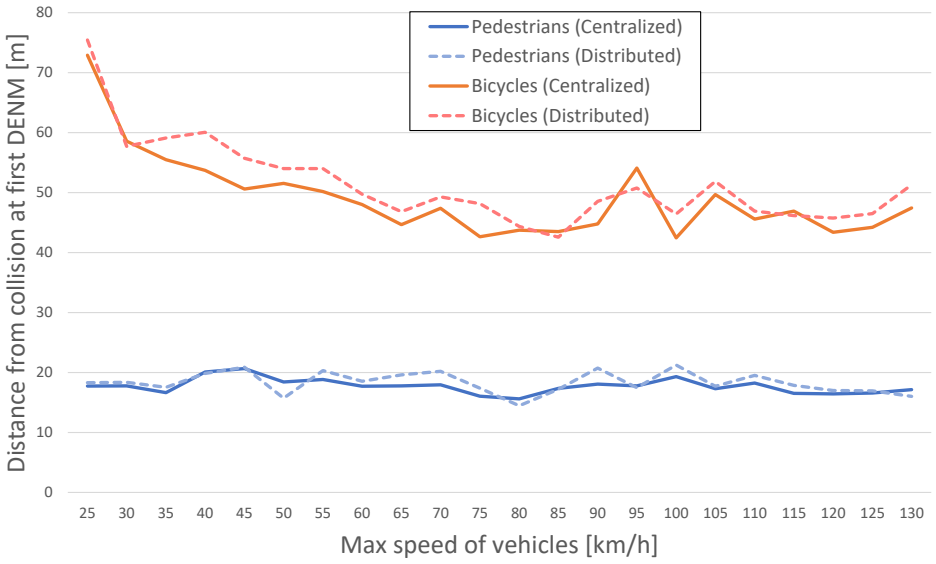


Figure 4.14: Distance that separates the entity and the collision position identified by CAS at the reception of the first DENM, as a function of the max speed of the vehicles. The dashed lines refer to the corresponding results with CAS deployed in the distributed version.

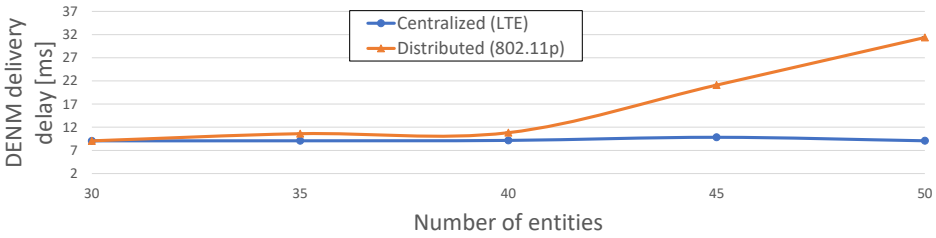


Figure 4.15: Delay computed from when a DENM is generated, to when it is received.

reception of the first DENM, shown in Figure 4.14, also in this case the different dynamics of pedestrians and bicycles produce different results. The pedestrians, which move slower with respect to bicycles, are warned on average 18 meters before the actual collisions, no matter the maximum speed held by vehicles. The results involving bicycles, instead, exhibit a certain correlation with the vehicles' maximum speed: the distance, initially around 70 meters, decreases and stabilizes around 45 meters.

Although it seems useless to warn a pedestrian with such a big advance, the application that receives the DENM can be safely configured to trigger an alarm only when the measured distance is below a certain threshold (and this is valid for all the users subscribed to CAS).

The last analysis for CAS including vulnerable users, reported in Figure 4.15, shows the delay experienced when transmitting/receiving DENMs. The results are grouped

by CAS version: the blue line corresponds to the centralized version, where LTE is the reference access technology; the orange line is instead referred to the distributed version using 802.11p. The results are plotted as a function of the number of entities running CAS: the vehicles increase from 10 to 30 while there are always 10 pedestrians and 10 bicycles.

This plot highlights the advantages of the centralized solutions in terms of delay; the high number of simultaneous transmission of DENMs when a collision is identified creates effects on the experienced delay (in the distributed version of CAS) starting from 40 entities. Then the delay increases almost linearly with the number of entities up to 32 ms, value that is measured with 50 entities (30 vehicles, 10 pedestrians and 10 bicycles). The same effect does not hold in the centralized version of CAS: in this case, indeed, the DENMs are only generated and sent by the server, thus the overhead of the system, in terms of traffic offered on the channel, is extremely reduced.

The version of CAS presented in this thesis is thought to only leverage on the information exchanged among the road players through V2X sensors. In a real-world scenario, it is likely that any user subscribed to a system like CAS, will also have other active safety services, based on other sensors such as cameras, radars, or LIDARs. Therefore, the information generated with CAS will just represent a small contribution to the bigger decision-making system of an hypothetical future autonomous vehicle.

4.4 CAS as an enabler for autonomous driving systems

In the previous sections it has been demonstrated how it is possible to use CAS to offer protection to vehicles and to vulnerable users, such as pedestrians and cyclists. As it is presented, the implementation of CAS builds a passive safety layer that, leveraging the exchange of DENMs, is used to signal the collision risk at intersections. It is, however, hard to figure out *how* this information can be actively used by all the road players involved, to effectively avoid collisions. For example, a pedestrian receiving a DENM may have an alert issued in its smartphone, or in its wearable device (e.g. a smart watch), and may take a closer look to the road before crossing. The same consideration holds for cyclists and scooter drivers, which may react by promptly braking to avoid the collision. For vehicles, instead, an additional degree of protection (rather than the human-based reaction to an alert issued in the HMI), can be played by autonomous driving system. Those systems, categorized by SAE in 5 increasing levels of automation [88], may directly take the full control of the vehicle in its acceleration, braking and steering maneuvers.

Therefore, the next sections are devoted to the presentation of a system that leverages the CAS information to actively avoid the collisions. The system rationale drafts on the idea of the virtual safety-shield built around each vehicle, introduced by ETSI in [34]. With this safety shield, whose size varies according to the kinetic properties of the

vehicle around which it is built, it is possible to determine the gravity of the collision event included in the DENM, and to take actions accordingly.

4.4.1 Virtual safety shield

The virtual safety shield is a circular area that each vehicle builds around itself. It helps assessing the gravity of the events carried by DENMs, and eventually contributes to the decision making process that the vehicle's control unit undergoes to avert the collision. The size of the virtual safety shield is a direct function of the instantaneous speed and acceleration of the vehicle, and in real scenarios may also include information about the vehicle's mass, deceleration capabilities, road surface conditions etc.

Figure 4.16 depicts the virtual safety shield that the vehicle A creates around itself. In that particular situation, A and B are supported by a centralized CAS, which detects the dangerous situation and broadcasts a DENM containing the coordinates of the collision event. At this point, the receiving entities are in charge of assessing the position of

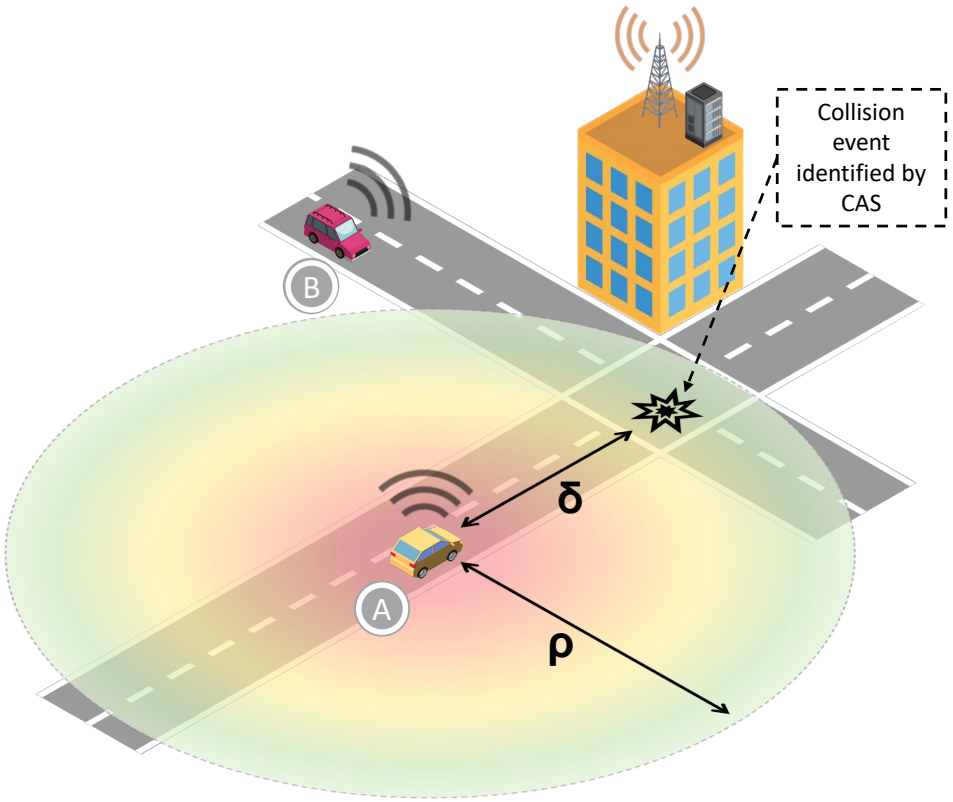


Figure 4.16: The circular area represents the virtual safety shield built by vehicle A; its size depends on the vehicle A's internal dynamics, but also on external factors. In this particular case, A is in collision course with B, and the server running CAS generated a DENM reporting the exact position of the future collision.

the collision event: if inside the virtual safety shield, then the evasive maneuver takes place.

The radius of the virtual safety shield is computed as follows:

$$\rho = V_0\tau + \frac{1}{2}a_0\tau^2 \tag{4.1}$$

Where V_0 and a_0 are respectively the speed and the acceleration at the moment in which the safety shield is generated, while τ is a value that determines the size *in time* of the shield; the higher the value of τ, the bigger the size of the virtual safety shield radius (and vice-versa). Selecting a small value for τ translates into a system that reacts too late to dangerous situations, making it impossible for the vehicle to implement any evasive strategy. On the contrary, selecting a big value for τ may end up in a system which overreacts and that extremely reduces the average speed of the vehicles supported by CAS, mining the satisfaction of the end user.

4.4.2 Reaction to DENMs generated by CAS

The implementation of this system starts from a simple concept: two vehicles that are set on a collision course do not have necessarily to completely stop to avert the collision. In principle, it is not even necessary that both of them take action, since by simply changing the trajectory of one of the two, the collision is avoided equally (and in a more efficient way). So, starting from this assumption, the vehicle's system builds the aforementioned virtual safety shield of radius ρ. As soon as a DENM corresponding to a collision event is received, the "*Event Position*" field is extracted; this field contains the position (in longitude and latitude) of the collision event, as computed by CAS. The receiving entity then computes the distance δ separating it from the actual collision, using the Haversine formula. Then, it uses ρ and δ to compute a coefficient λ, that is used to eventually scale down its speed. The calculation of λ is reported in Equation 4.2.

$$\lambda = \begin{cases} \delta/\rho & \delta < \rho \\ 1 & \delta \geqslant \rho \end{cases} \tag{4.2}$$

Basically, if the collision event is *inside* the safety shield, the value of λ represents (using a value between 0 and 1) the position of the collision event inside the safety shield. The further it gets from the center of the safety shield, the more λ becomes 1. The closer it gets to the center, the closer the λ value gets to zero. If instead the collision event is *outside* the safety shield (i.e., when $\delta \geqslant \rho$), λ assumes the value 1.

At this point, the receiving vehicle adjusts its own speed by multiplying it to the coefficient λ:

$$V_1 = \lambda \times V_0 \tag{4.3}$$

Where V_1 is the new speed, while V_0 is the speed that the vehicle had when it received the triggering DENM.

Additionally, to avoid any possible traffic congestion generated by values of λ tending to 0, the minimum speed set by this last equation is capped to 10 km/h. This is something that should also be considered in realistic scenarios: when the entities speed is very low, it is more efficient to let the driver override any autonomous driving systems, or to delegate the decision process to different sensors like cameras, radars or lidars, which have the advantage of having a clearer perception of the surrounding environment. Moreover, to avoid that two vehicles on a collision course follow the same deceleration profile (situation that inevitably ends in a collision), each time λ is added of a random uniform variable extracted from -10% and $+10\%$ of its value.

4.4.3 Evasive maneuver pseudocode

The entire evasive maneuver is coded in Algorithm 1. The algorithm is divided in two distinct procedures: the first, called "Speed Override", is in charge of computing the values of ρ, δ and λ and to adjust the speed of the vehicle accordingly. The latter, called "Distance Check", monitors the position of the vehicle with respect to the collision event: as soon as it starts to move away, the speed is restored to its initial value.

The first procedure requires as input the position of the collision event $P_{collision}$, the current position, speed and acceleration of the vehicle (P_0, V_0, a_0) and the value of τ. In Line 2, the algorithm computes the radius of the safety shield, as explained in Section 4.4.1. Then, the distance δ between the vehicle and the collision event is extracted using the Haversine formula; now ρ and δ are used to derive the value of the coefficient λ (Line 4).

A random salt is added to λ to avoid that the involved vehicles follows exactly the same deceleration profile; from Line 10 the new speed is computed, it is capped to 10 km/h and finally, in Line 14, the vehicle's speed is adjusted accordingly. The last two lines of this procedure are used to trigger the next routine, that is in charge of restoring the vehicle's speed to its original value, as soon as it exits from the dangerous area.

This procedure requires as input the position of the collision event $P_{collision}$, the current position P_0 and the δ_{old} computed in the previous iteration. In Line 19, the new value of δ is computed: if it is observed to be greater than δ_{old}, then it means that the vehicle is moving away from the collision event, therefore the algorithm restore its initial speed (Line 21). On the contrary, if δ is lower than δ_{old} it means that the vehicle is approaching the collision event location, therefore a new iteration of the procedure is rescheduled after 1 second.

The whole algorithm is computationally inexpensive and can be deployed even in devices with very low performances, since no recursive structures nor particular memory requirements are needed. Some of the parameters, such as the value of τ, determine the effectiveness of the system; other parameters, such as the frequency at which the "Distance Check" is called, determine instead the rapidity of the system in recovering

Algorithm 1 Evasive maneuver pseudocode

procedure SPEED OVERRIDE
Require: $P_{collision}, P_0, V_0, a_0, \tau$
    $\rho = V_0\tau + \frac{1}{2}a_0\tau^2$
    δ = computeHaversine($P_{collision}$, P_0)
    $\lambda = \delta/\rho$
    rand = uniform(-0.1λ,0.1λ)
    λ = λ+rand
    if $\lambda > 1$ **then**
        $\lambda = 1$
    end if
    $V_{new} = V_0\lambda$
    if $V_{new} < 10km/h$ **then**
        $V_{new} = 10km/h$
    end if
    setSpeed(V_{new})
    wait(1)
    checkDistance(δ,$P_{collision}$)
end procedure
procedure DISTANCE CHECK
Require: $P_{collision}, P_0, \delta_{old}$
    δ = computeHaversine($P_{collision}$, P_0)
    if $\delta > \delta_{old}$ **then**
        restoreInitialSpeed()
    else
        wait(1)
        checkDistance(δ,$P_{collision}$)
    end if
end procedure

the original speed after a collision event is reported by CAS.

When deployed, this system makes the vehicles automatically slow down, enabling a safer approach to dangerous intersections; each time a collision is reported by CAS, the speed of the involved vehicles is reduced. If multiple collision events are reported, the vehicles involved slow down up to 10 km/h, so that they can safely cross the intersection.

In the next section, the proposed system is evaluated using MS-VAN3T. The evasive strategy is implemented directly in the vehicles and supported by the centralized version of CAS (with LTE as access technology). Nevertheless, the system can be deployed in the distributed version, using any of the available access technologies, as long as CAMs and DENMs are correctly delivered. The analysis will focus on the algorithm effectiveness by measuring, for each simulation, the number of collisions happening

with or without the proposed strategy in place.

4.4.4 Simulation results

The scenario where the proposed strategy is deployed and validated is the same used in Section 4.3.1 to test the centralized version of CAS. The simulation involves 10 vehicles randomly driving around the map, a low number that avoids traffic congestion problems and keeps the average speed of vehicles the highest possible; each vehicle is connected to the LTE core network, where a server runs all the CAS logic and transmit DENMs when needed. CAS is configured, in this case, to generate DENMs for collision happening no more than 15 seconds in the future. Each simulation lasts 3600 seconds of simulated time, and the number of collisions happening in each simulation is studied by increasing the maximum speed of vehicles, and by increasing the value of τ, the parameter determining the size of the virtual safety shield.

The speed of vehicles is tested from 40 to 130 km/h while the value of τ from 3 to 15 seconds. The histograms shown in Figure 4.17, report the total number of collisions registered as a function of the maximum speed of vehicles. Each bar corresponds to a different value of τ, with the first one (the light blue striped bar) corresponding to the case in which the system is disabled. Note that the Figure includes only the values of τ for which at least a collision has occurred. Therefore, the first important result derived from Figure 4.17 is the total absence of collisions for value of τ higher than 8 seconds.

By analyzing the lower values of τ, instead, it can be noted that there are certain combinations of maximum speed and τ that make the system even worse than if no action is taken at all. For example, the simulation with vehicles running at 120 km/h reported 19 collisions with the algorithm disabled, and 22 when τ is set to 3 seconds. This strange behavior happens because the system overrides the vehicles' speed control too late, causing them to just slow down a little bit and to occupy the intersection region more than the case in which the algorithm is disabled.

Beside this single case, in which the system introduces a worsening of the studied KPI, it can be noted that even for small value of τ, the evasive strategy causes, on average, the number of collision to be lower. Moreover, it is worth mentioning that the speed at which the collision happens is reduced, since in some measure the algorithm takes the control of vehicles, and reduces the speed at which they approach the intersection (even if too late).

The full analysis on collisions is shown in Figure 4.18. The Figure depicts a heat-map where each point corresponds to the number of collisions as a function of the maximum speed set on vehicles and of the value of τ. Of course, the lower the value of τ, the higher the possibility that the vehicles do not have the time to avert the collision. At the same time, the higher the maximum speed of vehicles, the higher the possibility that the evasive maneuver is unsuccessful (especially for low values of τ). The heat-map highlights that the minimum value of τ ensuring the full protection for the scenario studied is between 8 and 9 seconds. As a matter of facts, every simulation that has been

run with τ greater than 8 seconds experienced 0 collisions, even at high speeds.

To conclude, in this last section it has been shown how the information generated by CAS can be used to effectively actuate collision avoidance strategies directly on the vehicles' control unit. The solution here developed uses the information on speed and acceleration to build the virtual safety shield around the vehicle: the reason for leveraging only on those parameters is driven by the fact that speed and acceleration are the only useful values that SUMO returns. In a real scenario, the size of the virtual safety shield may take into account the information coming from other sensors present on board (e.g., those indicating the presence of ice on the road surface), as well as other information like the driver psycho-physical status, the current mass of the vehicle or other information coming from the road side infrastructure. Nevertheless, the proposed system (if correctly configured) is able to tackle all the collisions at the intersections and to build a safer environment by only using the information contained in CAM and DENM messages generated by the vehicles and the road side units.

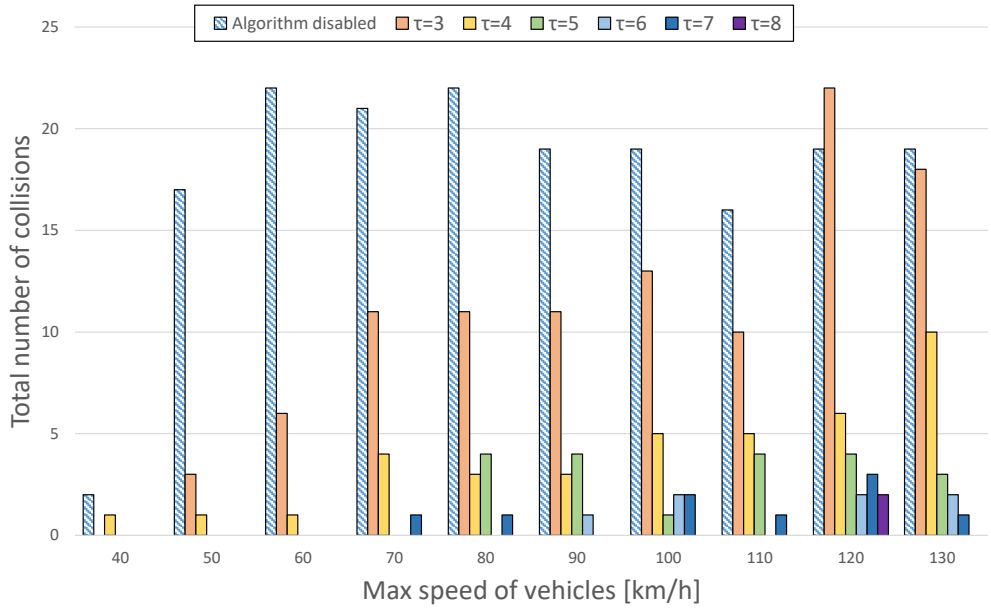


Figure 4.17: Number of collisions happening as a function of the maximum speed of vehicles. Each bar corresponds to a different value of τ. The striped light blue bar corresponds to the case in which no action is taken by vehicles to avert the collisions. The plot reports only values of τ for which at least a collision has occurred.

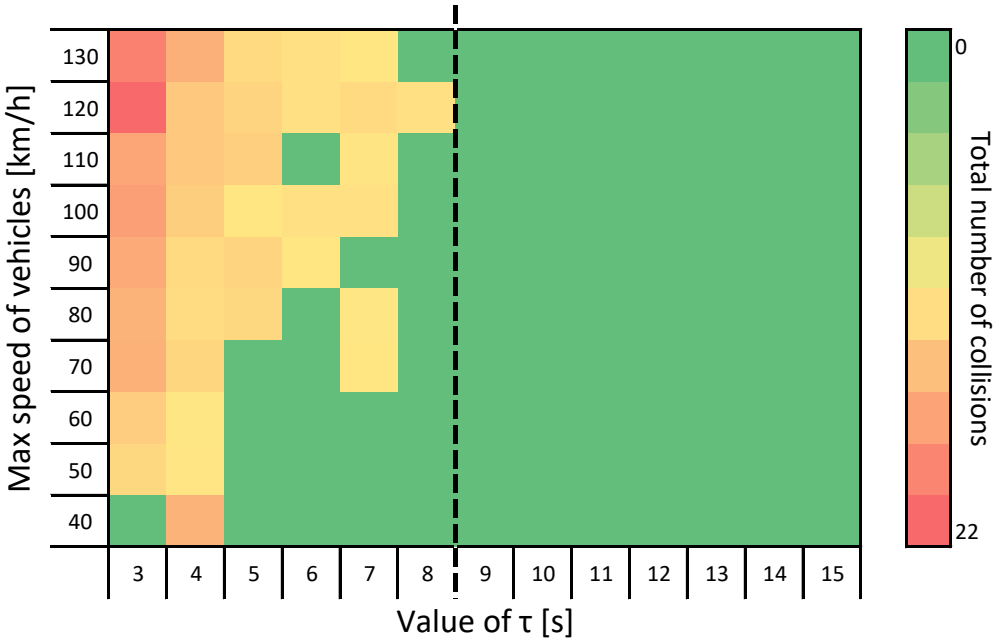


Figure 4.18: Heat-map representing the total number of collisions as a function of the maximum speed of vehicles and of the selected value of τ. The vertical dashed line represents the minimum value of τ guaranteeing 0 collisions.

Chapter 5

Open source solutions for V2X-enabled embedded devices

Beside the implementation of MS-VAN3T and of the safety application for intersection collision avoidance, this thesis presents the efforts that have been made to realize open source tools for the development and the performance assessment of fully working V2X devices.

This chapter presents the implementation of an open source testing platform for 802.11p wireless cards; the platform is used to assess the performance of Unex DHXA-222, a model of wireless network interface controller (WNIC) enabling 802.11p-like connection. Nevertheless, the proposed solution (which is based on the open source distribution OpenWrt [78]) can be used to test any card supporting the *ath9k* Linux driver. The choice of releasing the platform as open source allows other researchers to validate this work, possibly on different hardware, making it easier to pinpoint any discrepancies between expected and actual results. For this purpose, the source code of every element developed in this work is available through GitHub [77, 84, 58].

The proposed system is initially tested using common tools, like `iPerf`; successively, the need for precise measurements of the end-to-end latency experienced by V2X applications running on top of the platform, drives the design of a novel protocol, named LaMP (**La**tency **M**easurement **P**rotocol). The LaMP protocol is then used as a baseline for the development of an open source application, LaTe (**La**tency **Te**ster), that can perform micro-seconds precise application layer latency measurements using LaMP.

5.1 Embedded devices for IEEE 802.11p communication: state of the art

As already introduced and discussed in Chapter 2, the current 802.11p wireless access technology for V2X communication is based on IEEE 802.11-2016 [51] and it is designed to operate in the 5.9 GHz band. IEEE is currently working on the introduction of a new amendment, termed IEEE 802.11bd [45], to cater for emerging use cases for V2X communication technology and strengthen its foothold on 802.11-based technology for V2X applications.

However, before any 802.11bd card hits the market, legacy 802.11p is likely to be the hardware of choice for car manufacturers seeking to equip their latest models with wireless-LAN communication capabilities.

This explains the need for a clear characterization of IEEE 802.11p cards, starting from a controlled lab environment. The challenge of equipping vehicles with standard-compliant OBUs, able to communicate in the 5.8-5.9 GHz frequency band, was undertaken by major car manufacturers all over the world. The Volkswagen Group announced the adoption, from 2019 onward, of 802.11p-compliant equipment in their vehicles with the aim of reducing road accidents and enable services such as platooning, that helps also in the reduction of fuel consumption and carbon dioxide emissions [98]. Toyota- and Lexus-branded vehicles in the USA will see the introduction of DSRC systems in 2021 [92]. It is likely that the majority of car manufacturers will start adopting these solutions in the next few years.

During the last years, some commercial ready-to-use solutions have been made available on the market; those systems can support the 802.11p-based vehicular communication. In the awareness of not providing a complete and exhaustive list, some available solutions are enumerated here, updating the list in [11].

- The LocoMate series, produced by ARADA Systems, which provides hardware solutions for both RSUs and OBUs. Every board is integrated with Bluetooth, GPS with one-meter accuracy, high-power radios, and a full WAVE stack.
- MK5, developed by Cohda Wireless, implementing both the WAVE and ITS-G5 stack. According to Cohda their hardware is rugged, small and relatively low cost, with dual 802.11p radio [68]. These systems are based on the automotive-based chip RoadLINK produced by NXP and on a firmware written by Cohda.
- KVE-3320 V2X ECU, produced by Kapsch TrafficCom, a DSRC 5.9 GHz device supporting all the V2X standards. This product is advertised as "a rapid development framework for V2X applications with a small footprint and high reliability" [55]. A variety of V2X applications are, at the time of writing, being developed by Kapsch and an SDK is part of the software suite, for developers willing to build V2X applications on top of KVE.

The cost of the listed devices is usually high and the customization possibility is, conversely, often null. For these reasons, open solutions are often preferred in the academic world since they have the advantage of being more accessible and customizable for research and testing purposes. Two of the main companies developing hardware that can be used to build custom V2X embedded solutions are:

- Unex [93], producing some of the wireless cards most commonly found in the vehicular communications field, including the DHXA-222 MIMO 2x2. Those cards can be used to transmit over the 5.8/5.9 GHz frequency band and fully support the *ath9k* Linux driver. Other solutions proposed by Unex are the brand-new OBU-301E/U and OBU-351x, enabling respectively wireless or cellular-based V2X communications on top of ITS-G5 or WAVE stack.

- PC Engines, producing embedded boards targeted at networking applications and supporting any compatible WNIC; PC Engines devices provide a quite customizable architecture: they always include at least one miniPCI slot to accommodate a proper WNIC, such as the aforementioned Unex cards.

There is a large number of published studies proposing combinations of off-the-shelf hardware and open-source software to build working 802.11p solutions. Qin *et al.* [80], developed a testbed using PC Engines' ALIX3D3 system boards together with Unex's DCMA-86P2 and a Voyage Linux distribution. Their work focuses on the measurement of packet loss rate and latency under different speed and distances; their work highlights that, using standard-compliant systems, the devices can communicate within a distance of 300 m and at a speed of 18km/h.

Kamal *et al.* [54] designed and validated, through a DSRC spectrum analysis, a testbed using PC Engine's ALIX1D and Unex's DCMA-86P2. The same configuration, using PC Engines and Unex hardware combined with OpenWrt, was used by Agafonovs *et al.* [11] to investigate achievable bandwidth and RSSI (Received Signal Strength Indicator). Their results differ from those of [80], and claim that the proposed solution provides reliable data transmission (more than -85dBm) up to 1.2km and partial communication up to 2.4km.

Another important portion of the literature proposes and evaluates V2X-dedicated software frameworks, such as the OpenWrt 10.03 Backfire modified in the GCDC challenge, the CVIS OS, the C2X platform, and other several patches enabling services in the automotive domain [59, 43, 41, 16, 97].

The platform proposed in this thesis implements a fully-working 802.11p solution using PC Engines APU1D and Unex DHXA-222 with patched *ath9k* drivers and the Linux kernel 4.14.63. The system is analyzed by firstly measuring the packet loss and the reachable throughput under different conditions, as well as by assessing the functionalities of the 4 MAC layer traffic classes introduced with EDCA. Then, thanks to the design of the custom protocol LaMP and to the implementation of the measurement software LaTe, the end-to-end latency experienced by the platform is analyzed.

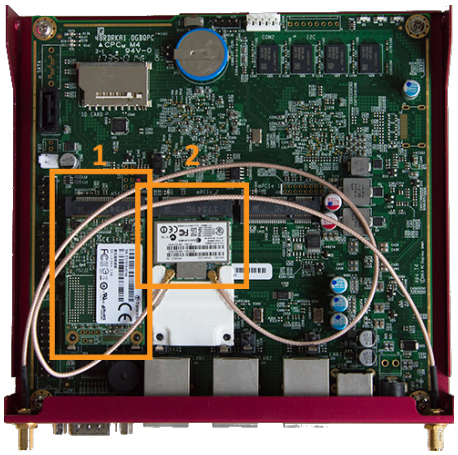


Figure 5.1: PC Engines' APU1D boards without enclosure; the picture highlights: (1) the Transcend 16 GB SSD, mounted on the mSATA slot, and (2) the Unex DHXA-222 WNIC, mounted on one of the two mini PCIe slots, with two U.FL to RP-SMA pigtail connectors.

5.2 Testbed description

The implementation and integration work was performed on PC Engines APU1D boards, a flexible, customizable Linux-based platform for networking systems. These boards are still in production (at the time of writing) and provide an up-to-date solution matching all the requirements to support complex VANETs applications.

The processor is a dual-core AMD G-Series T40E x86, while the memory installed is a 2 GB DDR3-1066 DRAM; the system additionally allows the installation of a mSATA SSD as secondary storage. As SSD device, a 16 GB SATA III Transcend MSA370 is mounted.

The wireless card installed is a Unex DHXA-222, whose main characteristics are summarized below:

- Half-size mini PCI express chips
- Dual band
- MIMO 2x2 operations
- Support for ITS frequencies
- Support for Bluetooth 4.0
- Declared maximum output power: 17 dBm

DHXA-222 is based on the Atheros AR9462 chipset, and fully supports the *ath9k* Linux driver. This chipset represents one of the main characteristics that drives the selection

of this kind of cards, since other chipsets supported by *ath9k* have been successfully tested and used in many research works, including the development of the OpenC2X embedded platform [59].

The operating system on top of which the platform is built is the OpenWrt release 18.06.1, with Linux kernel 4.14.63. Both Linux kernel and OpenWrt are being constantly updated with bug fixes and new features; as a matter of fact, working with recent software can be quite important when testing and deploying vehicular network applications, as newer software can include and improve some of the features which already made their way inside the kernel (e.g., the OCB mode, which is included in the kernel bundled with OpenWrt 18.06.1). Indeed, the capability of updating the various firmware can play a relevant role in future VANETs, since the system deployed inside the vehicles will likely receive constant updates to fix inefficiencies and bugs.

The choice of OpenWrt was also driven by a slew of benefits it can provide to a vehicular networking system like the one under study:

- It is well suited to networking in embedded devices; it is used massively, for instance, as a replacement firmware for routers.
- It supports several devices, not only the ones based on x86 as the PC Engines boards.
- It is highly customizable, allowing the user to choose all the needed packages and configurations for the specific application it is meant to, instead of providing just a static firmware.
- It is well supported by a strong community, including an official forum and a quite complete online documentation.
- Last but not least, several other research works, such as [9], select OpenWrt as preferred choice among embedded Linux distributions.

The next sections describe the platform setup more in details, introducing MAC and physical layers.

5.2.1 MAC layer

For what concerns the MAC layer, the work focuses on the EDCA functionality and its performance assessment, since it is one of the main features supported by 802.11p [51]. The proposed platform includes a patch of the *mac80211* Linux wireless subsystem; the patch allows the development of soft-MAC driver, allowing the MLME to be implemented in software. The patch was again provided by the OpenC2X project and ported to OpenWrt 18.06.1: using this solution it is possible to directly select the EDCA traffic class by calling the function `setsockopt()` with particular flags.

A tool for measuring the traffic transmitted over different Access Categories (ACs) was however missing: for this reason, it was required to slightly modify `iPerf` 2.0.12

(which, in general, is the tool of choice as far as throughput measurements are concerned). With this patch, `iPerf` is added of an additional command-line option to generate traffic in any of the 4 traffic classes (BK, BE, VI or VO).

5.2.2 Physical layer

The physical layer is managed in hardware by the Unex WNICs, as opposed to SDR solutions in which physical layer algorithms are directly implemented in software.

To enable the usage of the DSRC channels in the 5.9 GHz frequency band, the *ath9k* driver requires a patch for the support of both OCB operational mode and the selection of 10 MHz-wide channels. For this purpose, the platform includes the kernel patches provided with the OpenC2X platform, developed by the *CCS Labs* group at the University of Paderborn [59].

These patches were developed for the Linux kernel version 3.18, in devices running LEDE 17.01 (an older branch of OpenWrt). With very few modifications, they were ported to OpenWrt 18.06.1, effectively enabling the use of ITS channels (from 172 to 184), together with an an-hoc patch for the Italian (IT) and German (DE) regulatory database flags. With the built-in capability of the `iw` Linux wireless configuration tool to select OCB mode and 10 MHz-wide channels, it was possible to use any of the 802.11p channels as required by the standard. Through the usage of configuration scripts it was eventually possible to transmit on 802.11p channels, and to tweak both the transmission power and the physical bit rate. Unfortunately, among the bit rate foreseen by the 802.11p standard, the Unex DHXA cards (when configured with 10 MHz-wide channels) are enabled to only transmit at 3, 6 and 12 Mbit/s.

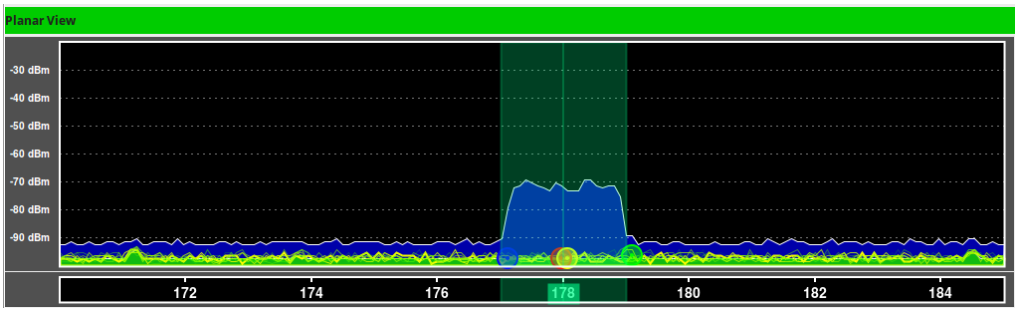


Figure 5.2: Planar view of Kismet Spectrum-Tools showing the captured spectrum during a measurement session with the `iPerf` tool, with a board transmitting in channel 178.

5.3 Performance evaluation

Before starting the tests, the system was validated by analyzing the frequency spectrum used by the boards running the proposed platform. The analysis was made using an external spectrum analyzers (*MetaGeek Wi-Spy analyzer*), coupled with the open source software Kismet Spectrum-Tools.

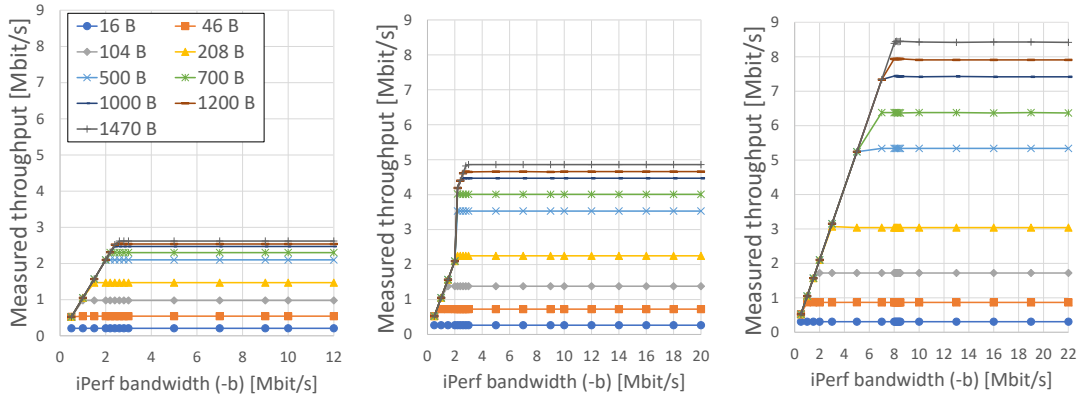
As Figure 5.2 depicts, it was possible to verify the correct 10 MHz-wide channel usage, together with the absence of interference in the lab where the tests took place; the maximum collected signal level outside the band of interest is of -92 dBm, a value that can be interpreted as pure noise.

The tests presented in the next sections are collected using two APU1D boards, each one equipped with a DHXA-222 WNIC and configured as previously detailed.

5.3.1 Throughput and packet loss measurements

As a first step in the performance assessment, the platform underwent a series of measurements involving the `iPerf` network measurement tool. The usage of a widely known tool like `iPerf`, makes the collected results easier to be compared with others, since `iPerf` is available in (almost) all Linux and non-Linux distributions and operating systems. All the tests are performed using UDP as transport protocol.

The analysis focuses on throughput and packet loss when selecting different physical data rates (among 3, 6 or 12 Mbit/s) and by varying the payload size of the offered traffic from 16 B to 1470 B. The goal is to measure the KPIs under almost "ideal" conditions, with the two boards placed next to each other. Each test, yielding a single data point, lasted for 60 seconds.



(a) Bit rate = 3 Mbit/s. (b) Bit rate = 6 Mbit/s. (c) Bit rate = 12 Mbit/s.

Figure 5.3: Throughput measurements with different payload size as a function of the traffic offered by `iPerf`. Each plot is referred to a different 802.11p bit rate; txpower = 3 dBm, antennas gain = 5 dBi, and distance of ~16 cm between the enclosures.

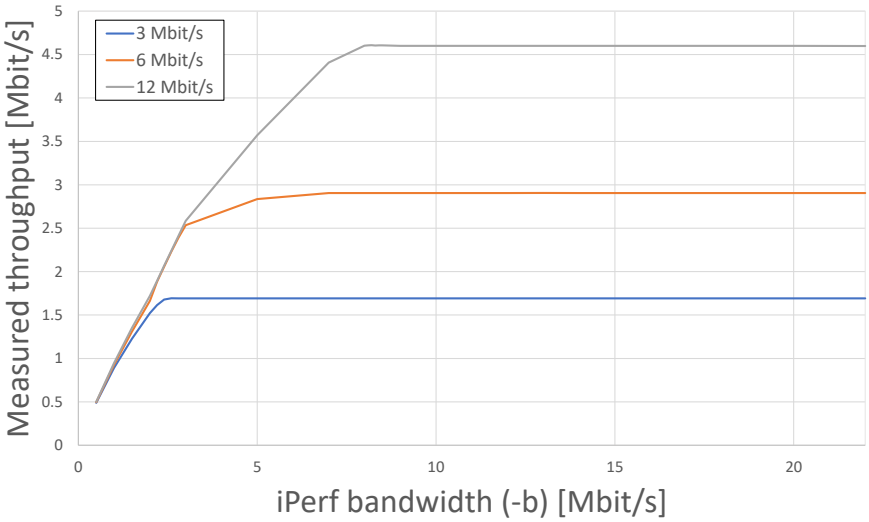


Figure 5.4: Throughput measurements with different 802.11p bit rate as a function of the traffic offered by `iPerf`; txpower = 3 dBm, antennas gain = 5 dBi, and distance of ~16 cm between the enclosures.

The plots obtained by testing the boards at 3, 6 and 12 Mbit/s are depicted in Figure 5.3 and Figure 5.4, where the measured throughput is reported as a function of the traffic offered by `iPerf`. As expected, the reachable throughput depends on the payload size: the higher the payload, the higher the measured throughput; therefore, the maximum efficiency is reached with packets of size 1470 B. The maximum observed throughput, with a bit rate of 3 Mbit/s, is 2.63 Mbit/s (Figure 5.3a); a bit rate of 6 Mbit/s ensures a maximum throughput of 4.86 Mbit/s (Figure 5.3b); 8.42 Mbit/s is instead the maximum throughput when the bit rate is 12 Mbit/s (Figure 5.3c).

Therefore, the plot in Figure 5.4 averages the measured throughput over all payload sizes. The result highlights that by selecting a bit rate of 3 Mbit/s, and by sending packets of various size, the average throughput will not exceed 1.7 Mbit/s; for a bit rate of 6 Mbit/s the average is 2.9 Mbit/s, and finally for 12 Mbit/s the average throughput is 4.6 Mbit/s.

Figure 5.5 suggests that, as we offer too much traffic with respect to what the network can deliver (for instance when trying to offer 10 Mbit/s with a physical data rate of 3 Mbit/s, generating a congested situation), the packet loss tends to increase, up to a certain point, where the value stabilizes. The reasons for such a packet loss are to be sought in the interaction between the iPerf application and the lower layers, mediated by buffers at the socket level. The buffers, as the offered amount of traffic increases, get full and start to drop packets. The same type of behavior occurs with any application that writes on UDP socket buffer.

Another important hint on the system performances when transmitting at different data rate, is that from the point of view of the packet loss, the transmission at 6 Mbit/s

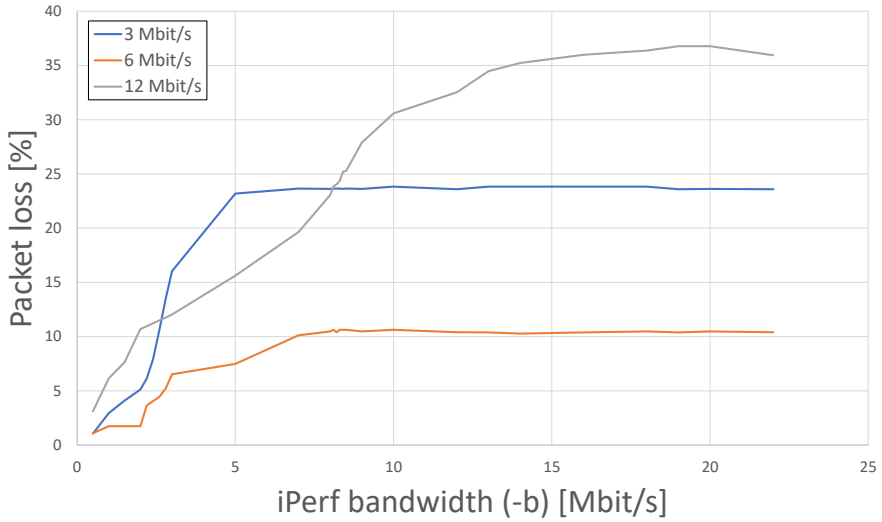


Figure 5.5: Packet loss measurements with different 802.11p bit rate as a function of the traffic offered by `iPerf`; txpower = 3 dBm, antennas gain = 5 dBi, and distance of ~16 cm between the enclosures.

seems to perform better than the one at 3 Mbit/s.

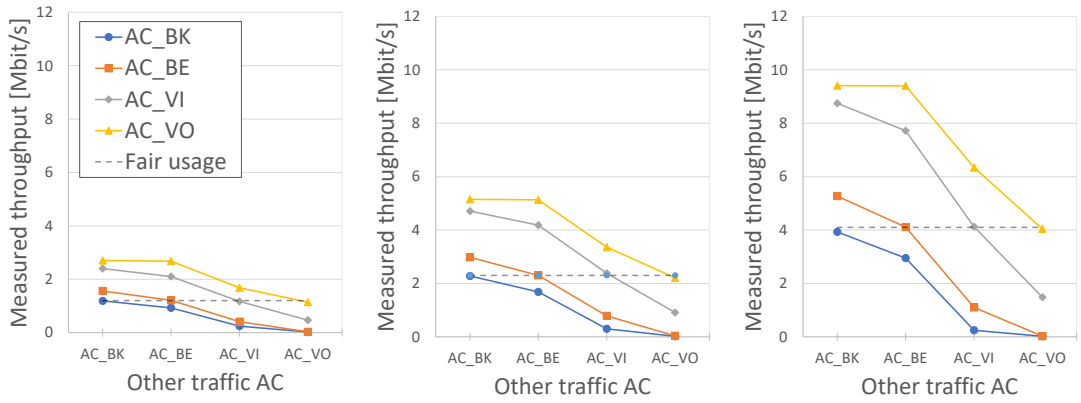
The packet loss trend is less linear than the throughput one, due to the combined effect of the UDP buffers and due to the WNIC transmitting packets over the wireless medium. In any case, as expected, the increase in the amount of offered traffic always corresponds to an increase in packet loss values.

5.3.2 Traffic classes and access categories

An important part of the platform validation and evaluation phase is related to measurements of the transmissions at different EDCA Access Categories (ACs), obtained by using the patched version of `iPerf` introduced in Section 5.2.

The tests are performed by launching an `iPerf` client and an `iPerf` server on each of the two boards, with the client instance on one board transmitting to the server instance on the other board, and vice versa. The clients are configured to transmit over different ACs, with one of the two boards transmitting all the "useful" data, while the other only generating interfering traffic on different ACs. The focus, in this case, is on the reachable throughput measured by `iPerf`, and on the estimated connection stability, computed as the maximum percentage variation in the throughput values reported by `iPerf` every 2 seconds. Each test, representing a single data point, is set to last 60 seconds.

The plots showing the results with the boards transmitting at 3, 6 and 12 Mbit/s are reported in Figures 5.6 and 5.7. Figure 5.6 reports the reachable throughput as a function of the interfering traffic AC. Every line corresponds to a flow with different



(a) Bit rate = 3 Mbit/s. (b) Bit rate = 6 Mbit/s. (c) Bit rate = 12 Mbit/s.

Figure 5.6: Throughput measurements when using different ACs, with background interfering traffic transmitting on different ACs (horizontal axis).

priority, and the value in the horizontal axis are sorted by increasing priority.

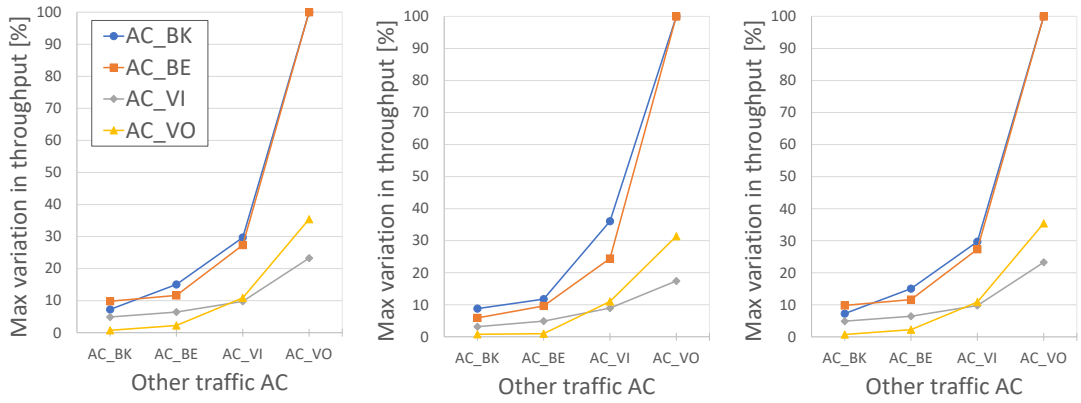
Figure 5.7 shows instead the maximum percentage throughput variation as a function of the interfering traffic AC. A low percentage value means that the connection is quite stable, while a high value indicates some instability. The value of this variation is computed as:

$$\frac{throughput_{max} - throughput_{min}}{throughput_{max}} * 100 \tag{5.1}$$

Where $throughput_{max}$ and $throughput_{min}$ are respectively the maximum and minimum values of throughput reported by iPerf in each measurement session. By observing the plots, it seems that everything works as expected: the system exhibits a behavior which is coherent with theory, with better results when transmitting packets over a higher priority queue. When a board transmits using a high priority AC, such as AC_VI or AC_VO, it can always reach a higher throughput value even in presence of another traffic stream at a lower priority AC, such as AC_BK or AC_BE, as expected. The connection stability is also improved when transmitting higher priority traffic, no matter the traffic class used by the other board, generating the interfering traffic.

It is worth observing that, when two contending flows use the same traffic class, the channel usage is fair. This can be seen in Figure 5.6a depicted as the grey dashed horizontal line ("fair usage", in the plot) around 1.2 Mbit/s. The same consideration holds for the other physical data rates: a similar "fair channel line" is drawn at 2.3 Mbit/s for the bit rate equal to 6 Mbit/s and at 4.1 Mbit/s for the bit rate equal to 12 Mbit/s.

In Figure 5.8, the throughput when testing a single flow is plotted as a function of the selected AC. Each line corresponds to a different data rate, ranging again from 3 to 12 Mbit/s. The throughput measurements were performed trying to offer a higher



(a) Bit rate = 3 Mbit/s. (b) Bit rate = 6 Mbit/s. (c) Bit rate = 12 Mbit/s.

Figure 5.7: Maximum % variation in the throughput, with background interfering traffic transmitting on different ACs (horizontal axis).

amount of traffic with respect to the available physical data rates, to test the maximum reachable throughput. The measured values increase as the priority is increased, thanks to shorter AIFS (which are all different when working in OCB mode, as shown in Section 2.2.3) and smaller contention window sizes.

The presented results suggest that the WNICs used in the platform can maintain a stable connection; moreover the coherence of the results validated the proposed open source platform and the patched `iPerf` version as far as EDCA traffic classes are concerned.

5.3.3 Received power and connectivity measurements

Part of the performance evaluation work is related to indoor throughput, packet loss and connection stability measurements, with a single data stream over a chosen AC (AC_BE, in this case). The results are obtained by deploying a couple of `iPerf` client/server instances on the boards, putting them at increasing distances and finally correlating the achievable throughput with the average received signal level.

Since the measurements were performed indoors with the boards not transmitting at the maximum power, instead of plotting the results as a function of the distance, they are plotted as a function of the measured received power. In this case, in fact, the measured received power represents a more meaningful quantity: the communication, when performed indoors, is affected by the various obstacles standing in the way. Different values of received power are thus likely to be observed at the same distance, when the boards are moved to different places.

The results are shown in Figure 5.9, which shows the throughput as a function of the received power values. Each line corresponds to a different packet size.

Taking into account that these tests are all static, it is possible to show that the

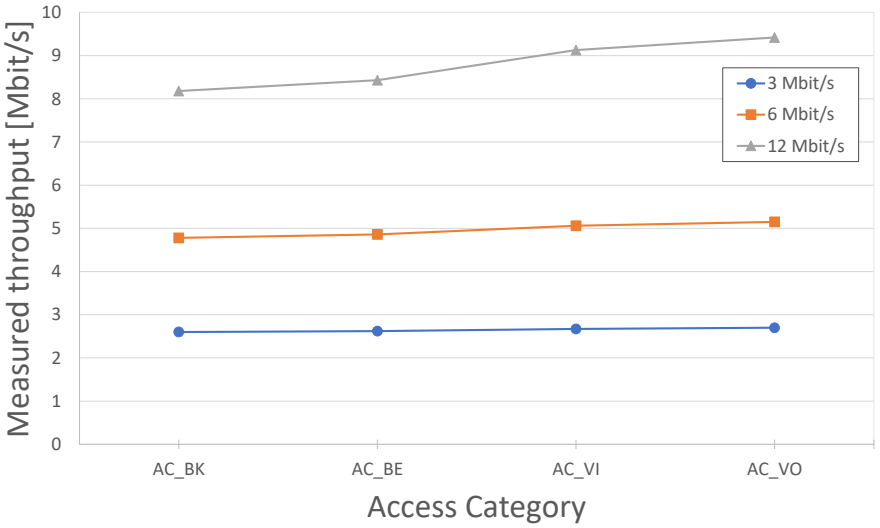


Figure 5.8: Throughput of a single stream (i.e., no interfering traffic is present) as a function of the selected access category. Each line corresponds to a different data rate; payload length: 1470 B, offered traffic: 10 Mbit/s

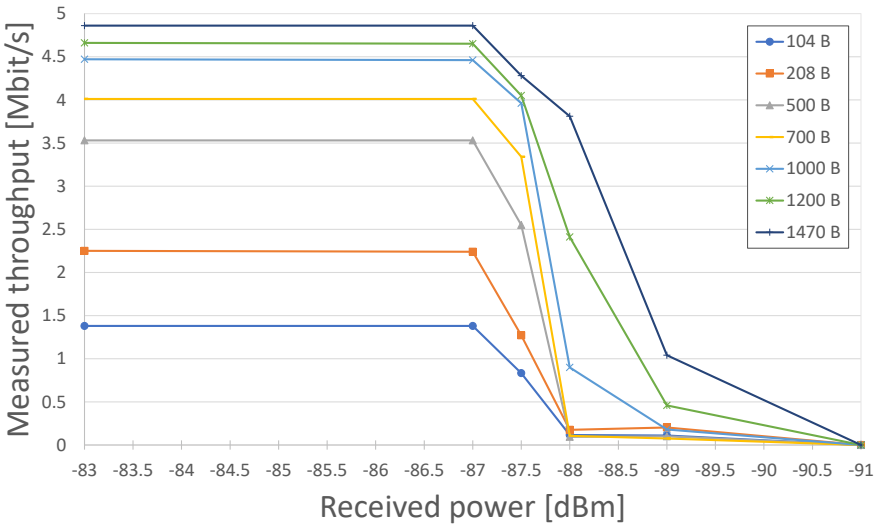


Figure 5.9: Throughput measurements for different values of received power; physical data rate: 6 Mbit/s, payload length: 1470 B, *txpower*: 10 dBm, antennas gain = 5 dBi, offered traffic: 10 Mbit/s

connection stability and the reachable throughput remain quite high until the received power is above *-87 dBm*; below this value, an abrupt drop of the throughput is observed until the connection can no longer be established. When the signal is weaker, small distance variations were sufficient to cause instabilities in the throughput and in the

received power.

5.4 LaMP: a novel protocol for precise latency measurements

Another analysis required to complete the performance assessment of the aforementioned WNICs is the one on end-to-end latency. Many of the day-1 applications enabled by VANETs are going to heavily increase the demand for ultra-Reliable Low-Latency Communication (uRLLC), one of the tenets of 5G networks. For this reason, network latency measurements tools will play a crucial role in the provisioning of reliable and efficient networked services.

The target of the work presented in the next sections is to provide a tool that helps to precisely characterize the latency experienced by applications deployed on top of V2X platforms. More in general, the proposed solution can be used to assess the latency performance on any networked system.

An important concept, which will be recurrent all over the next sections, is that of *application-layer latency*: the term refers to the real latency that applications experience when sending data to and receiving data from a communication system. The main contributions on this KPI are introduced by the channel access scheme of the adopted technology: as shown in Chapter 3, the various access technologies adopt different strategies, offering different performance to the overlying application. However, there are also other layers of the communication stack introducing non-negligible contributions.

A latency measurement must take into account three components: processing, transmission, and propagation delay. The first value depends on the computing capability of the devices under test (DUTs), the second on the physical media and on the network capabilities, while the third on the distance between the involved DUTs. High latency in a network generates bottlenecks that prevent the data from fully exploiting the network pipe and, in certain cases (such as in TCP-based connections), it decreases the communication bandwidth.

5.4.1 Existing solutions for precise latency measurement

In the literature, it is possible to find several examples of tools and protocols that have been designed to measure the latency between endpoints; the most notable one is the ICMP protocol, in particular for what the “Echo Reply” and “Echo Request” packets are concerned. Every IP compliant system should be able to reply to ICMP requests coming from another node in the network. This mechanism is exploited by tools like `ping` and, using timestamps which are embedded in the packets or stored inside the application, latency measurements are made possible.

Another example is Metherxis [62], a system leveraging Virtualized Network Measurements Functions (VNMFs) to measure network device latency with micro-second grade accuracy. The idea at the base is to use Linux Containers to create a range of VNMFs under a single Linux host; some of them are used to instantiate packet generators, others packet analyzers.

Other specifically targeted protocols, such as the recently proposed Two-Way Active Measurement Protocol (TWAMP) [8], works over a client/server architecture allowing one or two way delay measurements. Finally, a dedicated IP Timestamp Option, that can be used to determine the latency between single links is also available, as highlighted by Sherry in [89]. After identifying the above solutions as the most common for this kind of measurements, it is important to remark that they have two main drawbacks.

First of all, `ping` relies on ICMP. Even though it can be very practical to leverage something that is implemented inside almost every networking system (i.e., the ICMP echo reply mechanism), only ICMP can actually be used to transport the timestamp data needed to compute the latency between nodes. Moreover, by using ICMP, it is possible to compute the network latency, but tools like `ping` do not provide the user with a clear estimate on the latency experienced at the application layer. Indeed, the echo reply mechanism is implemented, at least as far as Linux is concerned, as part of the kernel, without any server-side transition to the user space.

Then, Metherxis, TWAMP, and other specific protocols, even though very precise in giving the desired measurement values, require the tested device to be compliant to specific standards (or to specific options, such as in the IP case) and possibly need some additional capabilities to be implemented, which may not be the case for all the network nodes.

For these reasons, this thesis introduces the Latency Measurement Protocol (LaMP), an application-layer protocol which can be encapsulated inside any lower-layer protocol. LaMP addresses the need for a lightweight framework encapsulating the basic information needed to perform accurate latency measurements. In addition, this work introduces the first open-source tool leveraging LaMP, i.e., LaTe (*La*tency *Te*ster), a flexible client-server application that currently supports LaMP over UDP/IP to perform latency measurements under different conditions.

5.4.2 LaMP protocol description

LaMP has been developed as a flexible application layer protocol which can be encapsulated inside any lower layer protocol, and is designed to be as much self-contained as possible. For instance, it can be encapsulated without requiring any modification to its rules and packet format, inside UDP over IPv4, or directly inside a local network raw Ethernet packet, or inside WSMP packets and transmitted over 802.11p.

In a nutshell, the protocol follows a client-server paradigm: LaMP requests are sent by a client and received by a server, which replies back to the client. In a typical scenario, the application implementing LaMP computes the latency as the time difference

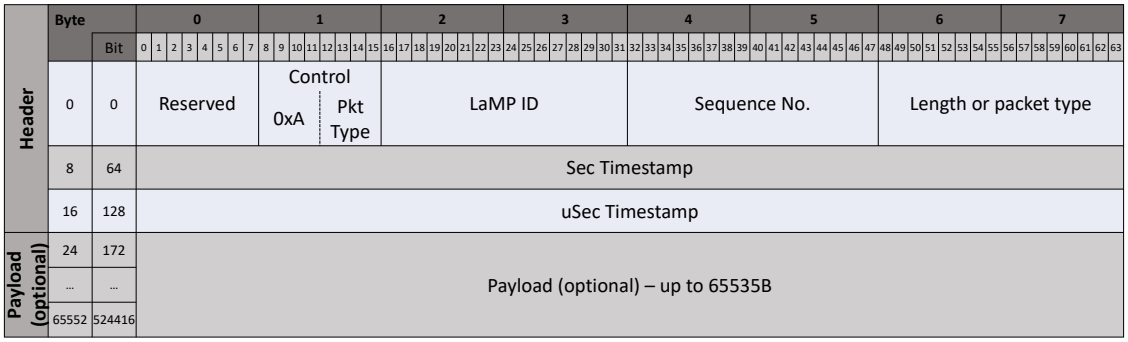


Figure 5.10: Structure of the LaMP PDU with a description of all the fields, from [58].

between a "send timestamp" and a "receive timestamp", which are managed by the applications participating in the measurement session.

Typically, the "send timestamp" is inserted inside the LaMP packet by the entity sending a request and the "receive timestamp" is instead gathered by the entity when a reply is correctly parsed. Before any session is started, a connection initialization packet is sent by the client and properly acknowledged by the server, in order to ensure that measurements can start and the connection is correctly established.

Additional modes are also defined, including the unidirectional mode, in which the client only sends requests to the server, which is responsible for computing the latency and returning it to the client at the end of the test. This mode enables the computation of the one-way latency, but, as of now, it requires the clocks of the different devices to be precisely synchronized through the Precision Time Protocol (PTP) [46] or through the Network Time Protocol (NTP) [73].

The LaMP packet is composed by a header and by an optional payload. The header size is equal to 24B, accounting for the need of a lightweight protocol but including all the necessary data, as shown in Figure 5.10. Every LaMP packet starts with a *reserved header* field, which is used by the nodes to identify whether the received data really belongs to a LaMP SDU, and with a *control field*, which is further divided into a reserved sub-field (4 B, used to lengthen the reserved field and make the protocol more robust) and in a field indicating the packet type (4 B).

Therefore, LaMP can account for a total of 16 different packet types. Then a 16-bit long field, the *LaMP ID*, contains a number which is used to identify each LaMP client-server measurement session, and which allows the server to correctly associate each client with the corresponding session. This mechanism can be used as a basic system to identify each LaMP session and can be integrated, if necessary, with other protocol-specific session identification mechanisms, such as ports when UDP is used, or the Provider Service ID when WSMP is used.

A 16-bit *sequence number* is then included, enabling the association between each

request and reply. The following 16 bit-long field, named *length or packet type* is instead used to store the payload size or, if the packet is of type *connection initialization*, the type of connection which should be established (unidirectional or bidirectional).

Finally, LaMP is designed to embed, together with two 64-bit precise *second and microsecond timestamps*, any additional *payload* (up to 65535 bytes) in which other data and possibly other user-defined protocols can be encapsulated.

As each bidirectional measurement session is completed, the client should gather some statistics and report them to the user. In the one-way mode instead, the server is responsible for the latency computation and for returning a final report to the client.

The full protocol specifications are open and published on the project website, hosted on GitHub [58].

Type of packets defined in LaMP

The control field in a LaMP-compliant header is composed by 4 reserved bits (always set to 0xA) and by 4 bits that determine the type of packet. In total, LaMP defines 15 different packets, that are shortly described in this section.

1. *Ping-like bidirectional request* (0xA0), representing a typical ping-like request. When a client sends this kind of packet, the server shall reply with the corresponding ping-like reply; this *ping pong* mechanism is similar to that of ICMP Echo Request/Reply. The client is responsible for the insertion of the timestamp, and for its accurateness. At each transmission, the corresponding sequence number shall increase.

2. *Ping-like bidirectional reply* (0xA1), used to implement the reply to the aforementioned ping-like bidirectional request. The content of the header shall match the one of the corresponding request (with the exception of the control field). The payload can be the same, but it can also be used to transmit other user-defined data, or to perform asymmetric measurements with respect to the payload size.

3. *Ping-like bidirectional end-request* (0xA2), which is used to signal the server about the transmission of the last packet for the specific session. As the normal ping-like request, it shall include the timestamp. This packet allows the server to perform the final memory cleanup and to gracefully terminate the measurement session.

4. *Ping-like bidirectional end-reply* (0xA3), which implements the natural reply to a Ping-like bidirectional end-request, and that signal to the client that the server received the instruction to end the measurement session.

5. *Unidirectional-continue* (0xA4), is the equivalent of a ping-like request in unidirectional mode. As usual, it shall include the timestamp and it tells the server that the client will continue to generate and send packets in the future. In this case, since the measurement session is unidirectional, there is no reply from the server.

6. *Unidirectional-stop* (0xA5), used to gracefully terminate the unidirectional measurement sessions. It tells the server that no other unidirectional packets will be sent in the future.

7. *Report data* (0xA6), this packet contains, in its payload, a user-formatted report data. The report contains statistics on the current measurement session and is typically used in unidirectional mode, to transmit to the client the values measured by the server (since, in that case, all the calculation is performed at server side). This is the only packet in LaMP which compulsorily requires the payload to be filled with data.

8. *Acknowledgment* (0xA7), used as a general acknowledgment for connection initialization and report.

9. *Connection initialization* (0xA8), a packet that is used by every LaMP client to perform the initial handshake with the server. A server receiving this kind of packet is in charge of setting other session-related parameters (such as the LaMP ID), and uses the corresponding acknowledgment to communicate it to the client. Connection initialization packets have no payload, and the "*length or packet type*" is set to 0x0001 to initiate a bidirectional session and to 0x0002 to initiate a unidirectional session.

10. *Ping-like timestamp-less bidirectional request, reply, end-request, and end-reply* (0xA9, 0xAA, 0xAB, 0xAC), working exactly as the first four packets, but which do not include any timestamp. In this case the client is responsible for keeping the send timestamp into a proper internal data structure. The data structure can be successively used to compare the timestamp at which the corresponding reply is received and correctly compute the timestamp.

11. *Follow-up control* (0xAD), which is used to implement the LaMP follow-up mechanism. This mechanism allows the client to start a session where additional packets are exchanged. Beside normal requests/replies, the client and the server send follow-up packets including different type of timestamp and, possibly, additional information. A follow-up control shall include in the "*length or packet type*", the type of information to be included in the follow-up packets. The server shall accept or deny the request, by sending a follow-up control packet with "*length or packet type*" set to 0x0001 (deny), 0x0002 (accept) or 0xFFFF (badly formatted or unknown request).

12. *Follow-up data* (0xAE), which is the follow-up packet enabled by the follow-up control. It is formatted as a normal request or reply packet, but its timestamp may be generated by an alternative source (hardware timestamps, kernel-level timestamps etc.).

13. *Reserved* (0xAF), reserved to future use.

5.5 LaTe: the first LaMP-compliant application

LaMP has been used as a base to develop a client-server command line tool to measure the latency between different devices running Linux, connected by means of wireless and/or wired physical media. This tool is written in C and released as open source software under the GPLv2 license [58].

LaTe (**La**tency **Te**ster) follows all the aforementioned LaMP specifications and currently supports tests over a LaMP/UDP/IPv4 protocol stack. Nevertheless, additional protocols are going to be supported as well as new features, such as the latency measure in broadcast flooding situations, which can be of interest in the vehicular networking context.

LaTe can measure different kinds of latency: in the current version two types of measurements are supported. The first one (*User-to-user*) collects the receive timestamp as soon as the receiving entity (typically, a client receiving a reply) has completely processed the LaMP packet. The second one (which is called *KRT*, i.e., *Kernel Reception Timestamp*) obtains the reception timestamp as the time when the LaMP packet is being passed from the hardware to the kernel stack, thanks to the Linux kernel capability of generating packet timestamps. Latency measurement in LaTe may be bidirectional or unidirectional.

LaTe tries to exploit the philosophy at the base of LaMP, offering a flexible Linux tool in which various options can be specified by the user, such as the possibility to:

- Choose a transport protocol (although, as already mentioned, only UDP is supported at the time of writing).
- Choose and compare raw sockets and non-raw sockets for supported transport protocols, such as UDP.
- Test over a specific interface (wired or wireless, including loopback).
- Select an Enhanced Distributed Channel Access (EDCA) traffic class to be used in both the client and the server. This choice can be of relevance in the vehicular use case, which uses EDCA in combination with the OCB mode and some specific EDCA parameters. The option is currently supported only when a patched kernel is available (such as the one included in the OpenC2X-Embedded platform [59] or the one proposed in Section 5.2).
- Choose the frequency and amount of request packets down to a periodicity of 1 ms (between each packet).
- Specify custom LaMP payload sizes.
- Perform bidirectional and unidirectional measurements.

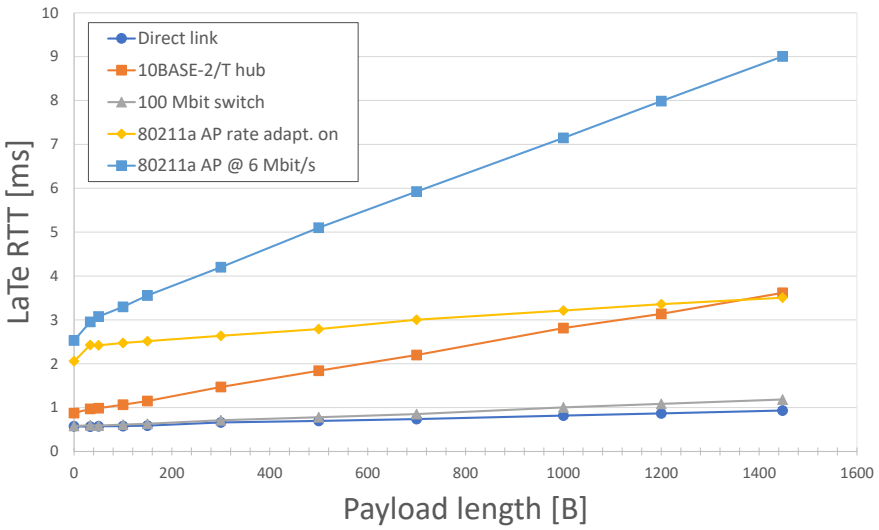


Figure 5.11: Average user-to-user latency (RTT), using normal UDP sockets, over direct Ethernet link, 10BASE-2/T hub, 100 Mbit switch and 802.11a.

All the useful data can be logged onto a separate *.csv* file for further manipulation.

Furthermore, LaTe can automatically compute the 90%, 95% and 99% confidence intervals, according to the Student's t-distribution, around the average value; this statistic is reported over the packets that are sent and received in a single test session.

In the next sections, LaMP and LaTe are validated and used to assess the latency performances of the proposed V2X platform.

5.5.1 Protocol and tool validation

The tests performed to validate LaMP and LaTe involves two laptops, one mounting an Intel Dual Band Wireless-AC NIC and the other one a Qualcomm Atheros AR9460 NIC. The connectivity between the laptops is realized with the following options: (i) directly connected through an Ethernet crossover cable; (ii) connected through a 8-port 10BASE-2/T hub; (iii) connected through a 100 Mbit/s switch; (iv) communicating over Wi-Fi with a 5 GHz 802.11a access point in between. The test were configured in the bidirectional mode, thus the measured latency can be interpreted as a Round Trip Time (RTT).

Every test lasts 60 seconds, over which the average values reported by LaTe are collected. The values are plotted as a function of the LaMP payload sizes, ranging it from 0 to 1448 B (which is the maximum supported by LaTe in order to never exceed the Ethernet MTU). The request packets are sent every 100 ms; therefore, a total of 600 packets for each test are sent. Note that, in order to obtain the full size of the transmitted packet, 24 B of the LaMP header should be added to each point.

The results, reported in Figure 5.11, fully validate the solution proposed: as expected, the higher the transmitted payload, the higher the observed RTT, because of the increase in transmission time. The measured increase is quite low when using a fast channel, i.e., a direct Ethernet connection or a 100 Mbit/s switch. Conversely, it is more evident when using a 10BASE-2/T hub, which limits the maximum transmission speed over the shared medium. The results show how communicating over a wireless, contention-based, medium causes the latency to be higher (likely due to retransmission): the slower access scheme is 802.11a @ 6 Mbit/s whereas the rate adaptation algorithm makes the system faster (note that the tests are performed in a controlled area, without interference on the selected channels).

5.6 Latency measurement in V2X-enabled embedded devices using LaTe

In the following tests LaTe is used to assess the latency performances of the proposed V2X platform, which was already evaluated (for what concerns throughput and packet loss) in Section 5.3. As in the previous analysis, the two APU1D boards are put in direct communication in the 802.11p OCB mode. Tests are performed both at a fixed distance inside the laboratory, and outdoors (both in line of sight, and in non-line of sight), by increasing the distance up to ~720 m, with a transmission power of 18 dBm. Moreover, the Access Categories tests are repeated by measuring, this time, the experienced latency when some `iPerf` interfering traffic is present; this is possible thanks to the LaTe feature enabling the selection of the AC to be used for LaMP packet transmission.

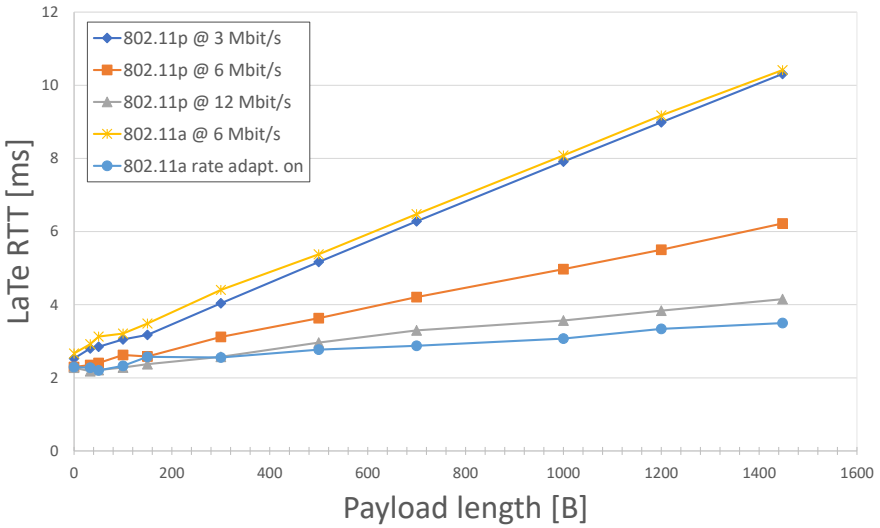


Figure 5.12: Average user-to-user latency (RTT) measured on the APU1D boards, using normal UDP sockets, over 802.11p and 802.11a.

Figure 5.12 compares the average user-to-user latency (RTT) measured with the boards in different configurations. Two scenarios are considered: (i) infrastructured 802.11a, with the boards acting as clients stations connected to an Access Point, and (ii) 802.11p, with the boards communicating in OCB mode. The measured latency values, in milliseconds, is plotted as a function of the LaMP payload length.

The results exactly match the expectations: starting from similar latency values for low payload sizes, the latency increases linearly with the payload length, and proportionally with the data rate associated with each modulation. Therefore, the higher the difference in data rate, the faster the corresponding lines tend to diverge.

The collected data show the greater performance, in terms of latency, of 802.11p with respect to 802.11a, when similar data rates are used: this is evident when comparing the 802.11p curve at 6 Mbit/s with the 802.11a one at the same data rate; having a direct communication reduces the measured latency, due to the fact that each packet can directly reach its destination, without the need of an additional hop through the AP. Interestingly, the 6 Mbit/s 802.11a curve proved to be very similar to the one related to 802.11p at 3 Mbit/s, i.e., half of the data rate. This result represents an additional validation of the platform, since each request and reply, when using 802.11a, has to pass through the Access Point, doubling the number of device-to-device transmissions. The difference between the two can be interpreted as the Access Point computation time, and shows how latency measurements can also be used to estimate internal delays.

Figure 5.13 depicts the latency measures obtained by selecting different EDCA ACs, using 802.11p at 12 Mbit/s. LaTe is set to generate and send LaMP packets with a payload of 1448 B, while the interfering `iPerf` sends 1470 B-long UDP datagrams. This plot

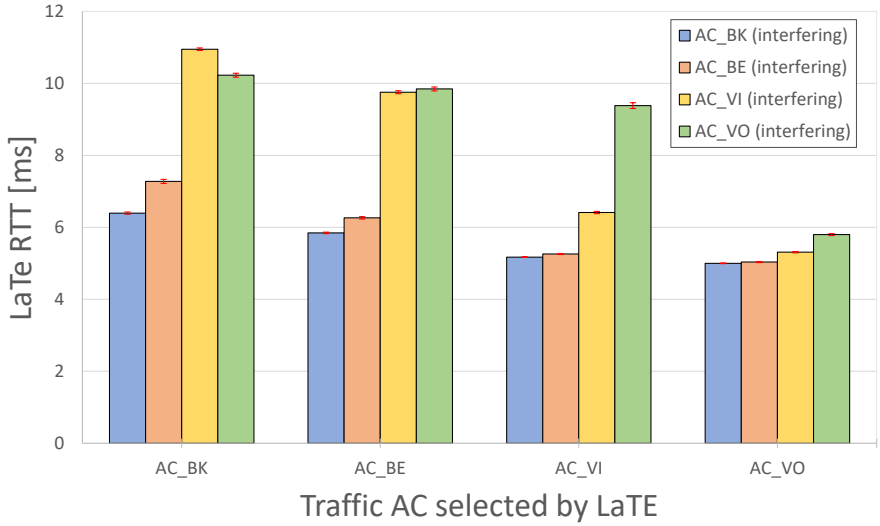


Figure 5.13: Average user-to-user latency (RTT), using normal UDP sockets, over 802.11p as a function of the AC used by LaTe. Each bar corresponds to a different Access Category in the interfering traffic generated by `iPerf`.

shows the average user-to-user latency (RTT) as a function of the Access Category used by LaTe. Every section of the histogram plot is then divided into four distinct bars, each showing the traffic class used by `iPerf` to generate the parallel interfering traffic. Each single test, providing the average latency over 600 packets, is performed 15 times. The average value among these attempts is then considered as reference. The 95% confidence intervals around the average value are highlighted in red, even though they are quite small, as all the tests provided similar results.

Also in this case the measured values are coherent with the theory, and show how increasing the AC of the interfering traffic causes the measured latency to increase as well. Additionally, it is possible to highlight how using a high-priority AC, such as AC_VO or AC_VI, allows the sending device to experience a lower latency as opposed to when AC_BK or AC_BE are selected. When testing the latency of AC_BK it was possible, however, to highlight a marginal decreasing trend, in the moment in which `iPerf` transmits an interfering traffic over the *Video* and *Voice* traffic classes. This can be explained by observing that the channel usage is quite unbalanced towards `iPerf`, which generates a higher amount of prioritized traffic than LaTe.

The next set of results have been obtained by performing outdoor tests in LOS and in NLOS. Two vehicles are equipped with the APU boards transmitting at 6 Mbit/s: one vehicle is stationary while the other is moving away to ~720 m away, with a speed between 10 and 15 km/h. The test is performed in LOS, without any kind of obstacle between the two vehicles, and in NLOS, with the stationary vehicle behind a reinforced concrete wall. The boards are equipped with two antennas with 6 dBi gain, and transmit with a power of 18 dBm. The metrics collected are: RTT using LaTe, throughput using `iPerf` and RSSI (Received Signal Strength Indicator), obtained through the information reported by the operating system of the two boards.

Figure 5.14 shows the results measured with LaTe: the plot suggests that, for the LOS situation, the two boards maintain a stable RTT up to 500 m. Then, due to the

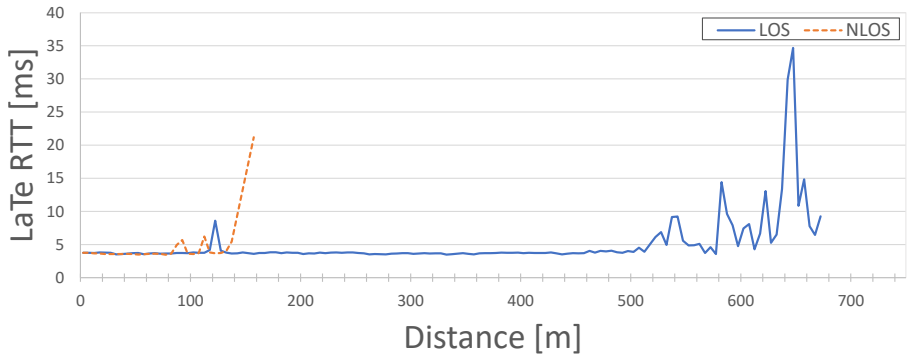


Figure 5.14: User-to-user latency (RTT) measured in LOS and in NLOS, using normal UDP sockets, over 802.11p as a function of the distance between the two vehicles mounting the 802.11p boards.

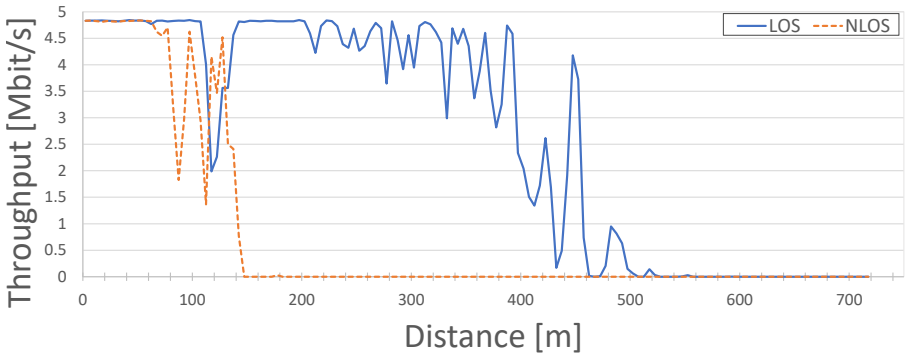


Figure 5.15: Throughput measured in LOS and in NLOS using `iPerf`, offering 100 Mbit/s with a payload of 1470 B over 802.11p as a function of the distance between the two vehicles mounting the 802.11p boards.

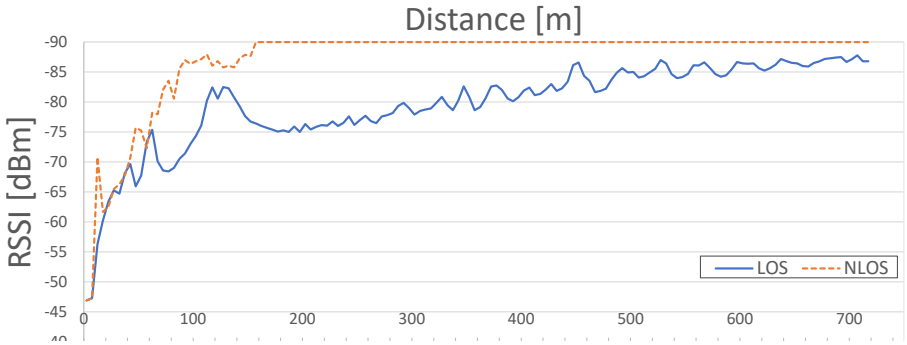


Figure 5.16: RSSI measured in LOS and in NLOS as a function of the distance between the two vehicles mounting the 802.11p boards.

high number of lost packets and to the overall degradation of the connection, the RTT increases (reaching peaks of 35 ms). In the LOS case, the LaTe communication is lost around 680 m. The NLOS results instead show that the two boards are able to keep the connection up to ~150 m. Then, at higher distances the LaTe client and server are not able to communicate anymore, due to the high number of packets lost.

The results of the throughput computed using `iPerf` are shown in Figure 5.15. In this case, `iPerf` is set to push 100 Mbit/s of UDP traffic with payload of 1470 B. As in the previous case, in NLOS the maximum distance reached by two APU boards before the `iPerf` connection is lost is less than 200 m. In LOS, instead, the throughput oscillates between the maximum value (i.e., 4.86 Mbit/s, as measured in Figure 5.3b with UDP payload of 1470 B) and values that are constantly lower as the distance increases. Then, at around 520 m, `iPerf` loses the connection and consequently the throughput drops to 0 Mbit/s.

Another experiment that analyzes the quality of the signal between the two boards

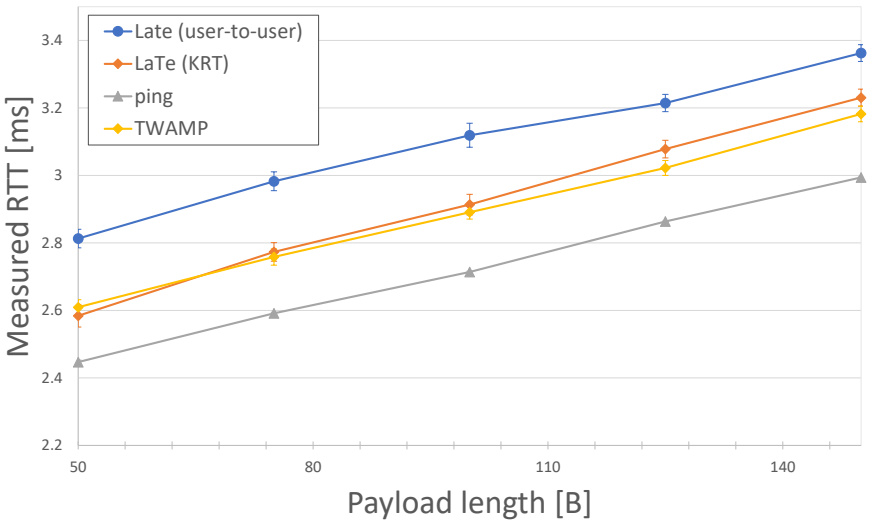


Figure 5.17: Average user-to-user latency (RTT), measured on the boards, averaged over 20 tests, using normal UDP sockets, over 802.11p at 3 Mbit/s. 95% confidence intervals are represented too.

is shown in Figure 5.16. In this case, instead of relying on the metrics returned by a particular application, the results are obtained by extracting the information available from the WNICs driver, using the command `iw dev <interface_name> station dump`. Indeed, during the first two experiments, even if the applications measuring RTT and throughput (i.e., LaTe and `iPerf`) lost their connection, in fact the two boards were able to communicate, albeit weakly. The results show that, in LOS, the two boards are still connected also at 720 m of distance, but unfortunately the RSSI is too low (less than -85 dBm) to allow any kind of application to exchange information. In NLOS, instead, the connection is completely lost at 180 m, where the signal drops to -90 dBm, meaning that the received signal is null.

The final test is performed to compare the results obtained by means of `ping` with the ones gathered through LaTe and by a cross-compiled version of `twping`, an open source implementation of TWAMP included in the perfSONAR project [79]. The results are shown in Figure 5.17; the two boards communicate over 802.11p at a physical rate of 3 Mbit/s. Interestingly, the results confirm the previous assumption on `ping`: being it a protocol operating at the lower layer of the communication stack, the measured latency will not include a part of the computational time required by the entity to pass the information to upper layers; therefore, the measured latency will be lower. The values reported by `ping` give a more precise estimate of the network latency, but not of the application layer one, which is the latency a user would really experience when using UDP to communicate with a given application.

Furthermore, this plot also compares the two latency types provided by LaTe, showing how using a *KRT latency* normally results in lower measured values, as the client-side time needed by the kernel (and then by LaTe itself) to handle each reply is not considered.

TWAMP instead provides comparable results with respect to the ones yielded by LaTe in KRT mode. This comparison can actually be performed as TWAMP test packets are encapsulated inside UDP, which is also the case of the LaMP packets managed by LaTe.

Chapter 6

Conclusions

The myriad of applications enabled by VANETs will completely change the way in which mobility is currently intended. All the safety mechanisms already present will leverage the information coming from the radio interfaces, and at the same time new, innovative services will be launched thanks to the introduction of these technologies.

The first part of this thesis gives an overview of the standardization efforts that have been made by the VANET community during the last decade, by showing the high-level structure of the two main communication stacks: ITS-G5 and WAVE; the two access technologies (IEEE 802.11p and 3GPP C-V2X) are introduced and discussed as well.

The importance of simulation tools for the testing of applications leveraging vehicular communications, is what has driven the development of the ns-3 framework named MS-VAN3T, introduced and discussed in Chapter 3. The proposed model, which at the time of writing is still being actively developed, provides the scientific community with the first implementation of ETSI ITS-G5 in the ns-3 simulator. The peculiarity of MS-VAN3T resides also in the number of access technologies that can be used to enable the overlying applications, since it comes with the possibility of transparently switch between 802.11p, C-V2X in transmission mode 4 and LTE. MS-VAN3T is provided with a couple of applications, namely the *area speed advisor* and the *emergency vehicle alert*, which can be used as development baseline. Additionally, MS-VAN3T can be used as a V2X message emulator, thanks to the integration with `fd-net-devices`. In this thesis, the framework is also used to test the network performances of the available access technologies; the tests highlighted the advantages, in terms of delay, of 802.11p with respect to C-V2X or LTE. The results, however, show that 802.11p is less stable as the bit rate increases (due to the MCS adopted), with very high jitter and non-negligible amount of packets lost. The solutions coming from the cellular world, namely LTE and C-V2X, exhibit instead a stabler behavior.

MS-VAN3T is also the simulation framework used to develop the Collision Avoidance Service (CAS), a system that, based on the Intersection Collision Risk Warning applications defined by ETSI, provides a way to warn the drivers or the vulnerable users about possible future collisions. CAS leverages the exchange of Cooperative Awareness

Messages (CAMs), that are periodically broadcasted by road entities and that include information about speed, position, heading and acceleration. The core of CAS is the so called Collision Avoidance Algorithm (CAA), which gathers all the information coming from CAMs and which builds future trajectories. The trajectories are then analyzed and, in case a collision is detected, CAA instructs the CAS to generate a Decentralized Environmental Notification Message (DENM) to be sent to the involved entities.

In this thesis, CAS is presented in a centralized and in a distributed architecture, and is tested through several simulation campaigns, which pointed out the effectiveness of the system in promptly warn the drivers about future collisions.

The CAS system, in its initial definition, builds a passive safety layer in which the information about collisions are encoded and sent through DENMs. *How* this information is used at the receivers depends on several factors, including the type of entity, the severity of the situation, the current speed and so on. This thesis proposes a novel system that can be used to automatically modify the vehicles' trajectory to avert the collisions. This system leverages the information generated by CAS, and overrides the vehicles' control by reducing the speed by a factor that is proportional to the gravity of the situation advertised by the corresponding DENM. Simulation results helped to correctly tune the proposed solution, and demonstrated that such systems can prevent the totality of collisions in the tested scenario.

Beside the work on MS-VAN3T and CAS, this thesis presents the efforts that have been made for the realization of an open source platform to enable V2X communications in commercial embedded devices. The resulting solution is used to assess the performance of a couple of 802.11p WNICs. The need for precise measurements of the latency that an application running on top of the V2X platform would experience, driven the definition of a novel protocol for micro-second precise latency measurement. The protocol, named LaMP (Latency Measurement Protocol), defines a LaMP PDU (Protocol Data Unit) and the general guidelines that LaMP-compliant applications should follow to generate timestamps, to compute latency and so on. Therefore, the first application entirely based on LaMP, named LaTe (Latency Tester), is used to test the V2X platform and to validate the proposed solution from the point of view of the latency experience at application layer.

All the applications and solutions proposed in this thesis highlighted the importance that VANETs will have in the future of mobility. The next generation of vehicles will see the number of services enabled by V2X communication to grow rapidly. For this reasons, it is important to provide the scientific community with tools like MS-VAN3T and LaTe, which can be used to prototype and test the applications in a safe environment, with detailed models that closely mimic the real behaviors of road players.

Appendix A

CA and DEN basic service models in MS-VAN3T

MS-VAN3T implements a model of the Facilities layers as defined in EN 302 637-2 and EN 302 637-3 [26, 27]. The two entities standardized by ETSI, namely CA (Cooperative Awareness) and DEN (Decentralized Environmental Notification) basic services, are in charge of managing V2X messages, from the creation to the final delivery at Applications layer. The CA basic service is in charge of managing the Cooperative Awareness Messages (CAMs), which are periodically exchanged between ITS-S (ITS-Stations) to create and maintain a distributed awareness of the surrounding area. The content of CAMs depends on the originating ITS-S type: in case of a vehicle the information includes time, position, motion state, etc. The DEN basic service is instead in charge of managing Decentralized Environmental Messages (DENMs), that are asynchronous messages used to signal and describe a variety of situations and events that may have an impact on the receivers' experience.

The next sections will be devoted to the description of CA and DEN basic services features included in MS-VAN3T.

A.1 CA basic service implementation

The CA basic service is the Facilities layers entity operating the CAM protocol. The two main services it provides are sending and receiving of CAMs. The following subfunctions have been developed in MS-VAN3T to support the required protocol operations:

- CAM encoding, using the primitives created through the integration with *asn1c* [91], and including the most updated information coming from the Vehicle Data Provider (VDP);
- CAM decoding, using the primitives created through the integration with *asn1c*;

- CAM Transmission Management, implementing the protocol operation of the originating ITS-S. In particular, this sub-function includes the tools to activate and terminate the CAM transmission, to determinate the CAM generation frequency and to trigger the generation of a CAM.
- CAM Reception Management, implementing the protocol operation for receiving ITS-S. This sub-function triggers the decoding of CAMs, and provides the received CAM data to the overlying ITS applications.

A.1.1 CAM encoding

As introduced in Section 2.2.2 and depicted in Figure 2.7, a CAM is composed of a common ITS PDU header and multiple containers. For a vehicular originating ITS-S, a CAM is compulsorily composed by at least one basic container, including basic information related to the originating station, and one high frequency container, containing highly dynamic information of the originating station (position, speed, acceleration, heading, etc.). Optionally, a CAM may include one low frequency container, containing static and not highly dynamic information, and one or more special containers, containing information for non-ordinary vehicles.

The C++ code developed to model the CAMs encoding and transmission is reported below.

```
  CABasicService_error_t
CABasicService::generateAndEncodeCam()
{
   CAM_t *cam;
   CURRENT_VDP_TYPE::CAM_mandatory_data_t cam_mandatory_data;
   CABasicService_error_t errval=CAM_NO_ERROR;

   // Optional CAM data pointers
   AccelerationControl_t *accelerationcontrol=NULL;
   LanePosition_t *laneposition=NULL;
   SteeringWheelAngle_t *steeringwheelangle=NULL;
   LateralAcceleration_t *lateralacceleration=NULL;
   VerticalAcceleration_t *verticalacceleration=NULL;
   PerformanceClass_t *performanceclass=NULL;
   CenDsrcTollingZone_t *tollingzone=NULL;

   RSUContainerHighFrequency_t* rsu_container=NULL;

   Ptr<Packet> packet;
   asn_encode_to_new_buffer_result_t encode_result;
   int64_t now;

   if(m_vehicle==false)
   {
```

```
25      rsu_container=m_vdp.getRsuContainerHighFrequency();
26
27      if(rsu_container==NULL)
28      {
29          LOG_ERROR("Cannot send RSU CAM: the current VDP does not ↩
                provide any RSU High Frequency Container.");
30          return CAM_NO_RSU_CONTAINER;
31      }
32  }
33
34  /* Collect data for mandatory containers */
35  cam=(CAM_t*) calloc(1, sizeof(CAM_t));
36  if(cam==NULL)
37  {
38      return CAM_ALLOC_ERROR;
39  }
40
41  cam_mandatory_data=m_vdp.getCAMMandatoryData();
42
43  /* Fill the header */
44  cam->header.messageID = FIX_CAMID;
45  cam->header.protocolVersion = FIX_PROT_VERS;
46  cam->header.stationID = m_station_id;
47
48  /*
49  * Compute the generationDeltaTime, "computed as the time ↩
        corresponding to the
50  * time of the reference position in the CAM, considered as time ↩
        of the CAM generation.
51  * The value of the generationDeltaTime shall be wrapped to 65 ↩
        536. This value shall be set as the
52  * remainder of the corresponding value of TimestampIts divided by↩
         65 536 as below:
53  * generationDeltaTime = TimestampIts mod 65 536"
54  */
55  cam->cam.generationDeltaTime = compute_timestampIts (m_real_time)↩
        % 65536;
56
57  /* Fill the basicContainer's station type */
58  cam->cam.camParameters.basicContainer.stationType = m_stationtype↩
        ;
59
60  if(m_vehicle==true)
61  {
62      /* Fill the basicContainer */
63      cam->cam.camParameters.basicContainer.referencePosition.↩
            altitude = cam_mandatory_data.altitude;
64      cam->cam.camParameters.basicContainer.referencePosition.↩
            latitude = cam_mandatory_data.latitude;
```

```
65      cam->cam.camParameters.basicContainer.referencePosition.↩
            longitude = cam_mandatory_data.longitude;
66      cam->cam.camParameters.basicContainer.referencePosition.↩
            positionConfidenceEllipse = cam_mandatory_data.↩
            posConfidenceEllipse;
67
68      /* Fill the highFrequencyContainer */
69      cam->cam.camParameters.highFrequencyContainer.present = ↩
            HighFrequencyContainer_PR_basicVehicleContainerHighFrequency↩
            ;
70      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.heading = ↩
            cam_mandatory_data.heading;
71      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.speed = ↩
            cam_mandatory_data.speed;
72      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.driveDirection = ↩
            cam_mandatory_data.driveDirection;
73      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.vehicleLength = ↩
            cam_mandatory_data.VehicleLength;
74      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.vehicleWidth = ↩
            cam_mandatory_data.VehicleWidth;
75      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.longitudinalAcceleration↩
             = cam_mandatory_data.longAcceleration;
76      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.curvature = ↩
            cam_mandatory_data.curvature;
77      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.curvatureCalculationMode↩
             = cam_mandatory_data.curvature_calculation_mode;
78      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.yawRate = ↩
            cam_mandatory_data.yawRate;
79
80      // Manage optional data
81      accelerationcontrol = m_vdp.getAccelerationControl();
82      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.accelerationControl = ↩
            accelerationcontrol;
83
84      laneposition = m_vdp.getLanePosition();
85      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.lanePosition = ↩
            laneposition;
86
87      steeringwheelangle = m_vdp.getSteeringWheelAngle();
```

```
88      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.steeringWheelAngle = ↩
            steeringwheelangle;
89
90      lateralacceleration=m_vdp.getLateralAcceleration();
91      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.lateralAcceleration = ↩
            lateralacceleration;
92
93      verticalacceleration=m_vdp.getVerticalAcceleration();
94      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.verticalAcceleration = ↩
            verticalacceleration;
95
96      performanceclass=m_vdp.getPerformanceClass();
97      cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.performanceClass = ↩
            performanceclass;
98
99      tollingzone=m_vdp.getCenDsrcTollingZone();
100     cam->cam.camParameters.highFrequencyContainer.choice.↩
            basicVehicleContainerHighFrequency.cenDsrcTollingZone = ↩
            tollingzone;
101   }
102   else
103   {
104     /* Fill the basicContainer */
105     /* There is still no full RSU support in this release */
106     cam->cam.camParameters.basicContainer.referencePosition.↩
            altitude.altitudeConfidence = AltitudeValue_unavailable;
107     cam->cam.camParameters.basicContainer.referencePosition.↩
            altitude.altitudeValue = AltitudeValue_unavailable;
108     cam->cam.camParameters.basicContainer.referencePosition.↩
            latitude = Latitude_unavailable;
109     cam->cam.camParameters.basicContainer.referencePosition.↩
            longitude = Longitude_unavailable;
110     cam->cam.camParameters.basicContainer.referencePosition.↩
            positionConfidenceEllipse.semiMajorConfidence = ↩
            SemiAxisLength_unavailable;
111     cam->cam.camParameters.basicContainer.referencePosition.↩
            positionConfidenceEllipse.semiMinorConfidence = ↩
            SemiAxisLength_unavailable;
112     cam->cam.camParameters.basicContainer.referencePosition.↩
            positionConfidenceEllipse.semiMajorOrientation = ↩
            HeadingValue_unavailable;
113     /* Fill the highFrequencyContainer */
114     cam->cam.camParameters.highFrequencyContainer.present = ↩
            HighFrequencyContainer_PR_rsuContainerHighFrequency;
115     cam->cam.camParameters.highFrequencyContainer.choice.↩
            rsuContainerHighFrequency = *rsu_container;
```

```
116     }
117
118     // Store all the "previous" values used in checkCamConditions()
119     m_prev_distance=m_vdp.getTravelledDistance ();
120     m_prev_speed=m_vdp.getSpeedValue ();
121     m_prev_heading=m_vdp.getHeadingValue ();
122
123     LowFrequencyContainer_t *lowfrequencycontainer=m_vdp.↩
            getLowFrequencyContainer();
124
125     if(lowfrequencycontainer!=NULL)
126     {
127         // Send a low frequency container only if at least 500 ms have↩
                passed since the last CAM with a low frequency container
128         if(lastCamGenLowFrequency==-1 ||(computeTimestampUInt64 ()-↩
                lastCamGenLowFrequency)>=500)
129         {
130             cam->cam.camParameters.lowFrequencyContainer = ↩
                    lowfrequencycontainer;
131             lastCamGenLowFrequency=computeTimestampUInt64 ();
132         }
133     }
134
135     SpecialVehicleContainer_t *specialvehiclecontainer=m_vdp.↩
            getSpecialVehicleContainer();
136
137     if(specialvehiclecontainer!=NULL)
138     {
139         // Send a low frequency container only if at least 500 ms have↩
                passed since the last CAM with a low frequency container
140         if(lastCamGenSpecialVehicle==-1 ||(computeTimestampUInt64 ()-↩
                lastCamGenSpecialVehicle)>=500)
141         {
142             cam->cam.camParameters.specialVehicleContainer = ↩
                    specialvehiclecontainer;
143             lastCamGenSpecialVehicle=computeTimestampUInt64 ();
144         }
145     }
146
147     /* Construct CAM and pass it to the lower layers (now UDP, in the↩
            future BTP and GeoNetworking, then UDP) */
148     /** Encoding **/
149     char errbuff[ERRORBUFF_LEN];
150     size_t errlen=sizeof(errbuff);
151
152     if(asn_check_constraints(&asn_DEF_CAM,(CAM_t *)cam,errbuff,&↩
            errlen) == -1) {
153         LOG_ERROR("Unable to validate the ASN.1 contraints for the ↩
                current CAM."<<std::endl);
154         LOG_ERROR("Details: " << errbuff << std::endl);
```

```
155         errval=CAM_ASN1_UPER_ENC_ERROR;
156         goto error;
157     }
158
159     encode_result = asn_encode_to_new_buffer(NULL,↩
        ATS_UNALIGNED_BASIC_PER,&asn_DEF_CAM, cam);
160     if (encode_result.result.encoded==-1)
161     {
162         errval=CAM_ASN1_UPER_ENC_ERROR;
163         goto error;
164     }
165
166     packet = Create<Packet> ((uint8_t*) encode_result.buffer, ↩
        encode_result.result.encoded+1);
167     if(m_socket_tx->Send (packet)==-1)
168     {
169         errval=CAM_CANNOT_SEND;
170         goto error;
171     }
172
173     m_cam_sent++;
174
175     // Store the time in which the last CAM (i.e. this one) has been ↩
        generated and successfully sent
176     now=computeTimestampUInt64 ()/NANO_TO_MILLI;
177     m_T_GenCam_ms=now-lastCamGen;
178     lastCamGen = now;
179
180     error:
181     // Free all the previously allocated memory
182     if(m_vehicle==true)
183     {
184         // After encoding, we can free the previosly allocated ↩
            optional data
185         if(accelerationcontrol) m_vdp.vdpFree(accelerationcontrol);
186         if(laneposition) m_vdp.vdpFree(laneposition);
187         if(steeringwheelangle) m_vdp.vdpFree(steeringwheelangle);
188         if(lateralacceleration) m_vdp.vdpFree(lateralacceleration);
189         if(verticalacceleration) m_vdp.vdpFree(verticalacceleration);
190         if(performanceclass) m_vdp.vdpFree(performanceclass);
191         if(tollingzone) m_vdp.vdpFree(tollingzone);
192     }
193     else
194     {
195         if(rsu_container) m_vdp.vdpFree(rsu_container);
196     }
197
198     if(lowfrequencycontainer) m_vdp.vdpFree(lowfrequencycontainer);
199     if(specialvehiclecontainer) m_vdp.vdpFree(specialvehiclecontainer↩
        );
```

```
200
201     if(cam) free(cam);
202
203     return errval;
204 }
```

The CAM information is included in the `CAM_t` object, a type that is generated automatically by feeding *asn1c* with the ASN.1 definition of CAM (taken, for example, by the ASN.1 definition present in [26]). From Line 23 to Line 145, the mandatory and optional containers are filled. Once the `CAM_t` is ready, in Line 159 the *asn1c* function `asn_encode_to_new_buffer()` is used to encode the CAM inside a buffer. The same buffer is finally used to create the packet (Line 166) to be sent to the UDP socket (Line 167). As soon as the BTP and GeoNetworking modules will be available, this part will be changed, and the object of type `CAM_t` will be delivered to the Networking and Transport layer.

A.1.2 CAM Transmission Management

The CAM Transmission Management is in charge of managing everything related to the CAM generation. The CAM generation frequency in not fixed and is thought to reflect the kinetics properties of the originating entity: the faster the entity changes its state, the higher the generation frequency will be. In this way, a slow or still originating entity will not occupy radio resources to broadcast information about its state.

The algorithm developed in ns-3 for MS-VAN3T is described and reported below, with part of the description taken from [26].

The standard sets the maximum frequency to 10 Hz (i.e., one CAM each 100 ms, defined in the algorithm as *T_GenCamMin*), and the minimum to 1 Hz (i.e., one CAM every second, defined in the algorithm as *T_GenCamMax*). Within these limits, the CAM generation is triggered depending on the originating ITS-S dynamics and on the channel congestion (at the time of writing, however, MS-VAN3T accounts only for the dynamics of the originating entity to determine the CAM frequency).

The condition triggering the CAM are checked every *T_CheckCamGen*, a value that should be always less than or equal to *T_GenCamMin*.

The current upper limit for the CAM generation interval is defined as *T_GenCam*, that defaults its value to *T_GenCamMax*. The conditions that may trigger the generation of a CAM are two:

1. One of the following dynamics conditions are verified in the originating ITS-S:
    - the absolute difference between the current heading and the heading included in the CAM previously transmitted exceeds 4°;
    - the distance between the current position and the position included in the CAM previously transmitted exceeds 4 m;

- the absolute difference between the current speed and the speed included in the CAM previously transmitted exceeds 0,5 m/s.

2. The time elapsed since the last CAM generation is equal or greater than *T_GenCam*.

Whenever a CAM is triggered due to condition 1), *T_GenCam* is set to the time elapsed since the last CAM generation. After triggering a number equal to *N_GenCam* CAMs due to condition 2), *T_GenCam* is set to *T_GenCamMax*. The value of *N_GenCam* defaults to 3.

The C++ code developed to model the CAMs dynamic generation is reported below.

```
1  void
2  CABasicService::checkCamConditions()
3  {
4     int64_t now=computeTimestampUInt64 ()/NANO_TO_MILLI;
5     CABasicService_error_t cam_error;
6     bool condition_verified=false;
7     static bool dyn_cond_verified=false;
8
9     // If no initial CAM has been triggered before checkCamConditions↩
         () has been called, throw an error
10    if(m_prev_heading==-1 || m_prev_speed==-1 || m_prev_distance==-1)
11    {
12       FATAL_ERROR("Error. checkCamConditions() was called before ↩
            sending any CAM and this is not allowed.");
13    }
14    /*
15    * ETSI EN 302 637-2 V1.3.1 chap. 6.1.3 condition 1) (no DCC)
16    * One of the following ITS-S dynamics related conditions is given↩
         :
17    */
18
19    /* 1a)
20    * The absolute difference between the current heading of the ↩
         originating
21    * ITS-S and the heading included in the CAM previously ↩
         transmitted by the
22    * originating ITS-S exceeds °4;
23    */
24    double head_diff = m_vdp.getHeadingValue () - m_prev_heading;
25    head_diff += (head_diff>180.0) ? -360.0 : (head_diff<-180.0) ? ↩
         360.0 : 0.0;
26    if (head_diff > 4.0 || head_diff < -4.0)
27    {
28       cam_error=generateAndEncodeCam ();
29       if(cam_error==CAM_NO_ERROR)
30       {
31          m_N_GenCam=0;
32          condition_verified=true;
```

```
33          dyn_cond_verified=true;
34       } else {
35          LOG_ERROR("Cannot generate CAM. Error code: "<<cam_error);
36       }
37    }
38
39    /* 1b)
40    * the distance between the current position of the originating ↩
         ITS-S and
41    * the position included in the CAM previously transmitted by the ↩
         originating
42    * ITS-S exceeds 4 m;
43    */
44    double pos_diff = m_vdp.getTravelledDistance () - m_prev_distance↩
         ;
45    if (!condition_verified && (pos_diff > 4.0 || pos_diff < -4.0))
46    {
47       cam_error=generateAndEncodeCam ();
48       if(cam_error==CAM_NO_ERROR)
49       {
50          m_N_GenCam=0;
51          condition_verified=true;
52          dyn_cond_verified=true;
53       } else {
54          LOG_ERROR("Cannot generate CAM. Error code: "<<cam_error);
55       }
56    }
57
58    /* 1c)
59    * the absolute difference between the current speed of the ↩
         originating ITS-S
60    * and the speed included in the CAM previously transmitted by the↩
          originating
61    * ITS-S exceeds 0,5 m/s.
62    */
63    double speed_diff = m_vdp.getSpeedValue () - m_prev_speed;
64    if (!condition_verified && (speed_diff > 0.5 || speed_diff < ↩
         -0.5))
65    {
66       cam_error=generateAndEncodeCam ();
67       if(cam_error==CAM_NO_ERROR)
68       {
69          m_N_GenCam=0;
70          condition_verified=true;
71          dyn_cond_verified=true;
72       } else {
73          LOG_ERROR("Cannot generate CAM. Error code: "<<cam_error);
74       }
75    }
76
```

```
77      /* 2)
78      * The time elapsed since the last CAM generation is equal to or ↩
            greater than T_GenCam
79      */
80      if(!condition_verified && (now-lastCamGen>=m_T_GenCam_ms))
81      {
82          cam_error=generateAndEncodeCam ();
83          if(cam_error==CAM_NO_ERROR)
84          {
85
86              if(dyn_cond_verified==true)
87              {
88                  m_N_GenCam++;
89                  if(m_N_GenCam>=m_N_GenCamMax)
90                  {
91                      m_N_GenCam=0;
92                      m_T_GenCam_ms=T_GenCamMax_ms;
93                      dyn_cond_verified=false;
94                  }
95              }
96          } else {
97              LOG_ERROR("Cannot generate CAM. Error code: "<<cam_error);
98          }
99      }
100
101     m_event_camCheckConditions = Simulator::Schedule (MilliSeconds(↩
            m_T_CheckCamGen_ms), &CABasicService::checkCamConditions, this↩
            );
102 }
```

From Line 24 to Line 75 the code checks the dynamics of the vehicle and verifies the constraints specified in condition 1). The vehicle's status information are collected through the VDP, which in our case is configured to be connected to TraCI (thus, to SUMO). From Line 80 to Line 99, instead, the code checks the time at which the last CAM was generated, and if greater than *T_GenCamMax* it generates a new CAM, as specified in condition 2). The last line schedules the next iteration of the `checkCamConditions()` routine after *T_CheckCamGen*.

A.1.3 CAM decoding and Reception Management

Upon the reception of a CAM, the CA basic service is required to decode the information and to make the content of the CAM available to the requesting ITS applications. The standard does not specify the interface over which the information is transmitted from the CA basic service to the ITS applications. For this reason, MS-VAN3T implements a callback mechanism, that allows the CAM data to be efficiently delivered to the upper layers.

The C++ code developed to model the CAMs decoding and transmission to the ITS

application is reported below.

```
1  void
2  CABasicService::receiveCam (Ptr<Socket> socket)
3  {
4     Ptr<Packet> packet;
5     CAM_t *decoded_cam;
6     Address from;
7
8     packet = socket->RecvFrom (from);
9
10    uint8_t *buffer = new uint8_t[packet->GetSize ()];
11    packet->CopyData (buffer, packet->GetSize ()-1);
12
13
14    /* Try to check if the received packet is really a DENM */
15    if (buffer[1]!=FIX_CAMID)
16    {
17       LOG_ERROR("Warning: received a message which has messageID '"<<buffer[1]<<"' but '2' was expected.");
18       return;
19    }
20
21    /** Decoding **/
22    void *decoded_=NULL;
23    asn_dec_rval_t decode_result;
24
25    do {
26       decode_result = asn_decode(0, ATS_UNALIGNED_BASIC_PER, &asn_DEF_CAM, &decoded_, buffer, packet->GetSize ()-1);
27    } while(decode_result.code==RC_WMORE);
28
29    if(decode_result.code!=RC_OK || decoded_==NULL) {
30       LOG_ERROR("Warning: unable to decode a received CAM.");
31       return;
32    }
33
34    decoded_cam = (CAM_t *) decoded_;
35
36    m_CAReceiveCallback(decoded_cam,from);
37 }
```

In this version, the packet is directly received with a UDP socket at the Facilities layer (Line 8). In future versions, when BTP and GeoNetworking will be available, the packet will be received by the lower layers and forwarded to the CA basic service. From Line 15 the packet is checked and decoded using the *asn1c* function `asn_decode()`. The decoded CAM is then forwarded to the ITS application using a dedicated callback, as shown in Line 36.

The callback is defined inside the CA basic service as:

```
1   std::function<void(CAM_t *, Address)> m_CAReceiveCallback;
```

And configured thanks to the public method `addCARxCallback()`:

```
1 void addCARxCallback(std::function<void(CAM_t *, Address)> ↩
    rx_callback) {m_CAReceiveCallback=rx_callback;}
```

At this point, the overlying application which is interested in receiving the CAM information, can simply enroll one of its function (in this example, AppServer enrolls the function `receiveCAM`) to the aforementioned callback:

```
1   m_caService.addCARxCallback (std::bind(&appServer::receiveCAM,↩
      this,std::placeholders::_1,std::placeholders::_2));
```

A.2 DEN basic service implementation

The DEN basic service is the Facilities layer entity operating the DEN protocol. It is designed to give the possibility to originating ITS applications to trigger, update and terminate the transmission of DENMs. At DENM reception, it provides the overlying applications with the requested data. Due to the complexity and variety of different events that must be characterized, the standard defines 4 types of DENM messages:

- *New DENM*, a DENM that is generated when the overlying ITS application detects a triggering event for the first time. Each *new DENM* is associated with an identifier, called *actionID*.
- *Update DENM*, a DENM that updates the information contained in a previous DENM. The updated information are transmitted by the same originating ITS application that generated the first DENM.
- *Cancellation DENM*, a DENM informing about the termination of an event. This type of DENM is generated by the originating ITS applications that triggered the first DENM.
- *Negation DENM*, a DENM that has the same functionalities of the *Cancellation DENM*, but that can be generated by a third ITS application, which may have noticed that the event present in the original DENM is not valid anymore.

As in the CA basic service case, MS-VAN3T implements the following sub-functions to support the required protocol operations:

- DENM encoding, using the primitives created through the integration with *asn1c*, and by relying on the information coming from the upper layers;
- DENM decoding, using the primitives created through the integration with *asn1c*;
- DENM Transmission Management, implementing the DENM generation protocol. It provides functions to generate *New DENMs*, *Update DENMs* and *Cancellation/Negation DENMs*, as well as the possibility to send DENMs with fixed frequency for a certain amount of time.
- DENM Reception Management, implementing the protocol for DENMs reception. It provides functions to check the DENMs validity, to update specific message tables, and to transmit the required information to the overlying ITS applications.

A.2.1 DENM encoding and Transmission Management

Differently from the CA basic service, which autonomously generates, encode and transmit CAMs, in DEN basic service DENMs are triggered by the overlying ITS applications. For this reason, in [27] ETSI defines the interface over which the information should be exchanged between applications and Facilities layers. The interface and the function vary depending on the type of DENM to be transmitted: to generate a *New DENM* the `AppDENM_trigger()` function is called, for an *Update DENM* the

Category	Data	Definition
Data from application to DEN basic service	Event detection time	Position of the originating ITS-S when the triggering event is first detected
	Event position	Position of the event
	Event validity duration	Validity duration of the event (optional)
	Repetition duration	DENM repetition duration (optional)
	Transmission interval	Interval for the DENM transmission (optional)
	Repetition interval	Interval of DENM repetition (optional)
	Situation container information	Information to be encoded in the Situation container (optional)
	Location container information	Information to be encoded in the Location container (optional)
	À la carte container information	Information to be encoded in the À la carte container (optional)
	Relevance area of the event	Area around which the event is relevant (optional)
	Destination area	Destination area (to be passed to BTP and Geonetworking)
	Traffic class	Geonetworking traffic class
Data returned from DEN basic service to application	actionID	ID identifying the DENM
	Failure notification	Failure notification (in case of errors in the DENM creation)

Table A.1: Data passed from the applications to `AppDENM_trigger()` for the generation of *New DENM*

Category	Data	Definition
Data from application to DEN basic service	actionID	actionID of the DENM to be terminated
	Event termination detection time	Time at which the event is detected to be terminated
	Event position	Original position of the event
	Event validity duration	Validity of the termination information
	Repetition duration	DENM repetition duration (optional)
	Transmission interval	Interval for the DENM transmission (optional)
	Repetition interval	Interval of DENM repetition (optional)
	Relevance area of the event	Area around which the event is relevant (optional)
	Destination area	Destination area (to be passed to BTP and Geonetworking)
	Traffic class	Geonetworking traffic class
Data returned from DEN basic service to application	actionID	ID identifying the DENM
	Failure notification	Failure notification (in case of errors in the DENM creation)

Table A.2: Data passed from the applications to the Facilities layers for the generation of a *Cancellation/Termination DENM*

`AppDENM_update()` function is called, and finally *Cancellation/Negation DENM* are generated by the `AppDENM_terminate()` function.

In Table A.1 and A.2, the interfaces used for the interaction between applications and DEN basic service to trigger and terminate DENMs are reported. In case an *Update DENM* is generated, the interface used is the same as in Table A.1 but containing the updated information.

The overlying applications are required to fill the event information in the DENM, as well to instruct the Networking and Transport layer about the destination area that should be covered by the message. In our case, since for the first implementation of MS-VAN3T there is no model for BTP and Geonetworking, the information about destination area and traffic class are not transmitted.

The class of DEN basic service proposed in MS-VAN3T exposes 3 public methods that are used by the applications to trigger the generation of DENMs. The C++ code developed to model the three methods is reported below, starting from the `appDENM_trigger()` function.

```
1  DENBasicService_error_t
2  DENBasicService::appDENM_trigger(denData data, ActionID_t &actionid)
3  {
4     DENBasicService_error_t fillDENM_rval=DENM_NO_ERROR;
5     DENM_t *denm;
6
7     if(!CheckMainAttributes ())
8     {
9        return DENM_ATTRIBUTES_UNSET;
10    }
11
```

```
12    if(m_socket_tx==NULL)
13    {
14       return DENM_TX_SOCKET_NOT_SET;
15    }
16
17    if(!data.isDenDataRight())
18    return DENM_WRONG_DE_DATA;
19
20    denm=(DENM_t*) calloc(1, sizeof(DENM_t));
21    if(denm==NULL)
22    {
23       return DENM_ALLOC_ERROR;
24    }
25
26    /* 1. If validity is expired return DENM_T_O_VALIDITY_EXPIRED */
27    if (compute_timestampIts (m_real_time) > data.↩
          getDenmMgmtDetectionTime () + (data.↩
          getDenmMgmtValidityDuration ()*MILLI))
28    return DENM_T_O_VALIDITY_EXPIRED;
29
30    /* 2. Assign unused actionID value */
31    actionid.originatingStationID = m_station_id;
32    actionid.sequenceNumber = m_seq_number;
33
34    std::pair <unsigned long, long> map_index = std::make_pair((↩
          unsigned long)m_station_id,(long)m_seq_number);
35
36    m_seq_number++;
37
38    /* 3. 4. 5. Manage Transmission interval and fill DENM */
39    fillDENM_rval=fillDENM(denm,data,actionid,compute_timestampIts (↩
          m_real_time));
40
41    if(fillDENM_rval!=DENM_NO_ERROR)
42    {
43       freeDENM(denm);
44       return fillDENM_rval;
45    }
46
47    /* 6. 7. Construct DENM and pass it to the lower layers (now UDP,↩
           in the future BTP and GeoNetworking, then UDP) */
48    /** Encoding **/
49    char errbuff[ERRORBUFF_LEN];
50    size_t errlen=sizeof(errbuff);
51
52    if(asn_check_constraints(&asn_DEF_DENM,(DENM_t *)denm,errbuff,&↩
          errlen) == -1) {
53       LOG_ERROR("Unable to validate the ASN.1 contraints for the ↩
             current DENM."<<std::endl);
54       LOG_ERROR("Details: " << errbuff << std::endl);
```

```
55      return DENM_ASN1_UPER_ENC_ERROR;
56    }
57
58    asn_encode_to_new_buffer_result_t encode_result = ↩
          asn_encode_to_new_buffer(NULL,ATS_UNALIGNED_BASIC_PER,&↩
          asn_DEF_DENM, denm);
59    if (encode_result.result.encoded==-1)
60    {
61      return DENM_ASN1_UPER_ENC_ERROR;
62    }
63
64    Ptr<Packet> packet = Create<Packet> ((uint8_t*) encode_result.↩
          buffer, encode_result.result.encoded+1);
65    free(encode_result.buffer);
66
67    m_socket_tx->Send (packet);
68
69    /* 8. 9. Create an entry in the originating ITS-S message table ↩
          and set the state to ACTIVE and start the T_O_Validity timer. ↩
          */
70    /* As all the timers are stored, in this case, for each entry in ↩
          the table, we have to set them before saving the new entry to ↩
          a map, which is done as last operation. */
71    /* We are basically adding the current entry, containing the ↩
          already UPER encoded DENM packet, to a map of <ActionID,↩
          ITSSOriginatingTableEntry> */
72    ITSSOriginatingTableEntry entry(*packet, ↩
          ITSSOriginatingTableEntry::STATE_ACTIVE,actionid);
73
74    m_originatingTimerTable.emplace(map_index,std::tuple<Timer,Timer,↩
          Timer>());
75
76    DENBasicService::setDENTimer(std::get<V_O_VALIDITY_INDEX>(↩
          m_originatingTimerTable[map_index]),Seconds(data.↩
          getDenmMgmtValidityDuration ()),&DENBasicService::↩
          T_O_ValidityStop,actionid);
77
78    /* 10. Calculate and start timers T_RepetitionDuration and ↩
          T_Repetition when both parameters in denData are > 0 */
79    if(data.getDenmRepetitionDuration ()>0 && data.↩
          getDenmRepetitionInterval ()>0)
80    {
81      DENBasicService::setDENTimer(std::get<T_REPETITION_INDEX>(↩
            m_originatingTimerTable[map_index]),MilliSeconds(data.↩
            getDenmRepetitionInterval()),&DENBasicService::↩
            T_RepetitionStop,actionid);
82      DENBasicService::setDENTimer(std::get<↩
            T_REPETITION_DURATION_INDEX>(m_originatingTimerTable[↩
            map_index]),MilliSeconds(data.getDenmRepetitionDuration ())↩
            ,&DENBasicService::T_RepetitionDurationStop,actionid);
```

```
83    }
84
85    /* 11. Finally create the entry after starting all the timers (it↩
          refers to the step '8' of appDENM_trigger in ETSI EN302 637-3↩
          V1.3.1 */
86
87    m_originatingITSSTable[map_index]=entry;
88
89    /* 12. Send actionID to the requesting ITS-S application. This is↩
          requested by the standard, but we are already reporting the ↩
          actionID using &actionID */
90
91    freeDENM(denm);
92
93    return DENM_NO_ERROR;
94 }
```

The appDENM_trigger() function is called by the applications with the data specified in Table A.1 and with a reference to the *actionID*. As soon as the packet is built and transmitted (Line 67), an entry in the OriginatingITSSTable is created. This table, as defined in the standard, contains the information of all the active DENMs that have been transmitted by the entity. To manage the DENMs *validity duration*, in Line 76 a timer is set. As soon as the validity expires, the timer will trigger a callback that removes the entry in the OriginatingITSSTable. From Line 79 to Line 83, some timers are also set to manage the DENM repetition. When the timer associated with the *repetition time* expires, a new DENM is triggered; when the timer associated with the *repetition duration* expires, the DENM retransmission is stopped.

```
1  DENBasicService_error_t
2  DENBasicService::appDENM_update(denData data, const ActionID_t ↩
       actionid)
3  {
4     DENBasicService_error_t fillDENM_rval=DENM_NO_ERROR;
5     std::pair <unsigned long, long> map_index = std::make_pair((↩
          unsigned long)actionid.originatingStationID,(long)actionid.↩
          sequenceNumber);
6
7     DENM_t *denm;
8
9     if(!CheckMainAttributes ())
10    {
11       return DENM_ATTRIBUTES_UNSET;
12    }
13
14    if(m_socket_tx==NULL)
15    {
16       return DENM_TX_SOCKET_NOT_SET;
17    }
```

```
18
19    denm=(DENM_t*) calloc(1, sizeof(DENM_t));
20    if(denm==NULL)
21    {
22       return DENM_ALLOC_ERROR;
23    }
24
25    /* 1. If validity is expired return DENM_T_O_VALIDITY_EXPIRED */
26    if (compute_timestampIts (m_real_time) > data.↩
         getDenmMgmtDetectionTime () + (data.↩
         getDenmMgmtValidityDuration ()*MILLI))
27    return DENM_T_O_VALIDITY_EXPIRED;
28
29    /* 2. Compare actionID in the application request with entries in↩
          the originating ITS-S message table (i.e. ↩
         m_originatingITSSTable, implemented as a map) */
30    /* Gather also the proper entry in the table, if available. */
31
32    T_Repetition_Mutex.lock();
33    std::map<std::pair<unsigned long,long>, ITSSOriginatingTableEntry↩
         >::iterator entry_map_it = m_originatingITSSTable.find(↩
         map_index);
34    if (entry_map_it == m_originatingITSSTable.end())
35    {
36       T_Repetition_Mutex.unlock();
37       return DENM_UNKNOWN_ACTIONID;
38    }
39
40    /* 3. Stop T_O_Validity, T_RepetitionDuration and T_Repetition (↩
         if they were started - this check is already performed by the ↩
         setTimer* methods) */
41    std::get<V_O_VALIDITY_INDEX>(m_originatingTimerTable[map_index]).↩
         Cancel();
42    std::get<T_REPETITION_INDEX>(m_originatingTimerTable[map_index]).↩
         Cancel();
43    std::get<T_REPETITION_DURATION_INDEX>(m_originatingTimerTable[↩
         map_index]).Cancel();
44
45    /* 4. 5. 6. Manage transmission interval, reference time and fill↩
          DENM */
46    fillDENM_rval=fillDENM(denm,data,actionid,compute_timestampIts (↩
         m_real_time));
47
48    if(fillDENM_rval!=DENM_NO_ERROR)
49    {
50       T_Repetition_Mutex.unlock();
51       freeDENM(denm);
52       return fillDENM_rval;
53    }
54
```

```
55    /* 7. 8. Construct DENM and pass it to the lower layers (now UDP,↩
          in the future BTP and GeoNetworking, then UDP) */
56    /** Encoding **/
57    char errbuff[ERRORBUFF_LEN];
58    size_t errlen=sizeof(errbuff);
59
60    if(asn_check_constraints(&asn_DEF_DENM,denm,errbuff,&errlen) == ↩
          -1) {
61       LOG_ERROR("Unable to validate the ASN.1 contraints for the ↩
             received DENM."<<std::endl);
62       LOG_ERROR("Details: " << errbuff << std::endl);
63       return DENM_ASN1_UPER_ENC_ERROR;
64    }
65
66    asn_encode_to_new_buffer_result_t encode_result = ↩
          asn_encode_to_new_buffer(NULL,ATS_UNALIGNED_BASIC_PER,&↩
          asn_DEF_DENM, denm);
67    if (encode_result.result.encoded==-1)
68    {
69       T_Repetition_Mutex.unlock();
70       return DENM_ASN1_UPER_ENC_ERROR;
71    }
72
73    Ptr<Packet> packet = Create<Packet> ((uint8_t*) encode_result.↩
          buffer, encode_result.result.encoded+1);
74    free(encode_result.buffer);
75
76    m_socket_tx->Send (packet);
77
78    /* 9. Update the entry in the originating ITS-S message table. */
79    entry_map_it->second.setDENMPacket(*packet);
80
81    /* 10. Start timer T_O_Validity. */
82    DENBasicService::setDENTimer(std::get<V_O_VALIDITY_INDEX>(↩
          m_originatingTimerTable[map_index]),Seconds(data.↩
          getDenmMgmtValidityDuration ()),&DENBasicService::↩
          T_O_ValidityStop,actionid);
83
84    /* 11. Calculate and start timers T_RepetitionDuration and ↩
          T_Repetition when both parameters in denData are > 0 */
85    if(data.getDenmRepetitionDuration ()>0 && data.↩
          getDenmRepetitionInterval ()>0)
86    {
87       DENBasicService::setDENTimer(std::get<T_REPETITION_INDEX>(↩
             m_originatingTimerTable[map_index]),MilliSeconds(data.↩
             getDenmRepetitionInterval()),&DENBasicService::↩
             T_RepetitionStop,actionid);
```

```
88        DENBasicService::setDENTimer(std::get<↩
              T_REPETITION_DURATION_INDEX>(m_originatingTimerTable[↩
              map_index]),MilliSeconds(data.getDenmRepetitionDuration ())↩
              ,&DENBasicService::T_RepetitionDurationStop,actionid);
89    }
90
91    T_Repetition_Mutex.unlock();
92
93    freeDENM(denm);
94
95    return DENM_NO_ERROR;
96 }
```

The code developed in MS-VAN3T for the appDENM_update() function is similar to the one developed to create a new DENM. The main difference is that the application should pass to the function, as a parameter, the *actionID* of the DENM that should be updated. Also in this case, timers are used to manage the *validity duration*, *repetition interval* and *repetition duration*.

```
1  DENBasicService_error_t
2  DENBasicService::appDENM_termination(denData data, const ActionID_t ↩
       actionid)
3  {
4     DENM_t *denm;
5     uint8_t termination=0;
6     Termination_t asn_termination;
7     long referenceTime;
8
9     DENBasicService_error_t fillDENM_rval=DENM_NO_ERROR;
10
11    if(!CheckMainAttributes ())
12    {
13       return DENM_ATTRIBUTES_UNSET;
14    }
15
16    if(m_socket_tx==NULL)
17    {
18       return DENM_TX_SOCKET_NOT_SET;
19    }
20
21    denm=(DENM_t*) calloc(1, sizeof(DENM_t));
22    if(denm==NULL)
23    {
24       return DENM_ALLOC_ERROR;
25    }
26
27    std::pair <unsigned long, long> map_index = std::make_pair((↩
          unsigned long)actionid.originatingStationID,(long)actionid.↩
          sequenceNumber);
```

```
28
29    /* 1. If validity is expired return DENM_T_O_VALIDITY_EXPIRED */
30    if (compute_timestampIts (m_real_time) > data.↩
          getDenmMgmtDetectionTime () + (data.↩
          getDenmMgmtValidityDuration ()*MILLI))
31    return DENM_T_O_VALIDITY_EXPIRED;
32
33    /* 2. Compare actionID in the application request with entries in↩
           the originating ITS-S message table and the receiving ITS-S ↩
          message table */
34    T_Repetition_Mutex.lock();
35
36    std::map<std::pair<unsigned long,long>, ITSSOriginatingTableEntry↩
          >::iterator entry_originating_table = m_originatingITSSTable.↩
          find(map_index);
37    /* 2a. If actionID exists in the originating ITS-S message table ↩
          and the entry state is ACTIVE, then set termination to ↩
          isCancellation.*/
38    if (entry_originating_table != m_originatingITSSTable.end())
39    {
40       T_Repetition_Mutex.unlock();
41       return DENM_UNKNOWN_ACTIONID_ORIGINATING;
42    }
43    else if(entry_originating_table->second.getStatus()==↩
          ITSSOriginatingTableEntry::STATE_ACTIVE)
44    {
45       asn_termination=Termination_isCancellation;
46       if(asn_maybe_assign_optional_data<Termination_t>(&↩
             asn_termination,&denm->denm.management.termination,↩
             m_ptr_queue)==-1)
47       {
48          T_Repetition_Mutex.unlock();
49          return DENM_ALLOC_ERROR;
50       }
51       else
52       {
53          termination=0;
54       }
55    }
56    else
57    {
58       T_Repetition_Mutex.unlock();
59       return DENM_NON_ACTIVE_ACTIONID_ORIGINATING;
60    }
61
62    /* 2b. If actionID exists in the receiving ITS-S message table ↩
          and the entry state is ACTIVE, then set termination to ↩
          isNegation.*/
```

```
63    std::map<std::pair<unsigned long,long>, ITSSReceivingTableEntry>::iterator entry_receiving_table = m_receivingITSSTable.find(map_index);
64    if (entry_receiving_table != m_receivingITSSTable.end())
65    {
66       T_Repetition_Mutex.unlock();
67       return DENM_UNKNOWN_ACTIONID_RECEIVING;
68    }
69    else if(entry_receiving_table->second.getStatus()==ITSSReceivingTableEntry::STATE_ACTIVE)
70    {
71       asn_termination=Termination_isNegation;
72       if(asn_maybe_assign_optional_data<Termination_t>(&asn_termination,&denm->denm.management.termination,m_ptr_queue)==-1)
73       {
74          T_Repetition_Mutex.unlock();
75          return DENM_ALLOC_ERROR;
76       }
77       else
78       {
79          termination=1;
80       }
81    }
82    else
83    {
84       T_Repetition_Mutex.unlock();
85       return DENM_NON_ACTIVE_ACTIONID_RECEIVING;
86    }
87
88    if(termination==1)
89    {
90       referenceTime=entry_receiving_table->second.getReferenceTime();
91
92       if(referenceTime==-1)
93       {
94          T_Repetition_Mutex.unlock();
95          return DENM_WRONG_TABLE_DATA;
96       }
97    }
98    else
99    {
100      referenceTime=compute_timestampIts (m_real_time);
101   }
102
103   /* 3. Set referenceTime to current time (for termination = 0) or to the receiving table entry's reference time (for termination = 1) and fill DENM */
104   fillDENM_rval=fillDENM(denm,data,actionid,referenceTime);
```

```
105    if(fillDENM_rval!=DENM_NO_ERROR)
106    {
107       T_Repetition_Mutex.unlock();
108       freeDENM(denm);
109       return fillDENM_rval;
110    }
111
112    /* 4a. Gather the proper timers from the timers table */
113    std::map<std::pair<unsigned long,long>, std::tuple<Timer,Timer,↩
          Timer>>::iterator entry_timers_table = m_originatingTimerTable↩
          .find(map_index);
114
115    /* 4b. Stop T_O_Validity, T_RepetitionDuration and T_Repetition (↩
          if they were started - this check is already performed by the ↩
          setTimer* methods) */
116    std::get<V_O_VALIDITY_INDEX>(entry_timers_table->second).Cancel()↩
          ;
117    std::get<T_REPETITION_INDEX>(entry_timers_table->second).Cancel()↩
          ;
118    std::get<T_REPETITION_DURATION_INDEX>(entry_timers_table->second)↩
          .Cancel();
119
120    /* 5. Construct DENM and pass it to the lower layers (now UDP, in↩
           the future BTP and GeoNetworking, then UDP) */
121    /** Encoding **/
122    char errbuff[ERRORBUFF_LEN];
123    size_t errlen=sizeof(errbuff);
124
125    if(asn_check_constraints(&asn_DEF_DENM,denm,errbuff,&errlen) == ↩
          -1) {
126       NS_LOG_ERROR("Unable to validate the ASN.1 contraints for the ↩
             received DENM."<<std::endl);
127       NS_LOG_ERROR("Details: " << errbuff << std::endl);
128       return DENM_ASN1_UPER_ENC_ERROR;
129    }
130
131    asn_encode_to_new_buffer_result_t encode_result = ↩
          asn_encode_to_new_buffer(NULL,ATS_UNALIGNED_BASIC_PER,&↩
          asn_DEF_DENM, &denm);
132    if (encode_result.result.encoded==-1)
133    {
134       T_Repetition_Mutex.unlock();
135       return DENM_ASN1_UPER_ENC_ERROR;
136    }
137
138    Ptr<Packet> packet = Create<Packet> ((uint8_t*) encode_result.↩
          buffer, encode_result.result.encoded+1);
139    free(encode_result.buffer);
140
141    m_socket_tx->Send (packet);
```

```
142
143     /* 6a. If termination is set to 1, create an entry in the ↩
            originating ITS-S message table and set the state to NEGATED. ↩
            */
144     if(termination==1)
145     {
146         ITSSOriginatingTableEntry entry(*packet, ↩
                ITSSOriginatingTableEntry::STATE_NEGATED,actionid);
147         m_originatingITSSTable[map_index]=entry;
148     }
149     /* 6b. If termination is set to 0, update the entry in the ↩
            originating ITS-S message table and set the state to CANCELLED↩
            . */
150     else
151     {
152         entry_originating_table->second.setDENMPacket(*packet);
153         entry_originating_table->second.setStatus(↩
                ITSSOriginatingTableEntry::STATE_CANCELLED);
154     }
155
156     /* 7. Start timer T_O_Validity. */
157     DENBasicService::setDENTimer(std::get<V_O_VALIDITY_INDEX>(↩
            entry_timers_table->second),Seconds(data.↩
            getDenmMgmtValidityDuration ()),&DENBasicService::↩
            T_O_ValidityStop,actionid);
158
159     /* 8. Calculate and start timers T_RepetitionDuration and ↩
            T_Repetition when both parameters in denData are > 0 */
160     if(data.getDenmRepetitionDuration ()>0 && data.↩
            getDenmRepetitionInterval ()>0)
161     {
162         DENBasicService::setDENTimer(std::get<T_REPETITION_INDEX>(↩
                entry_timers_table->second),MilliSeconds(data.↩
                getDenmRepetitionInterval()),&DENBasicService::↩
                T_RepetitionStop,actionid);
163         DENBasicService::setDENTimer(std::get<↩
                T_REPETITION_DURATION_INDEX>(entry_timers_table->second),↩
                MilliSeconds(data.getDenmRepetitionDuration ()),&↩
                DENBasicService::T_RepetitionDurationStop,actionid);
164     }
165
166     T_Repetition_Mutex.unlock();
167
168     freeDENM(denm);
169
170     return DENM_NO_ERROR;
171 }
```

The function appDENM_termination() is called whenever an application requires the DEN basic service to stop the generation of DENMs for a specific event. It can also be

used to generate *Negation DENM*: in Line 63, the `ReceivingITSSTable` (containing the DENM received from other entities) is checked, and in case the DENM is found to be present in the table, the generated DENM is tagged as *Negation DENM* (Line 71). Finally, from Line 116 to Line 118 all the timers previously associated to the DENM are stopped, the packet is created and sent through the UDP socket. Also the *Negation/Cancellation DENM* is associated with a *validity duration* and can be sent with a frequency set by the application, as implemented from Line 156 to Line 164.

A.2.2 DENM decoding and Reception Management

Upon the reception of a DENM, the DEN basic service is required to decode the information and to make the content of the DENM available to the requesting ITS applications. In this case, the standard specifies that the DENM information should entirely be transmitted at application layer. In MS-VAN3T, a callback mechanism is used to pass the information to the upper layers. Differently from the CA basic service, here a table containing the information about the each received DENM is created and maintained. The table, named `ReceivingITSSTable`, stores each DENM and its state, and is used to perform some of the operation of the DENM protocol.

The C++ code developed to model the DENM reception and subsequent operations is reported below.

```
1   void
2 DENBasicService::receiveDENM(Ptr<Socket> socket)
3 {
4    Ptr<Packet> packet;
5    DENM_t *decoded_denm;
6    denData den_data;
7    Address from;
8    ValidityDuration_t validityDuration;
9    ActionID_t actionID;
10   long detectionTime_long;
11   long referenceTime_long;
12   std::pair <unsigned long, long> map_index;
13
14   packet = socket->RecvFrom (from);
15
16   uint8_t *buffer = new uint8_t[packet->GetSize ()];
17   packet->CopyData (buffer, packet->GetSize ()-1);
18
19   if(!CheckMainAttributes ())
20   {
21      LOG_ERROR("DENBasicService has unset parameters. Cannot ↩
          receive any data.");
22      return;
23   }
24
```

```
25    /* Try to check if the received packet is really a DENM */
26    if (buffer[1]!=FIX_DENMID)
27    {
28       LOG_ERROR("Warning: received a message which has messageID '"<<buffer[1]<<"' but '1' was expected.");
29       return;
30    }
31
32    /** Decoding **/
33    void *decoded_=NULL;
34    asn_dec_rval_t decode_result;
35
36    do {
37       decode_result = asn_decode(0, ATS_UNALIGNED_BASIC_PER, &asn_DEF_DENM, &decoded_, buffer, packet->GetSize ()-1);
38    } while(decode_result.code==RC_WMORE);
39
40    if(decode_result.code!=RC_OK || decoded_==NULL) {
41       LOG_ERROR("Warning: unable to decode a received DENM.");
42       return;
43    }
44
45    decoded_denm = (DENM_t *) decoded_;
46
47    /* Compute T_R_Validity expiration time */
48
49    validityDuration = decoded_denm->denm.management.validityDuration != NULL ? *(decoded_denm->denm.management.validityDuration) : DEN_DEFAULT_VALIDITY_S;
50    asn_INTEGER2long (&decoded_denm->denm.management.detectionTime,&detectionTime_long);
51    asn_INTEGER2long (&decoded_denm->denm.management.referenceTime,&referenceTime_long);
52
53    long now = compute_timestampIts (m_real_time);
54
55    /* 1. If validity is expired return without performing further steps */
56    if (now > detectionTime_long + ((long)validityDuration*MILLI))
57    {
58       LOG_ERROR("Warning: received a DENM with an expired validity. Detection time (ms): "<<detectionTime_long<<"; validity duration (s): "<<(long)validityDuration);
59       LOG_ERROR("Condition: '"<<now<<" > "<< detectionTime_long + ((long)validityDuration*MILLI)<<"' is true. Omitting further operations.");
60       return;
61    }
62
```

```
63    /* Lookup entries in the receiving ITS-S message table with the ↩
          received actionID */
64    actionID=decoded_denm->denm.management.actionID;
65    map_index = std::make_pair((unsigned long)actionID.↩
          originatingStationID,(long)actionID.sequenceNumber);
66
67    std::map<std::pair<unsigned long,long>, ITSSReceivingTableEntry↩
          >::iterator entry_rx_map_it = m_receivingITSSTable.find(↩
          map_index);
68
69    if (entry_rx_map_it == m_receivingITSSTable.end ())
70    {
71       /* a. If entry does not exist in the receiving ITS-S message ↩
              table, check if termination data exists in the
72       received DENM. */
73       if(decoded_denm->denm.management.termination!=NULL &&
74       (*(decoded_denm->denm.management.termination)==↩
              Termination_isCancellation ||
75       *(decoded_denm->denm.management.termination)==↩
              Termination_isNegation))
76       {
77          /* if yes, discard the received DENM and omit execution of ↩
                 further steps. */
78          LOG_ERROR("Warning: received a new DENM with termination ↩
                 data (either cancelled or negated). Omitting further ↩
                 reception steps.");
79          return;
80       }
81       else
82       {
83          /* if not, create an entry in the receiving ITS-S message ↩
                 table with the received DENM and set the state to ACTIVE↩
                  (SSP is not yet implemented) */
84          ITSSReceivingTableEntry entry(*packet,↩
                 ITSSReceivingTableEntry::STATE_ACTIVE,actionID,↩
                 referenceTime_long,detectionTime_long);
85          m_receivingITSSTable[map_index]=entry;
86       }
87    }
88    else
89    {
90       /* b. If entry does exist in the receiving ITS-S message table↩
              , check if the received referenceTime is less than the ↩
              entry referenceTime,
91       * or the received detectionTime is less than the entry ↩
              detectionTime */
92       long stored_reference_time = entry_rx_map_it->second.↩
              getReferenceTime ();
93       long stored_detection_time =  entry_rx_map_it->second.↩
              getDetectionTime ();
```

```
94
95      if (referenceTime_long < stored_reference_time || ↩
            detectionTime_long < stored_detection_time)
96      {
97          /* i. if yes, discard received DENM and omit execution of ↩
               further steps. */
98          LOG_ERROR("Warning: received a new DENM with reference time↩
                < entry reference time or  detection time < stored ↩
               detection time.");
99          LOG_ERROR("reference time (ms): "<<referenceTime_long<<"; ↩
               stored value: "<<stored_reference_time
100         <<"; detection time (ms): "<<detectionTime_long<<"; store ↩
               value: "<<stored_detection_time);
101         return;
102     }
103     else
104     {
105         /* ii. Otherwise, check if the received DENM is a repeated ↩
               DENM of the entry, i.e. the received referenceTime ↩
               equals to the entry
106         * referenceTime, the received detectionTime equals to the ↩
               entry
107         * detectionTime, and the received termination value equals ↩
               to the entry state */
108         if(referenceTime_long == stored_reference_time &&
109         detectionTime_long == stored_detection_time &&
110         (
111         (decoded_denm->denm.management.termination==NULL &&  !↩
               entry_rx_map_it->second.isTerminationSet()) ||
112         (decoded_denm->denm.management.termination!=NULL && *(↩
               decoded_denm->denm.management.termination)==↩
               entry_rx_map_it->second.getTermination ())
113         ))
114         {
115             /* 1. If yes, discard received DENM and omit execution ↩
                   of further steps. */
116             LOG_ERROR("Warning: received a repeated DENM. It won't ↩
                   produce any new effect on the involved vehicle.");
117             return;
118         }
119         else
120         {
121             /* 2. Otherwise, update the entry in receiving ITS-S ↩
                   message table, set entry state according
122             * to the termination value of the received DENM. (SSP is↩
                    not yet implemented) */
123             ITSSReceivingTableEntry entry(*packet,↩
                   ITSSReceivingTableEntry::STATE_ACTIVE,actionID,↩
                   referenceTime_long,detectionTime_long,decoded_denm->↩
                   denm.management.termination);
```

```
124             entry_rx_map_it->second=entry;
125           }
126       }
127
128   }
129
130   /* Start/restart T_R_Validity timer. */
131   if(m_T_R_Validity_Table.find(map_index) == m_T_R_Validity_Table.↩
        end()) {
132       m_T_R_Validity_Table[map_index] = Timer();
133   }
134
135   DENBasicService::setDENTimer(m_T_R_Validity_Table[map_index],↩
        Seconds((long)validityDuration),&DENBasicService::↩
        T_R_ValidityStop,actionID);
136
137   /* Fill den_data with the received information */
138   DENBasicService::fillDenDataHeader (decoded_denm->header, ↩
        den_data);
139   DENBasicService::fillDenDataManagement (decoded_denm->denm.↩
        management, den_data);
140
141   if(decoded_denm->denm.location!=NULL)
142   DENBasicService::fillDenDataLocation (*decoded_denm->denm.↩
        location, den_data);
143
144   if(decoded_denm->denm.situation!=NULL)
145   DENBasicService::fillDenDataSituation (*decoded_denm->denm.↩
        situation, den_data);
146
147   if(decoded_denm->denm.alacarte!=NULL)
148   DENBasicService::fillDenDataAlacarte (*decoded_denm->denm.↩
        alacarte, den_data);
149
150   m_DENReceiveCallback(den_data,from);
151 }
```

As soon as the packet is received and the DENM is decoded, its validity is checked and the operations to populate and maintain the `ReceivingITSSTable` start. In case of a *Negation/Cancellation DENM* the corresponding entry is deleted from the table, while in the other cases it is created or updated. From Line 130 to Line 135, a timer associated to the validity of received DENM is set: as soon as it expires, the corresponding DENM will be erased from the `ReceivingITSSTable`. Finally, from Line 137 to Line 150, the data to be send to the applications is parsed and transmitted using a callback (similarly to the CA basic service).

Appendix B

The Collision Avoidance Algorithm

B.1 Introduction

The core of the Collision Avoidance Service (CAS) presented in Chapter 4 is the Collision Avoidance Algorithm (CAA). The main target of CAA is to determine whether the entities that send CAMs to the system are set on a collision course. The calculation takes into account the position, heading, speed, acceleration and dimension extracted from the CAMs, and the two parameters inferred are the Time-to-Collision (T2C) and the Space-to-Collision (S2C).

In a nutshell, the algorithm projects the position of the entities under test in the future, based on the current status and dynamics. Then, it computes the mutual distance and, through derivatives, it infers the time at which the aforementioned distance will be at its minimum. At this point, using the derived time, it is possible to understand what will be the minimum distance between the two entities under test.

Note that, although the algorithm has been originally thought to facilitate the vehicles to avert potential collisions, being it based on trajectory computation, it can be used to support any kind of road player, including vulnerable users such as cyclists, scooter drivers and even pedestrians.

B.2 Collision Avoidance Algorithm pseudocode

The pseudocode of CAA is shown in Algorithm 2. It is run each time a new CAM message is received; for simplicity, the entity that generated the CAM will be referred to as e_0. CAA requires as input: *(i)* the position $\vec{p}$, speed $\vec{v}$, acceleration $\vec{a}$ and size s of e_0; *(ii)* the set B containing the latest information of the other entities connected to the system. In Line 1 the set C (that at the end will contain the list of entities which

Algorithm 2 Collision Avoidance Algorithm pseudocode

Require: $\vec{p}, \vec{v}, \vec{a}, s, B$

1: $C \leftarrow \emptyset$
2: $p_x(t) \leftarrow p_x + v_x t + \frac{1}{2} a_x t^2$
3: $p_y(t) \leftarrow p_y + v_y t + \frac{1}{2} a_y t^2$
4: **for all** $b \in B$ **do**
5: **read** $\hat{\vec{p}}, \hat{\vec{v}}, \hat{\vec{a}}, \hat{s}$ **from** b
6: $\hat{p}_x(t) \leftarrow \hat{p}_x + \hat{v}_x t + \frac{1}{2} \hat{a}_x t^2$
7: $\hat{p}_y(t) \leftarrow \hat{p}_y + \hat{v}_y t + \frac{1}{2} \hat{a}_y t^2$
8: $D(t) \leftarrow (p_x - \hat{p}_x)^2 + (p_y - \hat{p}_y)^2 =$
$= \left[p_x^0 - \hat{p}_x^0 + (v_x - \hat{v}_x) t + \frac{1}{2} (a_x - \hat{a}_x) t^2\right]^2 + \left[p_y^0 - \hat{p}_y^0 + (v_y - \hat{v}_y) t + \frac{1}{2} (a_y - \hat{a}_y) t^2\right]^2$
9: $T \leftarrow t : \frac{\mathrm{d}}{\mathrm{d}t} D(t) = 0$
10: **for all** $t^\star \in T$ **do**
11: **if** $t^* < 0$ **or** $t^* > t2c_t$ **then**
12: **continue**
13: **end if**
14: $d^\star \leftarrow \sqrt{D(t^\star)}$
15: $s2c_t = \text{computeS2CThreshold}(s, \hat{s})$
16: **if** $d^\star \leq s2c_t$ **then**
17: $C \leftarrow C \cup \{b\}$
18: **break**
19: **end if**
20: **end for**
21: **end for**
22: **return** C

may potentially collide with e_0) is initialized, and in Line 2 and 3 the position of e_0 is projected in the future using the uniformly accelerated motion formula.

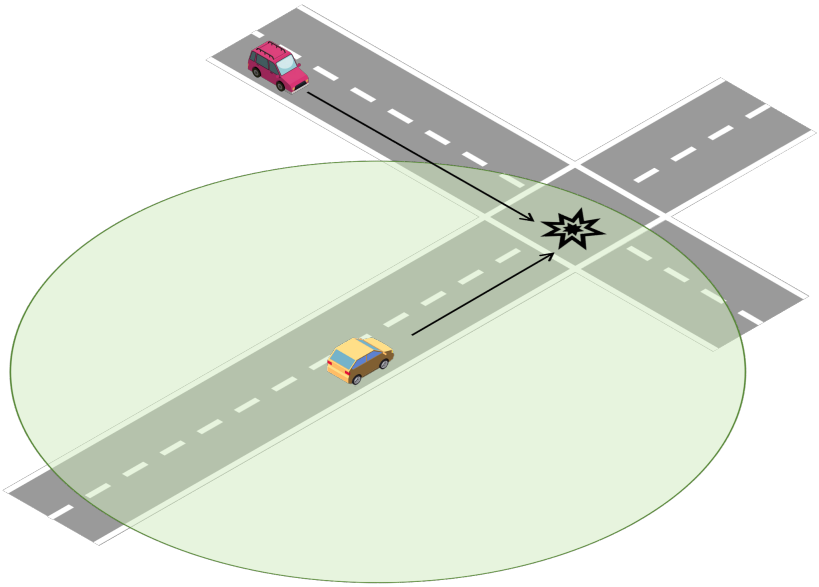
At this point, the algorithm iterates over all the entities b present in B (Line 4), and, similarly to the previous step, projects their position according to the information stored in B (Line 6 and 7). Then, in Line 8, the squared Euclidean distance between e_0 and b is computed.

Therefore, $D(t)$ represents the squared distance between e_0 and b as a function of time. Since the system is interested in knowing what will be the **minimum** of such a distance, in Line 9 the derivative of $D(t)$ in the time domain is computed and all the possible solutions are stored in T. Among the computed solutions $t^\star$, CAA discards those which correspond to negative times and to times that will be too far in the future (Line 11). Finally, CAA computes the minimum distance (S2C) in Line 14, and adds to C only those entities which will have a minimum mutual distance with e_0 lower than a certain distance threshold. The computation of this distance threshold depends on the size of the involved entities and will be explained in the next sections.

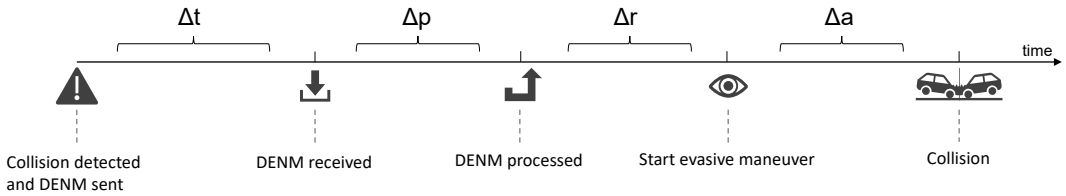
After processing all the entities present in B, the algorithm returns the set C containing the entities that will likely collide with e_0, together with the computed T2C and S2C values.

B.3 T2C threshold

The value of the T2C threshold determines the time at which the first DENM message related to a certain collision is sent to the involved entities. This value should be nor too big, to avoid receiving DENM related to dangerous situation that will happen too far in the future, nor too small, to avoid that the involved entities do not have time to properly react to the warning.



(a) The green region represents the virtual safety shield around the yellow vehicle, according to ETSI [34].



(b) Timeline comprising each phase between the DENM generation and the actual collision.

Figure B.1: The selection of the T2C threshold should take into account several factors, from the involved entities dynamics to the road surface conditions.

The ETSI standard in [34] provides some general guidelines on how to set this value. Each vehicle is in charge of defining a self virtual safety shield, as represented in Figure B.1a, and as detailed in Section 4.4.1. The dimension of this shield is dynamic and depends on a number of factors: the speed, the braking properties, the vehicle's mass, the tire status and the road surface conditions are among the most important. Whenever a vehicle receives a DENM, if it detects that it may be related to a dangerous situation within its safety shield, then the in-vehicle collision avoidance mechanisms (either HMI-based or autonomous evasive maneuver) should be triggered. Since CAS is not aware of the size of such a shield, it will be necessary to select a T2C that generates DENMs for events happening close to the outside border of the shield, so that the vehicles receiving it have the time to implement an appropriate reaction.

To better characterize the various phases involved in the setting of the T2C threshold, in Figure B.1b, the main events happening between the DENM generation and the actual collision are reported. Δt represents the time gap from when a DENM is sent, to when it is received at the radio interface of the involved entity. Therefore, Δt comprehends the communication delays (time to access the channel, transmission, propagation and reception delays), and it is strictly dependent on the access technology adopted. After the DENM is received, it takes Δp for the receiver to process the information. This value mostly depends on the receiving system performances, and comprises the time to decode the information, to present it at application layer (where the decision is taken), and to show the warning on the HMI system (or, in case of advanced autonomous driving system, to trigger the in-vehicle actuators). In case the HMI warning is triggered, another important contribution is played by the time Δr, denoting the time required for human driver to be aware of, to interpret the HMI information and to act on the vehicle system [34]. Finally, the last contribution is given by Δa, representing the time required for the entity to take actions to avoid the collision. Δa depends on the deceleration capabilities, on the mass of the entity, on its current speed and on other factors (also external, like the road surface conditions).

Therefore, the CAA should set the T2C threshold to be higher enough to comprehend all the aforementioned time contributions, plus an additional safety margin (ϵ, in Equation B.1) taking into account additional delays and possible positioning system errors.

$$T2C_{threshold} > \Delta t + \Delta p + \Delta r + \Delta a + \epsilon \tag{B.1}$$

B.4 S2C threshold

The value of the S2C threshold determines the minimum mutual distance that two projected trajectories should reach before a collision risk is reported by CAA. Differently from the T2C threshold, whose computation involves a number of external factors (also independent on the involved entities), in this case the CAA has all the information

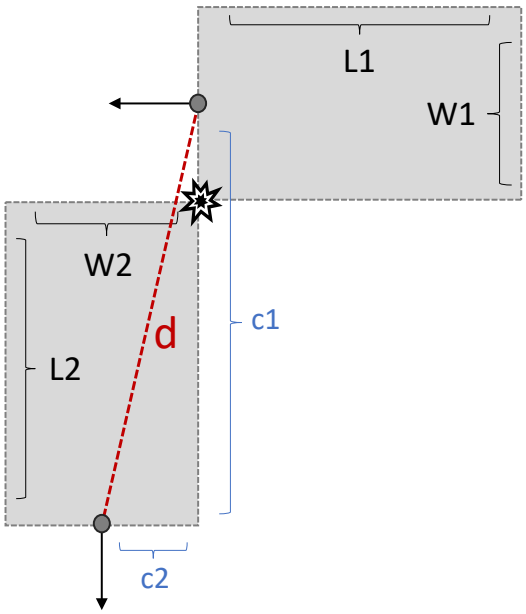


Figure B.2: Worst case, in terms of mutual distance (d), in which two entities collide at an intersection.

needed to compute such a minimum distance. In fact, each CAM contains the corresponding entity's length and width, that can be used to asses the worst case, in terms of mutual distance, in which two entities will collide. In Figure B.2, such a distance is represented by the red line, and referred to as d.

In this case, the distance between the entities is measured from the center of the front side. In case of vehicles, it will be the center of the front bumper; this choice was taken to reflect the way in which SUMO (the mobility simulator on top of which CAS was first built) reports the simulated entities positions. In a real scenario this parameter will be different: the GPS antenna is normally placed at the top of the vehicle center, or in the pocket of the vulnerable user (in case they use a smarphone), so that the distance calculation will be slightly different.

Therefore, the value of d is computed by applying the Pythagorean theorem on $c1$ and $c2$. The Equation B.2 shows how $c1$ and $c2$ are computed; *otherWidth(max(L1,L2))* is a simple function that returns W2 if the maximum between L1 and L2 is L1, and that returns W1 vice-versa. At the same time, *thisWidth(max(L1,L2))* returns W1 in case that the maximum between L1 and L2 is L1, and it returns W2 vice-versa.

$$\begin{aligned} c1 &= max(L1, L2) + otherWidth(max(L1, L2))/2 \\ c2 &= thisWidth(max(L1, L2))/2 \end{aligned} \tag{B.2}$$

Equation B.3 shows the computation of d, to which a safety margin γ is added, to

take into account possible positioning errors and to avoid that two entities do not pass very close to each other without the system detecting the actual danger.

$$d = S2C_{threshold} = \sqrt{c1^2 + c2^2} + \gamma \tag{B.3}$$

The value of d is the value that CAA associates to the S2C threshold, and the routines described in this section are those corresponding to the function *computeS2CThreshold* in Line 15 of Algorithm 2.

List of acronyms

3GPP	Third Generation Partnership Project
AC	Access Category
AIFS	Arbitration Inter-Frame Spacing
ASN.1	Abstract Syntaxt Notation revision One
BSM	Basic Safety Message
BSS	Basic Service Set
BTP	Basic Transport Protocol
CA	Certificate Authority
CAA	Collision Avoidance Algorithm
CA-BS	Cooperative Awareness Basic Service
CAM	Cooperative Awareness Message
CAS	Collision Avoidance System
CBR	Channel Busy Ratio
CCH	Control Channel
C-ITS	Cooperative Intelligent Transportation System
CRLVE	Certificate Revocation List Verification Entity
CSMA-CA	Carrier Sensing Multiple Access - Collision Avoidance
D2D	Device-to-Device
DCC	Decentralized Congestion Control
DE-BS	Decentralize Environmental Basic Service
DENM	Decentralized Environmental Notification Message
DMRS	Demodulation Reference Signal
DSRC	Dedicated Short-Range Communication
EARFCN	E-UTRA Absolute Radio Frequency Channel Number
EDCA	Enhanced Distributed Channel Access

EN	European Norm
eNB	E-UTRAN NodeB
EPC	Evolved Packet Core
ETSI	European Telecommunication Standards Institute
EV	Emergency Vehicle
EVA	Emergency Vehicle Alert
FCC	Federal Communications Commission
GN	GeoNetworking
gNB	Next Generation NodeB
gNB	Graphical User Interface
GPS	Global Positioning System
ICA	Intersection Collision Avoidance
ICRW	Intersection Collision Risk Warning
IEEE	Institute of Electrical and Electronics Engineers
ITS	Intelligent Transportation System
ITS-S	Intelligent Transportation System Station
IVI	Infrastructure to Vehicle Information
LaMP	Latency Measurement Protocol
LaTe	Latency Tester
LDM	Local Dynamic Map
LCRW	Longitudinal Collision Risk Warning
LTE	Long Term Evolution
MAC	Medium Access Control
MANETs	Mobile Ad-Hoc Networks
MBMS	Multimedia Broadcast Multicast Service
MCS	Modulation and Coding Scheme
MEC	Multi-Access Edge Computing
MLME	MAC Layer Management Entity
MME	Mobility Management Entity
MNO	Mobile Network Operator
NR	New Radio
ns-3	network simulator 3

NTP	Network Time Protocol
OCB	Outside the Context of an 802.11 BSS
OBU	On-board Units
OFDM	Orthogonal Frequency-Division Multiplexing
P2PCD	Peer-to-peer Certificate Distribution
PDR	Packet Delivery Ratio or Packet Drop Ratio (depends on the context)
PDU	Protocol Data Unit
PGW	Packet Data Network Gateway
PSID	Provider Service Identifier
PSSCH	Physical Sidelink Control Channel
PTP	Precion Time Protocol
QoS	Quality of Service
RHS	Road Hazard Signaling
RLT	Road and Lane Topology
RP	Resource Pool
RSSI	Received Signal Strength Indicator
RSU	Road Side Unit
RTT	Round Trip Time
S2C	Space-to-collision
SAE	Society of Automotive Engineers
SAM	Service Announcement Message
SC-FDMA	Single carrier frequency division multiple access
SCH	Service Channel
SCI	Sidelink Control Information
SDS	Secure Data Service
SGW	Serving Gateway
SPDU	Secured Protocol Data Unit
SSME	Station Security Management Entity
STA	802.11 Stations
T2C	Time-to-collision
TB	Transport Block
TLM	Traffic Light Maneuver

TS	Technical Specification
TDC	Transmission Datarate Control
TPC	Transmission Power Control
TRC	Transmission Rate Control
TTI	Transmission Time Interval
UE	User Equipment
URLLC	Ultra Reliable Low Latency Communication
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VANETs	Vehicular Ad-Hoc Networks
VDP	Vehicle Data Provider
VNMF	Virtualized Network Measurements Function
WAVE	Wireless Access for Vehicular Environment
WLAN	Wireless Local Area Network
WME	WAVE Management Entity
WNIC	Wireless Network Interface Controller
WSA	WAVE Service Advertisement
WSM	WAVE Short Message
WSMP	WAVE Short Message Protocol

Bibliography

[1] *3GPP TR 21.914 V14.0.0 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 14 Description; Summary of Rel-14 Work Items (Release 14)*. Technical Requirement. 3rd Generation Partnership Project, 2018.

[2] *3GPP TR 21.915 V15.0.0 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 15 Description; Summary of Rel-15 Work Items (Release 15)*. Technical Requirement. 3rd Generation Partnership Project, 2019.

[3] *3GPP TR 21.916 V0.6.0 - 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Release 16 Description; Summary of Rel-16 Work Items (Release 16)*. Technical Requirement. 3rd Generation Partnership Project, 2020.

[4] *5G-LENA simulator*. URL: https://5g-lena.cttc.es/.

[5] *5GAA - Coexistence of C-V2X and ITS-G5 at 5.9GHz*. White Paper. 5G Automotive Association, 2018.

[6] *8/671/EC- Commission Decision of 5 August 2008 on the Harmonised use of Radio Spectrum in the 5875-5905 MHz Frequency Band for Safety-Related Applications of Intelligent Transport Systems (ITS)*. Standard. European Commission, 2008.

[7] *A ns3 module for bidirectional coupling with SUMO*. URL: https://github.com/vodafone-chair/ns3-sumo-coupling.

[8] *A Two-Way Active Measurement Protocol (TWAMP)*. Tech. rep. RFC, 2008. DOI: 10.17487/RFC5357.

[9] A. Abunei, C. Comşa, and I. Bogdan. "Implementation of a Cost-effective V2X hardware and software platform." In: *2016 International Conference on Communications (COMM)*. June 2016, pp. 367–370. DOI: 10.1109/ICComm.2016.7528312.

[10] A. Abunei, C. Comşa, and I. Bogdan. "Implementation of ETSI ITS-G5 based inter-vehicle communication embedded system." In: *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*. 2017, pp. 1–4. DOI: 10.1109/ISSCS.2017.8034921.

[11] N. Agafonovs, G. Strazdins, and M. Greitans. "Accessible, customizable, high-performance IEEE 802.11p vehicular communication solution." In: *2012 The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*. June 2012, pp. 127–132. DOI: 10.1109/MedHocNet.2012.6257112.

[12] *An assessment of LTE-V2X (PC5) and 802.11p direct communications technologies for improved road safety in the EU*. White paper. 5GAA, 2017.

[13] G. Avino, M. Malinverno, C. Casetti, C. F. Chiasserini, F. Malandrino, M. Rapelli, and G. Zennaro. "Support of Safety Services through Vehicular Communications: The Intersection Collision Avoidance Use Case." In: *2018 International Conference of Electrical and Electronic Technologies for Automotive*. 2018, pp. 1–6.

[14] N. Baldo, M. Miozzo, M. Requena-Esteso, and J. Nin-Guerrero. "An Open Source Product-oriented LTE Network Simulator Based on ns-3." In: *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. 2011, pp. 293–298. URL: http://doi.acm.org/10.1145/2068897.2068948.

[15] A. Bazzi, G. Cecchini, M. Menarini, B. Masini, and A. Zanella. "Survey and perspectives of vehicular Wi-Fi versus sidelink cellular-V2X in the 5G era." In: *Future Internet* 11.6 (2019), p. 122.

[16] A. Böhm, M. Jonsson, and E. Uhlemann. "Performance comparison of a platooning application using the IEEE 802.11p MAC on the control channel and a centralized MAC on a service channel." In: *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. Oct. 2013, pp. 545–552. DOI: 10.1109/WiMOB.2013.6673411.

[17] G. Carneiro, P. Fortuna, and M. Ricardo. "FlowMonitor - a network monitoring framework for the Network Simulator 3 (NS-3)." In: (Jan. 2009). DOI: 10.4108/ICST.VALUETOOLS2009.7493.

[18] J. Choi, V. Marojevic, C. B. Dietrich, J. H. Reed, and S. Ahn. "Survey of Spectrum Regulation for Intelligent Transportation Systems." In: *IEEE Access* 8 (2020), pp. 140145–140160. DOI: 10.1109/ACCESS.2020.3012788.

[19] F. Eckermann, M. Kahlert, and C. Wietfeld. "Performance Analysis of C-V2X Mode 4 Communication Introducing an Open-Source C-V2X Simulator." In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. 2019, pp. 1–5. DOI: 10.1109/VTCFall.2019.8891534.

[20] ETSI. *ETSI Standard*. Oct. 2020. URL: https://www.etsi.org/standards#Transportation.

[21] *ETSI EN 302 636-1 V1.2.1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1: Requirements*. Standard. European Telecommunication Standard Institute, 2014.

[22] *ETSI EN 302 636-2 V1.2.1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 2: Scenarios*. Standard. European Telecommunication Standard Institute, 2013.

[23] *ETSI EN 302 636-3 V1.1.2 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 3: Network Architecture*. Standard. European Telecommunication Standard Institute, 2014.

[24] *ETSI EN 302 636-4-1 V1.4.1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality*. Standard. European Telecommunication Standard Institute, 2019.

[25] *ETSI EN 302 636-5-1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol*. Standard. European Telecommunication Standard Institute, 2017.

[26] *ETSI EN 302 637-2 V1.4.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. Standard. European Telecommunication Standard Institute, 2019.

[27] *ETSI EN 302 637-3 V1.3.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*. Standard. European Telecommunication Standard Institute, 2019.

[28] *ETSI EN 302 663 V1.2.0 - Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*. Standard. European Telecommunication Standard Institute, 2012.

[29] *ETSI EN 302 665 V1.1.1 - Intelligent Transport Systems (ITS); Communications Architecture*. Standard. European Telecommunication Standard Institute, 2010.

[30] *ETSI EN 303 613 V1.1.1 - Intelligent Transport Systems (ITS); LTE-V2X Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*. Standard. European Telecommunication Standard Institute, 2020.

[31] *ETSI TR 102 492-1 V1.1.1 - Electromagnetic compatibility and Radio spectrum Matters (ERM); Intelligent Transport Systems (ITS); Part 1: Technical characteristics for pan-European harmonized communications equipment operating in the 5 GHz frequency range and intended for critical road-safety applications; System Reference Document*. Technical Requirement. European Telecommunication Standard Institute, 2006.

[32] *ETSI TR 102 492-2 V1.1.1 - Electromagnetic compatibility and Radio spectrum Matters (ERM); Intelligent Transport Systems (ITS); Part 2: Technical characteristics for pan European harmonized communications equipment operating in the 5 GHz frequency range intended for road safety and traffic management, and for non-safety related ITS applications; System Reference Document.* Technical Requirement. European Telecommunication Standard Institute, 2006.

[33] *ETSI TS 101 539-1 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 1: Road Hazard Signalling (RHS) application requirements specification.* Technical Specification. European Telecommunication Standard Institute, 2013.

[34] *ETSI TS 101 539-2 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 2: Intersection Collision Risk Warning (ICRW) application requirements specification.* Technical Specification. European Telecommunication Standard Institute, 2018.

[35] *ETSI TS 101 539-3 V1.1.1 - Intelligent Transport Systems (ITS); V2X Applications; Part 3: Longitudinal Collision Risk Warning (LCRW) application requirements specification.* Technical Specification. European Telecommunication Standard Institute, 2013.

[36] *ETSI TS 102 637-1 V1.1.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 1: Functional Requirements.* Technical Specification. European Telecommunication Standard Institute, 2010.

[37] *ETSI TS 102 687 V1.2.1 - Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part.* Technical Specification. European Telecommunication Standard Institute, 2018.

[38] *ETSI TS 102 894-2 V1.2.1 - Intelligent Transport Systems (ITS); Users and applications requirements; Part 2: Applications and facilities layer common data dictionary.* Technical Specification. European Telecommunication Standard Institute, 2014.

[39] *FCC 03-324 - Amendment of the Commission's Rules Regarding Dedicated Short-Range Communication Services in the 5.850-5.925 GHz Band.* Standard. Washington, D.C: Federal Communications Commission, 2004.

[40] A. Filippi, K. Moerman, V. Martinez, A. Turley, O. Haran, and R. Toledano. "IEEE 802.11p ahead of LTE-V2V for safety applications." In: *Autotalks NXP* (2017).

[41] G. Pastor Grau, D. Pusceddu, S. Rea, O. Brickley, M. Koubek, and D. Pesch. "Vehicle-2-vehicle communication channel evaluation using the CVIS platform." In: *2010 7th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP 2010).* 2010, pp. 449–453. DOI: 10.1109/CSNDSP16145.2010.5580394.

[42] *Global status report on road safety 2018*. Technical Report. World Health Organization, 2018. URL: https://www.who.int/violence_injury_prevention/road_safety_status/report/.

[43] L. Güvenç, I. M. C. Uygan, K. Kahraman, R. Karaahmetoglu, I. Altay, M. Sentürk, M. T. Emirler, A. E. Hartavi Karci, B. Aksun Guvenc, E. Altug, M. C. Turan, Ö. S. Tas, E. Bozkurt, Ü. Ozguner, K. Redmill, A. Kurt, and B. Efendioglu. "Cooperative Adaptive Cruise Control Implementation of Team Mekar at the Grand Cooperative Driving Challenge." In: *IEEE Transactions on Intelligent Transportation Systems* 13.3 (2012), pp. 1062–1074. DOI: 10.1109/TITS.2012.2204053.

[44] *Haversine Formula*. URL: https://en.wikipedia.org/wiki/Haversine_formula.

[45] IEEE. *IEEE 802.11bd Working Group*. URL: https://www.ieee802.org/11/Reports/tgbd_update.htm.

[46] IEEE. *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008) - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. Tech. rep. 2020, pp. 1–499. DOI: 10.1109/IEEESTD.2020.9120376.

[47] *IEEE 1609.0-2013 - IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture*. Standard. Institute of Electrical and Electronics Engineers, 2014.

[48] *IEEE 1609.2-2016 (Revision of IEEE Std 1609.2-2013) - IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages*. Standard. Institute of Electrical and Electronics Engineers, 2016.

[49] *IEEE 1609.3-2016 (Revision of IEEE Std 1609.3-2010) - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services*. Standard. Institute of Electrical and Electronics Engineers, 2016.

[50] *IEEE 1609.4-2016 (Revision of IEEE Std 1609.4-2010) - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation*. Standard. Institute of Electrical and Electronics Engineers, 2016.

[51] *IEEE 802.11-2016 - IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. Standard. Institute of Electrical and Electronics Engineers, 2016.

[52] *Intelligent Transport Systems (ITS); Cross Layer DCC Management Entity for operation in the ITS G5A and ITS G5B medium*. Technical Specification. 2015.

[53] *Intelligent Transport Systems (ITS); Security; Security Services and Architecture*. Technical Specification. European Telecommunication Standard Institute, 2015.

[54] F. Kamal, E. Lou, and V. Zhao. "Design and validation of a small-scale 5.9 GHz DSRC system for vehicular communication." In: *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. 2012, pp. 1–4. DOI: 10.1109/CCECE.2012.6334893.

[55] *Kapsch Connected Vehicle Software Suite*. URL: https://www.kapsch.net/ktc/downloads/datasheets/software/KTC%5C_DB-V2X-Software%5C_web.pdf.

[56] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil. "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions." In: *IEEE Communications Surveys Tutorials* 13.4 (2011), pp. 584–616.

[57] J. B. Kenney. "Dedicated Short-Range Communications (DSRC) Standards in the United States." In: *Proceedings of the IEEE* 99.7 (2011), pp. 1162–1182.

[58] *LaTe + LaMP home page*. URL: https://francescoraves483.github.io/LaMP_LaTe/.

[59] S. Laux, G. S. Pannu, S. Schneider, J. Tiemann, F. Klingler, C. Sommer, and F. Dressler. "Demo: OpenC2X — An open source experimental and prototyping platform supporting ETSI ITS-G5." In: *2016 IEEE Vehicular Networking Conference (VNC)*. 2016, pp. 1–2. DOI: 10.1109/VNC.2016.7835955.

[60] Yunxin (Jeff) Li. "An Overview of the DSRC/WAVE Technology." In: *Quality, Reliability, Security and Robustness in Heterogeneous Networks*. Ed. by X. Zhang and D. Qiao. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 544–558. ISBN: 978-3-642-29222-4.

[61] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner. "Microscopic Traffic Simulation using SUMO." In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2575–2582. DOI: 10.1109/ITSC.2018.8569938.

[62] D. R. Mafioletti, A. B. Liberato, C. K. Dominicini, R. Villaça, M. Martinello, and M. Ribeiro. "Metherxis: Virtualized Network Functions for Micro-Second Grade Latency Measurements." In: *Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks*. LANCOMM '16. Florianopolis, Brazil: Association for Computing Machinery, 2016, pp. 22–24. ISBN: 9781450344265. DOI: 10.1145/2940116.2940131. URL: https://doi.org/10.1145/2940116.2940131.

[63] M. Malinverno, G. Avino, C. Casetti, C. F. Chiasserini, F. Malandrino, and S. Scarpina. "Edge-Based Collision Avoidance for Vehicles and Vulnerable Users: An Architecture Based on MEC." In: *IEEE Vehicular Technology Magazine* 15.1 (2020), pp. 27–35.

[64] M. Malinverno, G. Avino, C. Casetti, C. F. Chiasserini, F. Malandrino, and S. Scarpina. "Performance Analysis of C-V2I-Based Automotive Collision Avoidance." In: *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. 2018, pp. 1–9.

[65] M. Malinverno, J. Mangues-Bafalluy, C. Casetti, C. F. Chiasserini, M. Requena-Esteso, and J. Baranda. "An Edge-Based Framework for Enhanced Road Safety of Connected Cars." In: *IEEE Access* 8 (2020), pp. 58018–58031.

[66] M. Malinverno, F. Raviglione, C. Casetti, C. F. Chiasserini, J. Mangues-Bafalluy, and M. Requena-Esteso. "A Multi-Stack Simulation Framework for Vehicular Applications Testing." In: *Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*. DIVANet '20. Alicante, Spain: Association for Computing Machinery, 2020, pp. 17–24. ISBN: 9781450381215. DOI: 10.1145/3416014.3424603. URL: https://doi.org/10.1145/3416014.3424603.

[67] V. Mannoni, V. Berg, S. Sesia, and E. Perraud. "A comparison of the V2X communication systems: ITS-G5 and C-V2X." In: *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*. IEEE. 2019, pp. 1–5.

[68] *MK5 OBU, Cohda Wreless*. URL: https://cohdawireless.com/solutions/hardware/mk5-obu/.

[69] K. Moerman, A. Filippi, and V. Marnix. "On the 5GAA comparison between LTE-V2X and DSRC/IEEE 802.11p." In: *Proceedings of the IEEE* 99.7 (2019).

[70] R. Molina-Masegosa and J. Gozalvez. "LTE-V for Sidelink 5G V2X Vehicular Communications: A New 5G Technology for Short-Range Vehicle-to-Everything Communications." In: *IEEE Vehicular Technology Magazine* 12.4 (2017), pp. 30–39. DOI: 10.1109/MVT.2017.2752798.

[71] *ms-van3t*. URL: https://github.com/marcomali/ms-van3t.

[72] G. Naik, B. Choudhury, and J. Park. "IEEE 802.11bd 5G NR V2X: Evolution of Radio Access Technologies for V2X Communications." In: *IEEE Access* 7 (2019), pp. 70169–70184. DOI: 10.1109/ACCESS.2019.2919489.

[73] *Network Time Protocol Version 4: Protocol and Algorithms Specification*. Tech. rep. RFC, 2010. DOI: 10.17487/RFC5905.

[74] T. V. Nguyen, P. Shailesh, B. Sudhir, G. Kapil, L. Jiang, Z. Wu, D. Malladi, and J. Li. "A comparison of cellular vehicle-to-everything and dedicated short range communication." In: *2017 IEEE Vehicular Networking Conference (VNC)*. 2017, pp. 101–108. DOI: 10.1109/VNC.2017.8275618.

[75] *ns-3 Network Simulator*. URL: https://www.nsnam.org/.

[76] *OMNeT++ - Discrete Event Simulator*. URL: https://omnetpp.org/.

[77] *OpenWrt - V2X patch*. URL: https://github.com/francescoraves483/OpenWrt-V2X.

[78] *OpenWRT - Wireless freedom*. URL: https://openwrt.org/.

[79] *perfSONAR Project*. URL: https://www.perfsonar.net/.

[80] Q. Zhen, M. Zhen, Z. Xiaoyi, X. Bin, and Z. Lin. "Performance evaluation of 802.11p WAVE system on embedded board." In: *The International Conference on Information Networking 2014 (ICOIN2014)*. 2014, pp. 356–360. DOI: 10.1109/ICOIN.2014.6799704.

[81] F. Raviglione, M. Malinverno, and C. Casetti. "A Flexible, Protocol-Agnostic Latency Measurement Platform." In: *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. 2019, pp. 1–5.

[82] F. Raviglione, M. Malinverno, and C. Casetti. "Characterization and Performance Evaluation of IEEE 802.11p NICs." In: *Proceedings of the 1st ACM MobiHoc Workshop on Technologies, MOdels, and Protocols for Cooperative Connected Cars*. TOPCars '19. Association for Computing Machinery, 2019, pp. 13–18.

[83] F. Raviglione, M. Malinverno, and C. Casetti. "Demo: Open Source Platform for IEEE 802.11p NICs Evaluation." In: *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. 2019, pp. 1–3.

[84] *Rawsock library*. URL: https://github.com/francescoraves483/rawsock_lib.

[85] R. Riebl, H. Günther, 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)*. 2015, pp. 450–456. DOI: 10.1109/MTITS.2015.7223293.

[86] *SAE J2735 - Dedicated Short Range Communications (DSRC) Message Set Dictionary*. Standard. Society of Automotive Engineers, 2016.

[87] *SAE J2945 - On-Board System Requirements for V2V Safety Communications*. Standard. Society of Automotive Engineers, 2016.

[88] *SAE J3016 - Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*. Standard. Society of Automotive Engineers, 2014.

[89] J. Sherry. "Applications of the IP timestamp option to Internet measurement." In: (2010).

[90] C. Sommer, R. German, and F. Dressler. "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis." In: *IEEE Transactions on Mobile Computing* 10.1 (2011), pp. 3–15. DOI: 10.1109/TMC.2010.133.

[91] *The ASN.1 to C compiler*. URL: http://lionet.info/asn1c/.

[92] *Toyota and Lexus to Launch Technology to Connect Vehicles and Infrastructure in the U.S. in 2021*. URL: https://pressroom.toyota.com/toyota-and-lexus-to-launch-technology-connect-vehicles-infrastructure-in-u-s-2021/.

[93] *Unex V2X solutions*. URL: https://unex.com.tw/.

[94] *V2X Functional and Performance Test Report; Test Procedures and Results*. White paper. 5GAA, 2019.

[95] *V2X White Paper*. White paper. NGMN V2X Task Force, 2018.

[96] A. Virdis, G. Stea, and G. Nardini. "SimuLTE - A modular system-level simulator for LTE/LTE-A networks based on OMNeT++." In: *2014 4th International Conference On Simulation And Modeling Methodologies, Technologies And Applications (SIMULTECH)*. 2014, pp. 59–70. DOI: 10.5220/0005040000590070.

[97] N. Vivek, P. Sowjanya, B. Sunny, and S. V. Srikanth. "Implementation of IEEE 1609 WAVE/DSRC stack in Linux." In: *2017 IEEE Region 10 Symposium (TENSYMP)*. July 2017, pp. 1–5. DOI: 10.1109/TENCONSpring.2017.8070033.

[98] *Volkswagen Group assumes pioneering role in rapid road safety improvement*. URL: https://www.volkswagenag.com/en/news/2018/02/volkswagen_group_rapid_road_safety.html.

[99] V. Vukadinovic, K. Bakowski, P. Marsch, I. Garcia, H. Xu, M. Sybis, P. Sroka, K. Wesolowski, L. David, and I. Thibault. "3GPP C-V2X and IEEE 802.11p for Vehicle-to-Vehicle communications in highway platooning scenarios." In: *Ad Hoc Networks* 74 (Mar. 2018). DOI: 10.1016/j.adhoc.2018.03.004.

[100] M. Wang, M. Winbjork, Z. Zhang, R. Blasco, H. Do, S. Sorrentino, M. Belleschi, and Y. Zang. "Comparison of LTE and DSRC-Based Connectivity for Intelligent Transportation Systems." In: *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*. 2017, pp. 1–5. DOI: 10.1109/VTCSpring.2017.8108284.

[101] A. Wegener, M. Piorkowski, M. Raya, H. Hellbrück, S. Fischer, and J. Hubaux. "TraCI: An Interface for Coupling Road Traffic and Network Simulators." In: *Proceedings of the 11th Communications and Networking Simulation Symposium, CNS'08* (2008). DOI: 10.1145/1400713.1400740.

[102] *White Paper on ITS spectrum utilization in the Asia Pacific Region*. White Paper. 5GAA, 2018.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was LuaLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.