Politecnico di Torino

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

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

Open Platforms for Connected Vehicles

Francesco Raviglione

* * * * *

Supervisors
Prof. Claudio Ettore Casetti, Supervisor

Doctoral Examination Committee:
Prof. Laura Galluccio, Università di Catania
Prof. Francesco Gringoli, Università di Brescia
Prof. Jérôme Härri, EURECOM
Prof. Alessandro Bazzi, Università di Bologna
Prof. Paolo Giaccone, Politecnico di Torino

Politecnico di Torino
December 14th, 2022

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.

. . . . . .

Francesco Raviglione

Turin, December 14$^{\text{th}}$, 2022

Summary

Vehicular technologies are being recently invested by astonishing innovations, aimed at making the future vehicles safer, smarter and greener. Future transportation systems are indeed expected to heavily leverage connected and autonomous vehicles, which have emerged into the market in the past decade.

In the past, since the introduction of the Anti-lock Braking Systems (ABS), almost 50 years ago, driving assistance and automation systems leveraged only onboard sensors. However, these sensors are characterized by a limited range, and cannot enable the most advanced use cases, such as collision avoidance to increase road safety.

This led to the birth of vehicular communication, represented by the data exchange between road users through dedicated wireless networks. Vehicle-to-Everything (V2X) communication thus become the fundamental enabler for all the next-generation of automotive services, towards the sought-after very high and full automation levels defined by the Society of Automotive Engineering. Vehicular networks are, however, characterized by several challenges due to the high mobility of nodes and the frequent topology changes. Furthermore, as most V2X applications are safety critical, they should be able to guarantee a very low latency and high network reliability.

This thesis starts by presenting an overview on the main use cases for connected and autonomous vehicles and on the different types of V2X communications. Then, it provides an analysis of the technologies specifically designed for vehicular networks. These wireless technologies have been standardized with the aim of overcoming the challenges of V2X and include IEEE 802.11p, LTE-V2X and NR-V2X, concerning the lower layers, and the ETSI ITS-G5 and IEEE WAVE protocol stacks for the higher layers.

A fundamental role in the development, deployment and testing of innovative V2X systems is played by open and customizable platforms. Indeed, the open source paradigm provides several advantages and enables knowledge sharing between researchers, and between academia and industry. Open platforms enable the reproducibility of results, foster further research, provide an unprecedented flexibility for experimental purposes, represent a low-cost way of analyzing and testing new protocols and applications and provide transparency and reliability, thanks to

the code being open to contributions.

This thesis thus focuses on open platforms and frameworks for connected vehicles, including the presentation of a fully-open Dedicated Short-Range Communications (DSRC) platform based on customizable hardware and a special version of the OpenWrt Linux distribution, effectively enabling V2X communication in the 5.8-5.9 GHz Intelligent Transport Systems (ITS) frequency range. Then, it presents a novel latency measurement protocol (LaMP) for automotive scenarios and the first tool relying on this protocol, called LaTe. The combination of LaTe and our open DSRC framework allowed us to perform several measurements to provide interesting insights on the behaviour of IEEE 802.11p, both concerning throughput and latency. Our platform has also been leveraged as a starting point for the development of an open, plug and play and low cost On-Board Unit (OBU), with open source software implementing the ETSI ITS-G5 stack. Furthermore, thanks to LaTe, it was possible to develop a long term network testing framework (LTNT), leveraged to assess the performance of an innovative automotive edge system, showing how it can provide low latency and high throughput.

The presentation of open platforms for V2X is complemented with the design, development and evaluation of both Edge-V, the first open framework enabling Vehicular Edge Intelligence thanks to the combination of unlicensed spectrum technologies, and Open Radio Network Information eXchange (ONIX), a flexible Radio Network Information Service designed to support 4G/5G networks and vehicular applications such as Collision Avoidance.

The evaluation of different technologies for V2X also plays a crucial role in development and deployment of an open platform for connected vehicles. Indeed, these technologies represent fundamental enablers for the overlying applications and use cases.

Therefore, this thesis presents a new open source framework for simulating and emulating V2X scenarios, called ms-van3t, which is compared with other existing solutions in literature, highlighting several novel features enabling advanced testing of vehicular applications. It is then exploited to provide a comparison study, thanks to realistic simulations, between IEEE 802.11p and C-V2X. The simulation studies are further complemented by several measurements in the field, both in a static laboratory context and on the road. These field tests provide several insights on the performance of IEEE 802.11p, which is also compared with millimeter Wave (mmWave) and IEEE 802.11ac in a Vehicle-to-Infrastructure scenario.

Finally, a novel open protocol for the exchange of raw Global Navigation Satellite System (GNSS) data between vehicles is proposed and evaluated. Thanks to this protocol, which is proved to reliably support both IEEE 802.11p and C-V2X, it becomes possible to enable a plethora of collaborative positioning approaches. We also present an innovative, 5G-enabled MEC service storing a centralized local dynamic map of the road, aimed at providing data to other MEC services, when needed, to manage highly automated maneuvers in a centralized way. This service

is evaluated, both in-lab and in the field, after being deployed to multiple MEC platforms managed by different Mobile Network Operators. Two Stellantis vehicles equipped with V2X OBUs have been used to test our service in the field.

The design and development of open platforms, the evaluation of different technologies (by simulations and in the field) and the deployment of novel protocols and services all represent fundamental steps in the deployment and testing of complete open V2X environments. As discussed in this thesis, this enables researchers to gather insights on different technologies for V2X and on their combination and test innovative applications for future road deployment, towards a new generation of safer, smarter and greener vehicles.

Acknowledgements

Before delving into the main content of this thesis, I want to thank all the people who supported me during my Ph.D. program on connected vehicles at the Department of Electronics and Telecommunications (DET). First, I would like to thank my tutor, Prof. Claudio Ettore Casetti, for all the incredible support, for his suggestions and core teachings during the whole duration of my doctoral program, and for the great opportunities he gave me. Then, I would like to thank all the people who worked with me during these years, and with whom I learned and produced a lot, especially Prof. Carla Fabiana Chiasserini, and all my friends and colleagues in our laboratory, both at DET and at the Department of Control and Computer Engineering (DAUIN).

Thanks a lot to my family, my dad Marco and mom Silvia, my brother Luca, and all my friends, who always supported me in these three years.

Thanks to Prof. Francesco Restuccia, who invited me to spend a period abroad in Portland, Maine, USA, giving me the special opportunity to learn a lot and meet people with whom I started several collaborations, and who I would like to thank for the wonderful experience in the US.

Thanks to all the partners of the 5G-CARMEN European project, which started almost together with my Ph.D. program and terminated as I am writing these lines. Thanks especially to the partners with whom we interacted most, and worked with us to develop and test innovative services such as the S-LDM, described in Chapter 5: the other universities participating as CNIT, Centro Ricerche Fiat, NEC Laboratories Europe, BMW and Motius, Fondazione Bruno Kessler, TIM.

Contents

Chapter 1

Introduction

Vehicular technologies are experiencing a significant evolution in recent years, as they aim at providing high levels of automation and increasingly complex centralized and decentralized services. The path towards automated vehicles began with the introduction of ADAS (Advanced Driver-Assistance Systems), which dates back to the 1970s, with the first deployment of ABS (Anti-lock Braking Systems) around 50 years ago [1].

The following years saw the introduction of more complex systems, towards the so-called SAE (Society of Automotive Engineers) Automated Driving Levels 1 and 2. SAE Level 1 requires the vehicle to be fully controlled by the driver, which is assisted in certain specific circumstances, while Level 2 enhances the vehicle to perform combined actions and enables more complex automated maneuvers. Both levels of automation require the driver to remain engaged all the time.

Level 1 systems include Cruise Control, Adaptive Cruise Control (which combines Cruise Control with data from radars to enable an automatic speed regulation depending on the behaviour of the preceding vehicle [2]), Lane Keeping-Assist (LKA) [3], while Autonomous Parking represents a Level 2 feature [1].

ADAS systems, however, are subject to several limitations. As an example, a camera sensor cannot detect objects in blind spots or Non-Line Of Sight (NLOS), and a radar or LiDAR is able to provide information on other road users (such as other cars or pedestrians) over a limited range.

Vehicular communication, represented by the data exchange between road users through dedicated wireless networks, becomes thus a fundamental enabler for all the next-generation automotive services, with the aim of reaching the sought-after SAE Levels 3 to 5 [1], towards high (Level 4) and full (Level 5) automation. As mentioned in the next Section and depicted in Figure 1.1, SAE defines five levels of automation (plus a "Level 0" for "No automation"), with level 5 representing fully autonomous vehicles, in which the steering wheel could be potentially removed [4].

Exchanging data between vehicles faces several important challenges, as vehicular networks, commonly referred to as VANETs (Vehicular Ad-Hoc Networks),

are subject to high nodes mobility, very dynamic topology changes, and high reliability and ultra low latency requirements [5]. When the penetration rate (i.e., the amount of vehicles equipped with vehicular communication technologies) will increase, channel congestion will also become an impacting issue. Furthermore, several projects funded by the European Commission are focusing on the development of centralized services, which can enable highly automated maneuvers, trough Vehicle-to-Infrastructure (V2I) communication. These services have a wider and more complete view of the road, as opposed to decentralized services relying solely on Vehicle-to-Vehicle (V2V) data exchange, which may not be enough to enable SAE Level 4 and beyond automation [6].

It becomes thus crucial to develop, deploy and evaluate different technologies and applications for both V2I and V2V communication, proposing, when possible, a combination of different standards to improve the vehicular communication capabilities available nowadays. The research world, together with the automotive industry, is increasingly focusing on the development of intelligent applications and services for autonomous vehicles and smart mobility, together with the deployment of new protocols and the assessment, both in laboratory and in the field, of several algorithms and technologies enabling the so-called Vehicle-to-Everything (V2X) communications.

Open platforms play a significant role in this broad and complex field, as there is currently a scarcity of open source solution for connected vehicles, and most products and software is owned by commercial companies or not disclosed to the public. Open solution are indeed of vital importance for the research community, as they enable the reproducibility of result, an easy and legal customization of services, protocols and applications for experimental purposes, and foster knowledge sharing between researchers, and between researchers and industry.

These platforms can then be exploited (i) to determine the advantages and disadvantages of different access technologies, (ii) to develop and test algorithm and services for automated driving, (iii) to assess the performance and propose improvements to existing and promising V2X technologies (such as 5G and New Radio V2X - NR-V2X), and many more.

Concerning the standardization efforts in the V2X field, it is worth mentioning how existing protocols could not be selected for automotive use cases due to the peculiar characteristics of VANETs, as mentioned earlier. The main international standardization bodies are thus focusing on the development of specialized standards for the automotive field, to cope with the challenges of high mobility and low latency requirements, which will be quite stringent for most V2X applications. Indeed, latency requirements will be as low as maximum 50 ms end-to-end, depending on the specific application and according to the European Telecommunications Standards Institute (ETSI) [7]–[10].

The main standardization bodies working on the development of standards for V2X include IEEE (Institute of Electrical and Electronics Engineers) and ETSI,

concerning the Wi-Fi-based standards (IEEE 802.11p, the upcoming IEEE 802.11bd, IEEE 1609.x, ETSI ITS-G5), and 3GPP (Third Generation Partnership Project), concerning an emerging and very promising application of cellular networking to VANETs, referred to as C-V2X (Cellular V2X). IEEE 802.11p is an IEEE 802.11 amendment with specific features targeted at the high mobility and requirements of VANETs, being an implementation of the so-called DSRC (Dedicated Short-Range Communications), which will be described more in details in Chapter 2. ETSI Intelligent Transport Systems (ITS)-G5 focuses instead on the higher layers of the stack, defining two Transport and Networking protocols, together with several messages types and Facilities for the standardized data exchange between vehicles. Alongside these specifically-targeted standards, a few other promising technologies are emerging for application to vehicular networks. These include, for instance, standard Wi-Fi (IEEE 802.11ac, IEEE 802.11ax) for V2I Internet access, or millimeter Wave (mmWave) for very high throughput and low latency short range data exchange [11], [12].

Therefore, it becomes evident how the comparison of different technologies, and the analysis of solutions combining various standards, is of vital importance in the V2X field, and open and affordable platforms play again a fundamental role in this context.

A large part of this thesis is focused on presenting novel solutions for the evaluation of V2X protocols and applications both in a simulated environment and in the field. These solutions include *ms-van3t*, a novel framework for small to large scale V2X simulation and emulation with a complete ETSI ITS-G5 implementation, *LaTe*, a Linux-based tool for advanced latency measurements in the automotive field, *Edge-V*, a framework combining mmWave with DSRC to enable Vehicular Edge Intelligence (VEI) use cases, together with an *open source testbed* for DSRC-based vehicular communications. Several results, gathered both in the field and in simulation, are also presented to extensively evaluate different technologies for V2X, providing at the same time a variety of novel insights related to different metrics such as latency, packet loss and throughput.

1.1 Vehicular communication motivations and research directions

As mentioned earlier, vehicular networks are at the core of the technological evolution in the automotive field. They are indeed enabling use cases that promise to make future transportation systems *safer*, *greener* and *smarter* [13], other than enhancing the *level of automation* currently available in vehicles.

Nowadays, one of the most impacting issues in transportation systems is represented by road fatalities, which are unacceptably high. A report by the US National Highway Traffic Safety Administration states how road fatalities increased

from 2020 to 2021, with a total of 42915 fatalities in 2021 only in the United States [14].

Starting from these figures, safety applications have emerged as one of the most relevant use cases enabled by vehicular networks. Indeed, an application such as *intersection collision avoidance* can greatly help in detecting possible hazardous situations and preventing (possibly fatal) collisions, especially under high penetration rates [15].

Communication between road entities can also help in noticeably improving the environmental footprint of urban transportation systems, with use cases such as Green Light Optimized Speed Advisory (GLOSA), which provides the driver with information about green phases of traffic lights at intersections. This information can then be leveraged to minimize the stop-and-go maneuvers, reducing pollutants by up to 22% [16].

Concerning instead the last point, as briefly described in the previous Section, SAE defines five levels of automation, summarized in Figure 1.1 [1], [4].

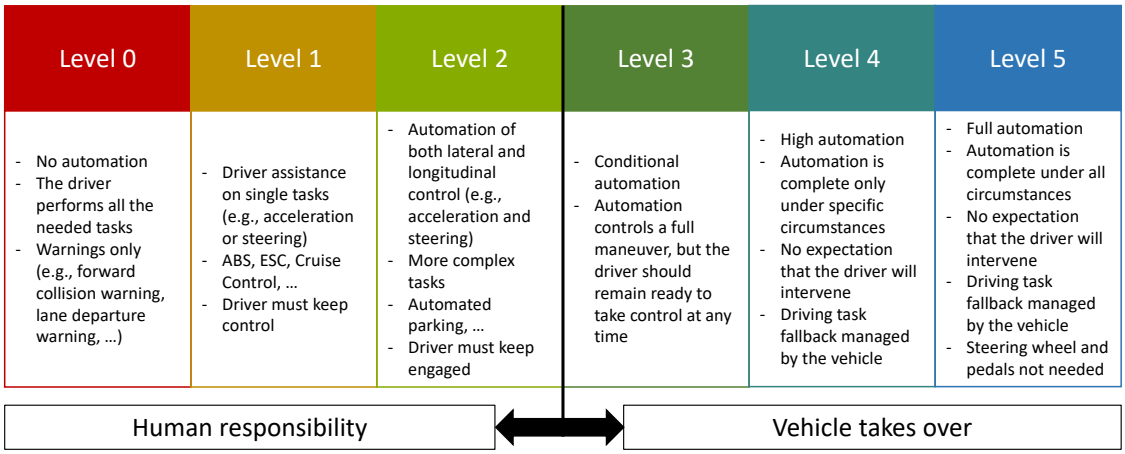


Figure 1.1: SAE Automation Levels according to the SAE J3016 standard.

As can be seen, starting from SAE Level 3, the complexity of the system increases exponentially, to let the vehicle take control of full maneuvers. Relying solely on ADAS makes this task extremely challenging, and this explains why advanced use cases, such as *platooning*, in which several vehicles travel together like a "train of cars", or *automated lane merge*, in which vehicles coordinate between each other to let another vehicle merge on the same lane from a different one, require vehicles to exchange data, as part of a vehicular network.

In this context, several projects funded by European Union are investigating high levels of automation enabled by vehicular networks, both decentralized, thanks to V2V communication, and centralized, thanks to V2I and 5G networks.

One of these projects, which ended in 2022 after almost four years of development, and involving 25 industrial and academic partners, is 5G-CARMEN[1]. The consortium focused on the development of a 5G cross-border connected corridor along the Modena-Munich motorway, aimed at reaching Level 4 automation thanks to 5G connectivity. The project, thanks to the cooperation between automotive manufacturers and Mobile Network Operators (MNOs), tested several advanced, mainly centralized, uses cases enabled by a 5G Non-Standalone installation and specialized Multi-Access Edge Computing (MEC) platforms deployed on Italy, Austria and Germany.

One of the main aims was to analyze the cross-border aspects of 5G communication applied to connected and automated cars, and to determine the advantages and limitations of such approaches, to foster the future mass deployment of 5G for V2X in Europe [17].

As part of the project, the feasibility of centralized services for automated driving was analyzed and successfully tested, showing the importance of vehicular communication when shifting towards high levels of automation. A novel service for centralized automated driving through 5G is presented in Chapter 5.

Together with the application of C-V2X and 5G to V2X, the automotive field is currently researching the application of Artificial Intelligence and Machine Learning (AI/ML) to vehicular networks, the deployment of novel technologies, such as mmWave [11], together with DSRC and C-V2X, or Terahertz communication [18], and the development of novel safety and "green" services for connected vehicles and urban mobility, with a focus on Vulnerable Road Users (VRUs) [19].

1.2 Main contributions

This Section summarizes the main contributions of this thesis:

- **Chapter 2**. Introduction to the main standards, protocols and use cases for connected vehicles. Concerning the standards, both Wi-Fi based (DSRC) and cellular-based (C-V2X) standards are presented, with a more extensive description of ETSI ITS-G5, which, thanks to its modular structure, can be adapted to work both with DSRC and C-V2X. Different types of vehicular communication and use cases are also presented, together with future directions. These include the transmission and reception of ETSI ITS-G5 and standardized vehicular messages over the Advanced Message Queuing Protocol (AMQP) protocol.
- **Chapter 3**. Brief presentation of the state-of-the-art in embedded systems

[1] https://5gcarmen.eu/

for connected vehicles, and presentation of a novel, low-cost and open hardware and software platform for DSRC-based vehicular communications. This platform includes OpenWrt-V2X [20], a patched version of the OpenWrt Linux distribution enabling DSRC communication over IEEE 802.11p at the dedicated 5.8/5.9 GHz spectrum. This system is then used to perform extensive DSRC technology assessment both in laboratory and in the field, focusing on latency, throughput, packet loss, and reachable radio range. One of the major advantages of our platform is represented by its openness, possibly enabling anyone with a supported Network Interface Card (NIC) to build a V2X device and test applications on the field.

- **Chapter 3**. Design and specification of a novel application-layer protocol for precise latency measurements, up to one microsecond accuracy, named LaMP (Latency Measurement Protocol) and released with open specifications. This protocol is characterized by being *lower layer agnostic*, thus enabling measurements over any lower layer protocol, such as User Datagram Protocol (UDP), Transmission Control Protocol (TCP), ETSI GeoNetworking [21] or even directly encapsulated over Ethernet. This protocol is focused on automotive applications, but it has been designed to be leveraged for the analysis of any network architecture, including industrial 5G Ultra-Reliable Low Latency Communication (URLLC). As opposed to other protocols commonly used for latency measurements, such as ICMP within `ping`, it provides a realistic application-layer latency, without placing itself at a specific layer of the ISO/OSI stack. Protocols such as LaMP are thus fundamental to evaluate the real latency performance of a plethora of applications, possibly leveraging different communication protocols. The first open source tool implementing LaMP is also presented. This tool, named LaTe (Latency Tester), currently supports LaMP over UDP (which represents the most suitable IP-based protocol for vehicular networks) and LaMP over AMQP. LaTe also provides several advances features, not available with other tools like `ping`, including the possibility of selecting an IEEE 802.11p traffic priority when launched within OpenWrt-V2X [20]. Several results are also presented to both validate LaTe and assess the latency performance of DSRC with embedded V2X devices.

- **Chapter 3**. Starting from LaTe and `iperf`, the de-facto state-of-the-art tool for throughput measurement, a novel long-term networking testing framework, called LTNT (Long Term Network Tester) [22] is presented. This framework requires two hardware boards that act respectively as master and slave and should be connected to both ends of the network architecture under test. It combines LaTe and `iperf` with an efficient open source orchestrator and a precise clock synchronization tool that coordinates the execution of latency and throughput measurements, providing safety mechanisms to automatically

manage and restart the tests in case of system failures. It can then output several timestamp-tagged logs with both unidirectional and bidirectional latency and throughput measurements, which can span over a very long time (up to several months of continuous testing). The obtained data can be exploited to thoroughly analyze the performance of the system under study during the time and detect possible issues. LTNT has been used to assess the performance of an experimental 5G automotive edge platform.

- **Chapter 3**. Design, development and testing of a prototype for a plug-and-play On-Board Unit (OBU), targeted at equipping already circulating vehicles with low cost and effective V2X technology. This project foresees an open implementation of ETSI ITS-G5, and combines off-the-shelf hardware with a custom architecture, software and Operating System, i.e., OpenWrt-V2X [20]. Lane-level positioning is provided thanks to a modular Real-Time Kinematic (RTK) Global Navigation Satellite System (GNSS) hardware. The developed prototype is then leveraged to perform interoperability tests with commercial Road Side Units (RSU).

- **Chapter 3**. Analysis, design and development of the first open framework, called Edge-V, combining different technologies (DSRC, mmWave and standard Wi-Fi) to enable Vehicular Edge Intelligence and on-board AI/ML without leveraging the already congested cellular networks. The framework accounts for the tight latency and throughput requirements of the emerging applications such as image segmentation and real-time analysis and sharing of sensor data. The advantages of the proposed solution, other than being based on open source software, are proved thanks to both laboratory and field tests with real vehicles.

- **Chapter 3**. With the emergence of the Multi-access Edge Computing (MEC) paradigm, which moves the computational resources at the network edge, latency can be substantially reduced with respect to standard cloud-based approaches. Given the importance of edge computing, ETSI has standardized its architecture and defined a very important component called Radio Network Information Service (RNIS). This component allows MEC applications to obtain information about current radio conditions and wireless network reliability, which can be very important in mission-critical vehicular applications. It can also enable on-board services such as adaptive video streaming. As very few works in the literature systematically discuss the requirements and challenges for the design of a scalable RNIS system, we provide an analysis on the use cases and challenges of RNIS and propose Open radio Network Information eXchange (ONIX), a scalable open RNIS architecture. ONIX is evaluated through several experiments focusing on average latency and resource consumption (in terms of CPU and RAM) and showing how it can scale with the

number of active users.

- **Chapter 4**. Design, development and validation of an open source simulation and emulation framework called *ms-van3t*, available on GitHub [23]. This framework supports several advanced features not available in other similar solutions (such as the possibility of relying on pre-recorded GNSS traces instead of a mobility model), and enables the simulation of large scale V2X scenarios based on the ETSI ITS-G5 standards. It also provides an emulation Hardware-In-the-Loop (HIL) feature, which enables simulated vehicles to communicate with real entities, such as other vehicles or centralized edge services. The framework is presented together with two significant applications, which prove the advantages of V2X-enabled solutions and showcase the capabilities of ms-van3t. It is finally exploited to provide a comparison study between C-V2X and IEEE 802.11p, which is still a hot topic in research. ms-van3t is currently the only framework integrating ETSI ITS-G5 with the ns-3 network simulator [24].

- **Chapter 4**. After the analysis of V2X technologies in a simulated environment, we present two field test campaigns, aimed at evaluating different IEEE 802.11-based technologies in the field, to assess their performance in a real-world environment. Indeed, as simulation tools are based on models, they only capture the most important aspects of reality. Field testing is thus fundamental, together with simulations and hardware-in-the-loop emulation, to really evaluate the capabilities of different technologies. Furthermore, field testing enables *(i)* the analysis of technologies considering the characteristics of the environment and of the hardware (the latter is typically never considered when simulating a V2X scenario), and *(ii)* the evaluation of how the context and the environmental conditions (which cannot be modeled perfectly) affect the performance of different protocols. The main focus of the field tests are latency, throughput, Received Signal Strength Indicator (RSSI) and reachable radio range in both Line-Of-Sight (LOS) and Non-Line-Of-Sight (NLOS) conditions. The second test campaign focuses, to the best of our knowledge for the first time, on the comparison in the field between DSRC, standard Wi-Fi and mmWave at 60 GHz.

- **Chapter 5**. Design, specification and evaluation of a novel protocol for the exchange of raw GNSS data between vehicles, namely the CEM (Cooperative Enhancement Message) protocol. Several Cooperative Positioning (CP) algorithms have been recently proposed with the aim of *(i)* enhancing the localization accuracy, especially in harsh environments, *(ii)* providing positioning integrity, *(iii)* enabling distributed time synchronization through cooperative exchange of raw positioning information, and many more GNSS-related applications. These approaches, when applied to vehicular networks, require the

exchange of raw GNSS data between road entities. However, there is currently no open and ETSI-compliant protocol for such a purpose, and most of the literature on CP gives the underlying network for granted. On the other hand, most V2X literature typically assumes positioning as very accurate, which is often not the case, especially in urban scenarios. To cope with this issue, we propose the CEM protocol and evaluate its usage both with IEEE 802.11p and C-V2X, thanks to a special version of ms-van3t [25].

- **Chapter 5**. Finally, we present an innovative service aimed at enabling centralized highly automated maneuver, developed in the context of the 5G-CARMEN project. 5G-enabled MEC services for automated maneuver management require the reception of data from vehicles. However, they very often require only a subset of pre-processed data, related to a specific "context" on the road (e.g., only data of vehicles and other non-connected objects involved in a centralized automated maneuver), which corresponds to a "map" of a portion of the road. This map should always be up-to-date and should be ready to provide data to other safety-critical MEC services, guaranteeing high reliability, low latency and offloading them from the burden of receiving and processing a huge amount of raw data from vehicles. We thus propose the novel Server Local Dynamic Map (S-LDM) service, which hinges upon the LDM concept by ETSI [26]. The S-LDM is an open source [27] 5G-enabled MEC service storing a centralized local dynamic map of the road, containing the most up-to-date and historical data of all vehicles (and other non-connected objects detected thanks to sensors) travelling in a given area. This service acts as "middleware" and can very efficiently provide a filtered and processed version of this data to other MEC services, when it detects certain "triggering" conditions on the road (e.g., a vehicle trying to perform a centralized lane merge). The S-LDM is evaluated both in a laboratory environment and through road test with real vehicles from Stellantis and the commercial 5G network from TIM.

Part of the work mentioned earlier has been presented either in scientific journals or in well-known international conferences focused on vehicular technologies and telecommunication networks. In particular, the open platform for the performance assessment and evaluation of DSRC-based vehicular communications has been presented for the first time in [28] and [29]. Both publications are related to interactive demonstrations in which users could directly interact with a DSRC system, experience in real-time the effect of the different traffic classes foreseen by IEEE 802.11p and tune the parameters of a low latency video streaming service. Then, the LaMP protocol, together with the LaTe tool and its validation, is included in [30], presented at the VTC2019-Fall main conference in Honolulu, HI, USA. Edge-V, instead, has been submitted to IEEE INFOCOM 2023.

Furthermore, the ONIX RNIS architecture has been proposed as part of an article published in IEEE Communications Magazine [31], while ms-van3t is described

in [32], which has been presented during the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications (DIVANet, hosted at ACM MSWiM 2020).

All the field test and performance assessment campaigns have been presented as part of two different contributions: (i) [33] further presents and details the platform introduced in [28], [29] and describes the results of several static tests, to study the performance of IEEE 802.11p cards for vehicular communication with the aim to provide a baseline characterization in a controlled environment, (ii) [12] presents instead the experimental assessment of three different IEEE 802.11-based technologies, focusing on V2I communications and comparing for the first time mmWave (IEEE 802.11ad) with DSRC (IEEE 802.11p), in the field and with open and low-cost devices.

Concerning instead the CEM protocol, a preliminary version and its validation has been presented at the 2021 IEEE Globecom Workshops [34], while an extension of the whole work, including updates to the protocol and its complete evaluation with IEEE 802.11p and C-V2X is included in [7], recently published in Elsevier Vehicular Communications.

Finally, the S-LDM has been introduced at the VTC2022-Spring conference in Helsinki, as part of the paper titled "S-LDM: Server Local Dynamic Map for Vehicular Enhanced Collective Perception" [6].

1.3 Outline

The remainder of this thesis is organized as follows:

- Chapter 2 presents an overview of the different V2X communication types and uses cases, followed by an analysis of the main standards and protocols for connected vehicles. It presents also future research directions, such as the usage of AMQP in the context of vehicular networks, and a brief description of intra-vehicle communications.

- Chapter 3 is the core of the thesis work and focuses on open source embedded solutions for V2X. It presents the state-of-the-art of embedded devices for vehicular networks, followed by the design, description and evaluation of different open platforms for V2X, as mentioned in Section 1.2. These solutions span from an open source testbed enabling low cost and customizable DSRC communications, to an open source advanced latency measurement tool (LaTe).

- Chapter 4 focuses on the performance assessment of V2X technologies and applications, both in a simulated environment, in our laboratory and in the field, using most of the solutions presented in the previous Chapter. This Chapter also presents the novel ms-van3t simulation and emulation framework.

- Chapter 5 provides the details of an innovative protocol for the exchange of raw GNSS data between vehicles and between vehicles and the infrastructure, namely, the CEM protocol, which is also evaluated in simulation in a realistic urban scenario. It also presents a novel centralized edge service, the Server Local Dynamic Map (S-LDM), collecting information from vehicles and storing it inside a very efficient database. This information is then provided, when needed, to other latency-critical MEC services managing highly automated maneuvers in a centralized way.
- Chapter 6 presents the conclusions and main take-away messages.

Chapter 2

Communication protocols and technologies for connected vehicles

This Chapter aims at introducing the reader to the most important concepts and standards for V2X.

Before delving into the different types of vehicular communications, it is worth describing the Multi-Access Edge Computing (MEC) paradigm, which has recently gained importance in the telecommunication field, and, in turn, in vehicular networks.

When deploying centralized services, latency represents one of the most critical requirements. It is thus mandatory to try to reduce the delay between the vehicle and the service, and the overall end-to-end delay.

Linked with this, other services which are emerging with 5G require an ultra-low latency, which is typically not reachable with remote cloud services. Indeed, as stated in the previous Chapter, ETSI foresees a maximum end-to-end latency of 50 ms for vehicular services, which is reduced down to 5 ms when looking at the performance requirements identified for the emerging 5G services [35].

This has led to the emergence of the MEC paradigm, in which the computing capabilities are moved at the edge, i.e., near to the end-users. An edge server, providing a service, could be for instance located nearby or at an RSU, or in proximity of a cellular base station, in case of cellular networks.

Therefore, MEC helps to reach the real-time requirements between devices and applications, in terms of both latency and network bandwidth [36].

Figure 2.1 exemplifies the advantages of a MEC approach with respect to a cloud-only paradigm, taking as a reference a cellular network architecture. Packets exchanged by the red vehicle, subscribed to a cloud service, will need to traverse the whole core network, and, possibly, several routers on the Internet, before reaching

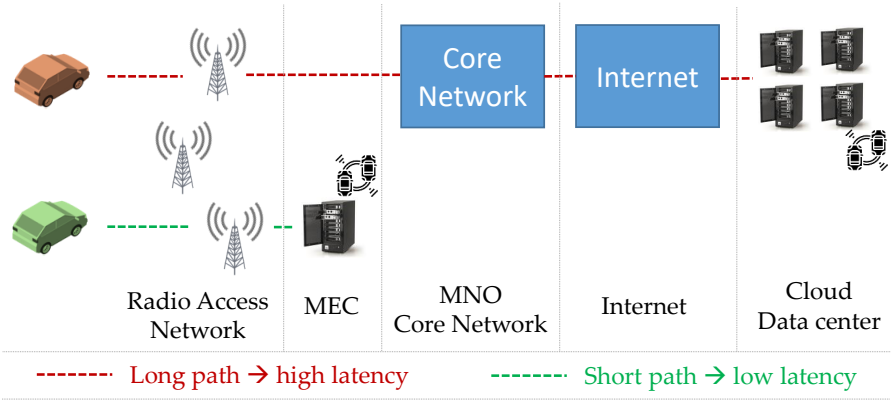


Figure 2.1: Scheme showing the Multi-Access Edge Computing (MEC) paradigm, taking as reference a cellular network and a centralized collision avoidance service (represented by the black icon with two vehicles). The red vehicle will experience a much higher latency than the green vehicle, which leverages the same service deployed on a MEC server, at the edge of the network.

the actual service. The traffic generated by the green vehicle, instead, will be able to directly reach an edge server located in proximity of the base station, experiencing a much reduced latency.

2.1 Types of Vehicle-to-Everything (V2X) communications

As briefly mentioned in the previous Chapter, the term Vehicle-to-Everything (V2X) is commonly used when referring to the data exchange between road users through a wireless network, and it is often used to discuss about vehicular networks in general.

Even though the most common type of communication a reader can think of is represented by Vehicle-to-Vehicle (V2V), i.e., the exchange of information between cars, vehicular networks refer to a more complex ecosystem, in which other players are involved too. These road users may be trucks, pedestrians, infrastructure nodes, remote servers providing services to vehicles, and so on.

It is thus possible to distinguish several sub-use cases of V2X. The list provided below is not exhaustive, but aims at listing the most common types of vehicular communications, as defined by 3GPP [37].

These types are schematized in Figure 2.2 and include:

- Vehicle-to-Vehicle (V2V), referring to the direct data exchange between vehicles (which include also trucks and emergency vehicles), thanks to Wi-Fi-based

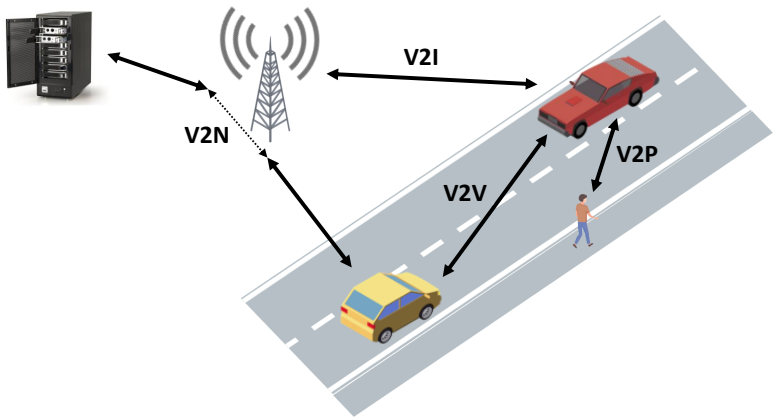


Figure 2.2: Scheme showing the most common sub-use cases of V2X communication: Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Network (V2N) and Vehicle-to-Pedestrian (V2P).

or cellular-based technologies, thanks to the *sidelink* communication. Each vehicle is equipped with an On-Board Unit (OBU), i.e., a communication device providing on-board services and enabling V2X.

- Vehicle-to-Infrastructure (V2I), when a vehicle transmit and receives messages to/from an infrastructure node, typically referred to as Road Side Unit (RSU). An RSU is a fixed device located along the road, acting as a fixed access point. It provides additional services and support to connected vehicles, often together with some level of connection to the Internet. The RSU can host edge services itself, and it can be controlled by road operators. V2I can also refer to the communication with an eNB, i.e., with the infrastructure of a MNO.

- Vehicle-to-Network (V2N), when a vehicle communicates with a remote application server hosting a network-based service, for instance through cellular connectivity.

- Vehicle-to-Pedestrian (V2P), when an application requires the exchange of data between vehicles and vulnerable road users, specifically, pedestrians (but the same sub-use case can also apply to cyclists). V2P applications usually aim at improving safety on the roads by making vehicles and pedestrians more aware of each other, and providing services such as *pedestrian collision avoidance*.

Alongside these communication types, it is worth mentioning that a Vehicle-to-Infrastructure-to-Vehicle (V2I2V) communication is also possible, when the infrastructure acts as relay between different vehicles, which represent the final recipients of the exchanges messages. This kind of communication requires the presence of

the infrastructure, but provides advantages as it enables the transmission of information over much wider ranges and in harsh NLOS conditions.

2.1.1 V2X Use Cases

Vehicular connectivity pushes towards safer and smarter vehicles. Several use cases are indeed made possible by making road users (vehicles, but also pedestrians, trucks and motorcycles) exchange data between each other and towards a supporting infrastructure.

The most common and interesting use cases, on which research is mostly focusing, are listed below:

- **Virtual Traffic Lights (VTL)**, in which vehicles exchange information such as position (thanks to accurate, lane-level, GNSS receivers) and speed and use them to autonomously coordinate at intersections, increasing safety, decreasing the waiting time at intersections and reducing the costs needed for physical traffic lights [38]. VTL is enabled by V2V communication, and, with lower automation levels, a Human-Machine Interface (HMI) can display the traffic light status to the driver.
- **Green Light Optimized Speed Advisory (GLOSA)**, in which vehicles exchange dynamic information through periodic messages, and a service (which can be both decentralized and centralized) provides the driver and the vehicle with information about green phases of traffic lights at intersections. This information can be used to optimally control the vehicle speed and reduce as much as possible the waiting time during a red phase (an optimal speed can be provided to the driver, such that it can maximize the *probability of green*), together with the stop-and-go maneuvers. Among several benefits for the driver, this use case can help to reduce the overall pollutant emissions [16].
- **Platooning**, in which automated vehicles (including heavy vehicles) can form a *string* and travel all together, like a *train of vehicles*. Thanks to the continuous transmission and reception of data between the platoon members, the vehicles can move all together, manage the entrance and exit of vehicles from the platoon and maintain a small inter-vehicle distance. This provides advantages in terms of increased road safety, better fuel economy thanks to the *slipstream* effect and improved road utilization. This use case could potentially be enabled by exploiting the information from radars only. However, the range of this kind of sensors is limited to the preceding platoon member. Vehicular networks are thus a fundamental enabler for platooning, as they enable each vehicle to be aware of all the other platoon members. As highlighted in [39], one of the key challenges of this use case is defining a proper (possibly time-varying) inter-vehicle distance, as a control objective. Platooning with trucks has been recently demonstrated by Volvo and FedEx [40].

- **Non-Line-of-Sight**, also commonly referred to as ***See Through***, in which vehicles exchange information about perceived objects (for instance through cameras, radars or LiDARs). This allows them to be aware of obstacles, pedestrians and other road users even when they are in NLOS, for instance due to blind spots, or due to a preceding heavy vehicle which impedes the view of the road. A decentralized See Through can be based both on V2V and V2I2V communication for an extended awareness horizon. See Through can aid vehicle passing, noticing pedestrians in blind spots, and noticing upcoming detours [41].

- **High-quality video streaming and task offloading**. As autonomous vehicles are slowly shifting towards L4 automation, entertainment and video streaming use cases are increasingly gaining importance. The real-time data transmission between vehicles can enable a plethora of infotainment use cases (including interactive entertainment for passengers and multimedia data exchange), together with ML-based task offloading. The latter is gaining more and more attention from the research community as smart vehicles will require the execution of complex Deep Learning (DL) tasks (such as *object detection* starting from a camera input), for navigation purposes, but also to further improve the on-board automation level. As these tasks are computationally expensive, and strict latency requirements must be met, vehicles can offload them to the infrastructure or to other vehicles which provide free resources, and then receive a lightweight version of the results, containing only the relevant information. Task offloading is made possible thanks to V2V and V2I communication, and, as it requires ultra-low latency and very high throughput (for instance, to transmit the raw camera frames for *object detection* offloading), it faces several challenges. These include, for instance, the choice of a proper technology, or combination of them, to guarantee at the same time high throughput, high reliability and low latency. As technologies such as Cellular-V2X Mode 4 or IEEE 802.11p can hardly provide a throughput higher than 30 Mbit/s [11], the usage of 5G, Terahertz communication [18] and mmWave [42] is currently being investigated.

- **Intersection Collision Risk Warning (ICRW)**. As discussed in the previous Section, safety use cases are among the most promising for vehicular networks, and they promise to progressively reduce the number of fatalities on the roads, which remains unacceptably high [43], [44]. Among these use cases, ETSI has defined ICRW [9], in which vehicles transmit periodic messages with their kinematic data, which are received by a dedicated road-side entity. This RSU analyzes the content of the messages and generates in turn warning messages as collision alerts, if an hazardous situation arises. The collision risk assessment can also be aided by the reception of other kind of messages, coming from other services, such as packets containing road signage information

at intersections. Thanks to this feature, ICRW can detect not only possible collisions, but also traffic sign violations. ETSI also specifies several requirements for this service to be effective, such as a maximum allowed processing time of 80 ms.

- **Collision Avoidance**. The concept of ICRW can be further generalized for a set of use cases aimed at detecting possible hazardous situations in any part of the road, and involving both vehicles and Vulnerable Road Users. This use case is realized by means of algorithms that rely, in turn, on vehicles exchanging data such as position, speed, heading and acceleration. These algorithms are able to detect potential hazardous situations, which may cause potential collisions, and act accordingly. The action can span from the generation of standard-compliant warning messages for the involved drivers, to the trigger of an autonomous emergency brake maneuver. The penetration rate, as briefly introduced at the beginning of Chapter 1, plays a fundamental role, as proved in [15]. This use case, which has been widely studied in simulated and emulated scenarios, is thus expected to be more effective in real-world scenarios when the number of circulating connected vehicles will start to reach at least 50%.
- **Centralized automated maneuvers**. In pure decentralized use cases vehicles can only rely on a limited view of the road. Both the industry and research worlds are thus investigating centralized automated maneuver management services, which can leverage a much wider view of the road thanks to a centralized Local Dynamic Map [26]. With the emergence of 5G and low-latency V2I communication, vehicles in a wide area can provide a MEC service with up-to-date information about their position, speed and acceleration. These services, upon request, can then optimally compute the vehicle trajectories to enable maneuvers such as SAE L4 automated lane merge, and provide the vehicles with detailed information on how to perform the needed actions. Finally, the involved cars can use this information to autonomously and safely perform the maneuver.

2.2 DSRC-based protocols

The main aim of this Section is to introduce the reader to a set of *Wi-Fi*-based standards specifically targeted at vehicular communications, and currently representing one of the most studied and deployed technologies for V2X.

These standards include *(i)* IEEE 802.11p, defining the Physical (PHY) and lower Medium Access Control (MAC) layers of the communication stack, and *(ii)* the IEEE 1609.x set of standards, defining the higher layers together with SAE, which in turn defines a set of standardized messages for the American market. Specifically, IEEE 802.11p is based on IEEE 802.11a and IEEE 802.11e (i.e., standard 5 GHz *Wi-Fi*).

While IEEE dominates the scene in North America, ETSI defines the ITS-G5 standard for Europe. Despite being mostly Europe-centric, this standard can actually be adopted worldwide, at least concerning the standardized messages types detailed in the next Sections. Indeed, it is flexible enough to be adapted to several access technologies, including both IEEE 802.11p and Cellular-V2X. Therefore, most of the work in this thesis is based on ETSI ITS-G5.

Finally, it is worth mentioning how a radically different standard has been defined in Japan, under the name *ARIB T109*. This standard uses a much lower frequency band (around 700 MHz) and a special medium access control method, partially based on Time Division Multiple Access (TDMA) [45]. As this standard is rarely used in Europe and in North America, its analysis falls outside the scope of this thesis.

2.2.1 IEEE 802.11p

IEEE 802.11p is a particular amendment to the IEEE 802.11 standard, released in 2010, six years after the related work groups at IEEE were formed. This amendment is now part of the 802.11-2020 standard [46], and defines the PHY and lower MAC layers of the ISO/OSI communication stack for V2X communications.

IEEE 802.11p is the most used DSRC (Dedicated Short-Range Communications) standard for V2X, and it is based on standard 5 GHz *Wi-Fi* (i.e., IEEE 802.11a, plus some features related to traffic prioritization adapted from IEEE 802.11e). Adopting the terminology from IEEE, this amendment represents the new lower layers for the so-called *WAVE (Wireless Access in Vehicular Environments)* stack.

As opposed to the other amendments, this standard employs a dedicated 75 MHz spectrum assigned by the Federal Communications Commission (FCC) in 1997 for vehicular communication. This frequency band ranges from 5.850 to 5.925 GHz and accommodates seven 10 MHz channel, plus a 5 MHz guard band between 5.850 and 5.855 GHz. This band is foreseen to avoid interference with other technologies due, for instance, to the Doppler effect.

IEEE also defines a standard channel numbering, based on the following formula:

$$f(CH) = 5000 + 5 \cdot CH[MHz] \tag{2.1}$$

Where CH is the channel number. This leads to the definition of the seven channels as depicted in Figure 2.3.

IEEE further divides these channels into one Control Channel (CCH, channel 178), as the default channel for high priority and control information, and six Service Channels (SCH).

It is also worth mentioning how ETSI foresees a slightly different channel assignment in the DSRC band. Indeed, five channels were standardized in 2010 for Europe, with four service channels from 172 to 178 and one control channel at 5.90 GHz (channel 180) [47]. All the other channels (182, 184) are reserved for

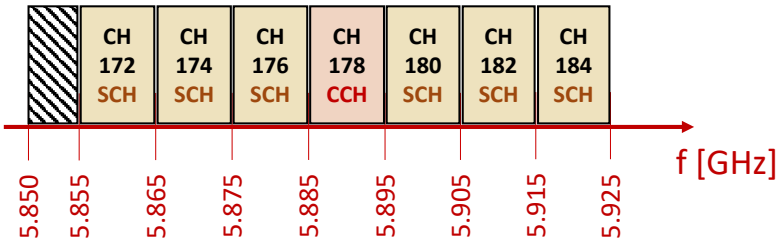


Figure 2.3: 10 MHz DSRC channels, as defined by IEEE [46].

future use. Consequently, the de-facto default channel for high priority and safety-critical V2X communication over IEEE 802.11p is channel 178 in North America and channel 180 in Europe.

As opposed to IEEE 802.11a, IEEE 802.11p foresees 10 MHz channels, corresponding to half the bandwidth of the "a" amendment (i.e., 20 MHz). As a consequence, all the timing parameters are doubled, and all the reachable data rates are halved. Indeed, IEEE 802.11p employs the Orthogonal Frequency-Division Multiplexing (OFDM) which includes 64 orthogonal subcarriers and supports data rates from 3 Mbit/s, with the simplest Binary Phase-Shift Keying (BPSK) modulation, up to 27 Mbit/s, with 64-QAM (Quadrature Amplitude Modulation). Instead, IEEE 802.11a data rates range from 6 Mbit/s to 54 Mbit/s. Furthermore, the OFDM symbol duration is now equal to $8\mu s$, instead of $4\mu s$. This approach has been adopted because, despite the throughput reduction, it can efficiently counteract Doppler, fading and multi-path effects, together with inter-symbol and inter-carrier interference, thanks to a much longer (2 times) Guard Interval. These effects may be negligible in static local networks based on IEEE 802.11a. However, their impact may start to be significant in vehicular networks, as highly mobile nodes are involved.

The IEEE 802.11-2020 standard also defines the available data rates and corresponding modulations. These data rates are schematized in Table 2.1. According to the standard, three of these data rates should be compulsorily implemented, i.e., 3 Mbit/s, 6 Mbit/s and 12 Mbit/s [46].

Concerning instead the MAC layer, its implementation is mostly based on IEEE 802.11a, with a few QoS enhancements from IEEE 802.11e. Among its features, two main characteristic are peculiar to IEEE 802.11p:

- The operation outside any defined Basic Service Set (BSS), with the so-called Outside the Context of a BSS (OCB) mode.
- The usage of the so-called Enhanced Distributed Channel Access (EDCA), in place of the standard IEEE 802.11 CSMA/CA DCF (Carrier Sense Multiple Access with Collision Avoidance Distributed Coordination Function) for

Modulation	Coding rate	Data bits per OFDM symbol	Data rate
BPSK	**1/2**	**24**	**3 Mbit/s**
BPSK	3/4	36	4.5 Mbit/s
QPSK	**1/2**	**48**	**6 Mbit/s**
QPSK	3/4	72	9 Mbit/s
16-QAM	**1/2**	**96**	**12 Mbit/s**
16-QAM	3/4	144	18 Mbit/s
64-QAM	3/4	192	24 Mbit/s
64-QAM	3/4	216	27 Mbit/s

Table 2.1: IEEE 802.11p data rates with 10 MHz channels. The *coding rate* represents the proportion between the useful data bits (i.e., non-redundant information) and the total number of coded bits. The mandatory data rates are highlighted in bold.

channel access. This channel access method foresees, instead of the DCF Interframe Space (DIFS) [46], four Arbitration Interframe Spaces (AIFS), whose lengths depend on the priority of the data traffic. Indeed, in case of contention, shorter AIFS values correspond, probabilistically, to lower waiting times for high priority traffic, while longer AIFS values are suitable for low priority traffic.

As mentioned earlier, vehicular networks are characterized by tight latency requirements. Thus, to reduce as much as possible the time that would be needed to establish a connection, IEEE 802.11p does not foresee any authentication and association (to a BSS) procedure, as it would happen in a standard client-access point Wi-Fi network. Being able to communicate with any other device without belonging to any BSS is called OCB mode[1]. Normally, each station compliant with IEEE 802.11 should encode, in the MAC header, a BSSID 48-bit value, corresponding to the BSS the device is currently belonging to. As in OCB mode there is no BSS involved, each DSRC-compliant device should set a wildcard BSSID (i.e., `0xFFFFFFFFFFFF`) [46].

Concerning instead EDCA, the prioritization of data is made possible thanks to four Access Categories (AC), or *traffic classes*, namely:

[1]IEEE also defines Tunneled Direct Link Setup (TDLS), as a way to make devices directly communicate between each other [48]. IEEE 802.11p, however, requires the usage of OCB mode, as it enables a seamless communication between devices without the necessity of any Access Point or Group Owner.

- AC_BK (Background): lowest priority
- AC_BE (Best Effort)
- AC_VI (Video)
- AC_VO (Voice): higher priority

The EDCA Access Categories are derived starting from eight User Priorities (UP), defined in IEEE 802.11D [46] and associated to the data which should be transmitted. In particular, UPs 1 and 2 correspond to AC_BK, 0 and 3 to AC_BE, 4 and 5 to AC_VI, 6 and 7 to AC_VO. When setting an AC on a patched Linux kernel, as explained in Chapter 3, the corresponding UP should be specified, since it is then automatically mapped to the proper class.

Each AC corresponds to a specific transmit queue inside the MAC layer. Each of these queues, before accessing the physical channel, will have to virtually contend the channel access, as it happens with DCF. However, different AIFS values are defined for each AC, instead of waiting for a DIFS time. The shorter the AIFS, the more likely a queue will be able to access the channel before the other queues. Thus, shorter AIFS are associated to high priority ACs, while longer AIFS are associated to low priority ACs. The contention window sizes are also different for each AC, such that a high priority queue likely transmits before a low priority one, but without monopolizing the channel thanks to the randomness introduced by the backoff procedure. After the virtual contention procedure, the "winning" queue will be able to access the physical channel.

The AIFS and contention window sizes for each AC are resumed in Table 2.2, and are computed starting from [46]. In particular, the AIFS for each AC is computed starting from an AIFS number (AIFSN), with the following formula:

$$AIFS(AC) = AIFSN(AC) \cdot aSlotTime + aSIFSTime. \tag{2.2}$$

According to Table 17-21 of IEEE 802.11-2020 [46] $aSlotTime$ is 13 μs for 10MHz channels, while the SIFS time is instead $aSIFSTime = 32\,\mu s$.

Access Category	CW_{min}	CW_{max}	AIFSN	AIFS
AC_BK	15	1023	9	149 μs
AC_BE	15	1023	6	110 μs
AC_VI	7	15	3	71 μs
AC_VO	3	7	2	58 μs

Table 2.2: AIFS and contention window sizes for each AC.

Finally, it is worth mentioning how IEEE 802.11p foresees a special EDCA mode, called *Alternate EDCA*, which extends the four queues and define a total of six

queues. In particular, the Video and Voice queues are divided into a *primary* and *alternate* queue, enabling a further prioritization of traffic.

2.2.2 IEEE 802.11bd

As wireless technologies evolve, several new Wi-Fi standards have been defined by IEEE, including IEEE 802.11ac, IEEE 802.11ax and IEEE 802.11ay. However, IEEE 802.11p, despite being a well-established technology, is still based on older amendments such as IEEE 802.11a and IEEE 802.11e.

IEEE is thus making efforts in defining a new standard for vehicular communication, which will represent an evolution of IEEE 802.11p and which is currently being standardized as IEEE 802.11bd. Indeed, the corresponding Task Group has been created in January 2019.

The main design objectives of IEEE 802.11bd can be summarized as follows [49]:

- Improve the available throughput, providing at least a mode achieving twice the data rates of IEEE 802.11p at relative speeds up to 500 km/h;
- Improve the reachable range, providing at least a mode achieving twice the range of IEEE 802.11p;
- It should be interoperable with IEEE 802.11p, at least in one communication mode (i.e., IEEE 802.11p devices must be able to decode transmissions from IEEE 802.11bd, and an IEEE 802.11bd device must be backward compatible and be able to decode IEEE 802.11p packets);
- It should coexist with IEEE 802.11p, deferring communication in case of concurrent communications;
- Equal and fair channel access should be guaranteed for the coexistence between IEEE 802.11p and IEEE 802.11bd;
- It should consider advanced PHY layer mechanisms, such as Dual Carrier Modulation (DCM), introduced in IEEE 802.11ax;
- Starting from the IEEE 802.11ac PHY layer, it introduces *midambles*, acting like preambles, but being located at the middle of each frame.

Finally, it is worth mentioning how the standardization efforts for IEEE 802.11bd are considering the usage of mmWave frequencies, for the use cases requiring very high throughput over small ranges. The usage of mmWave in IEEE 802.11bd could extend existing IEEE mmWave protocols, such as IEEE 802.11ad, working in the 60 GHz frequency range, or the new IEEE 802.11ay amendment.

2.2.3 IEEE WAVE

Building on IEEE 802.11p, for the PHY and lower MAC layers, IEEE standardized the higher layers of the ISO/OSI communication stack as part of the IEEE 1609.x family of standards.

These standards, combined together (with the addition of IEEE 802.2 for the Logical Link Control layer), form the so-called WAVE protocol stack, which is specifically targeted at the US market and defines several standards optimized for vehicular communications.

The scope of the main 1609.x standards is schematized in Figure 2.4.

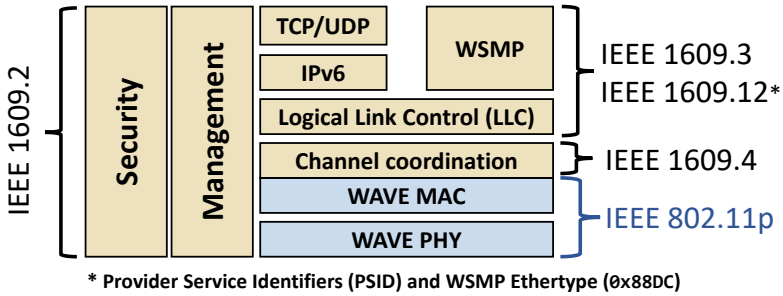


Figure 2.4: The IEEE WAVE family of standards [50]–[54]

The main IEEE 1609.x standards can be summarized as follows:

- **IEEE 1609.0 - IEEE Guide for Wireless Access in Vehicular Environments (WAVE) Architecture**: this document contains the definition of a WAVE system (i.e., a *"radio communication system intended to provide seamless, interoperable services to users of a transportation system"* [50]), together with the structure of the IEEE 1609.x standards, the architecture and the operations of a WAVE-compliant system.

- **IEEE 1609.2 - IEEE Standard for Wireless Access in Vehicular Environments - Security Services for Applications and Management Messages**: this document describes all the security aspects of IEEE WAVE. Security represents one of the most important challenges in vehicular networks, and it is thus crucial to provide standardized security procedures and certificate management. IEEE WAVE security is built upon five cardinal principles: (i) authenticity (authentication of the sender identity), (ii) authorization (making sure that the sender has the requested privileges), (iii) integrity (making sure that the data has not been altered during the transmission), (iv) non-repudiation (the ability of verifying authenticity, authorization and integrity to a third party), (v) confidentiality (only the intended recipient should be able to leverage the content of the message). If security is enabled, a Secure Data Service (SDS) is in charge of translating insecure packets into Secured

Protocol Data Units (SPDU) before the message transmission, and performing the opposite operation on the reception side.

- **IEEE 1609.3 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services**: this standard defines the WAVE Networking Services starting from the Network layer, and defines the usage of either TCP/UDP over IP version 6 or a dedicated protocol called WAVE Short Message Protocol (WSMP) [52]. WSMP is a lightweight protocol for safety-related vehicular applications, which perform addressing by means of Provider Service Identifiers (PSIDs). A PSID is a unique identifier for the source and target services, and it is leverage to provide the packet content to the proper application on the receiving side. PSID are better detailed in IEEE 1609.12 [53]. IEEE 1609.3 also describes the usage in WAVE of the LLC layer, which is defined in IEEE 802.2.

- **IEEE 1609.4 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-channel Operation**: this standard defines the multi-channel operations in WAVE, including channel coordination, channel routing and time synchronization. Multi-channel operations are defined as part of an "upper" MAC layer, as an extension to the IEEE 802.11p MAC layer. In particular, WAVE mandates the CCH to be used for the so-called WAVE Service Advertisements (WSA) and for management data, leaving all the other application-related messages to the Service Channels. This standard also defines three channel switching modes, to efficiently switch between the CCH and SCH, with a time granularity of 50 ms and dependant on the usage scenario [51]. As time synchronization assumes a pivotal role in channel switching, the standard mandates every WAVE-compliant device to comply with a common time reference, i.e., UTC (Coordinated Universal Time) time modulo 1 second.

- **IEEE 1609.12 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Identifier Allocations**: this standard defines how PSIDs should be used and encoded, and provides references for other miscellaneous identifier, such as the Ethertype for WSMP (i.e., `0x88DC`).

Finally, as can be seen, the Application, Presentation and Session layers from the ISO/OSI model are not explicitly defined. Indeed, these layers are actually defined by SAE, which collaborated with IEEE in the definition of the WAVE stack. The J2735 [55] and J2945 [56] standards focus on safety applications and define a set of standardized vehicular messages, together with their usage, similarly to the definition of the so-called Facilities Layer in ETSI ITS-G5 (as described in Section 2.2.4).

In particular, SAE J2735 defines several message types, spanning from safety-critical periodic messages to more application-related ones. Among these, it is

worth mentioning [55]:

- **Basic Safety Message (BSM)**: a periodic message, transmitted with a typical 10 Hz frequency, containing vehicle kinematic data, such as position, speed, acceleration, heading, brake system status and vehicle size.
- **Common Safety Request (CSR)**: a unicast message sent by a vehicle participating in the exchange of BSMs, and used to request additional information from other vehicles. This additional information is requested when required by the on-board safety applications.
- **Emergency Vehicle Alert (EVA)**: a message broadcasted as a warning message for all the vehicles in the vicinity of an emergency vehicle. As they receive the EVA, they are required to let the emergency vehicle pass and exercise additional caution.
- **Intersection Collision Avoidance (ICA)**: event triggered message, broadcasted as a warning when an hazardous situation arises in proximity of an intersection, increasing the risk for potential collisions. This message can be transmitted by a Collision Avoidance service residing either in another vehicle (decentralized approach) or in the infrastructure (centralized approach).
- **Road Side Alert (RSA)**: event triggered message, broadcasted by the infrastructure to alert travelers of nearby hazards. These hazards may include ice on the road, an approaching train at a level crossing or a construction zone [55].

2.2.4 ETSI ITS-G5

In parallel to IEEE, ETSI defined the Intelligent Transport Systems (ITS)-G5 set of standards (in short, ITS-G5). These standards define the upper layers of the ISO/OSI stack for the European market, from the Application to the upper MAC layer, together with the management and security aspects.

The lower layers (i.e., PHY and lower MAC) are defined by IEEE 802.11p. However, as mentioned earlier, ETSI ITS-G5 is flexible enough to be used in conjunction with other access technologies, such as Cellular-V2X [57]. The latter supports an infrastructure-less communication mode, called Mode 4 [58], which can be used as an alternative, or in conjunction with, IEEE 802.11p.

The connected vehicle network architecture foreseen by ETSI is depicted in Figure 2.5.

As can be seen, alongside the Decentralized Congestion Control (DCC), which is introduced more in detail later in this Section, ETSI defines the so-called ITS-G5 Transport and Networking layers. The usage of these dedicated standards

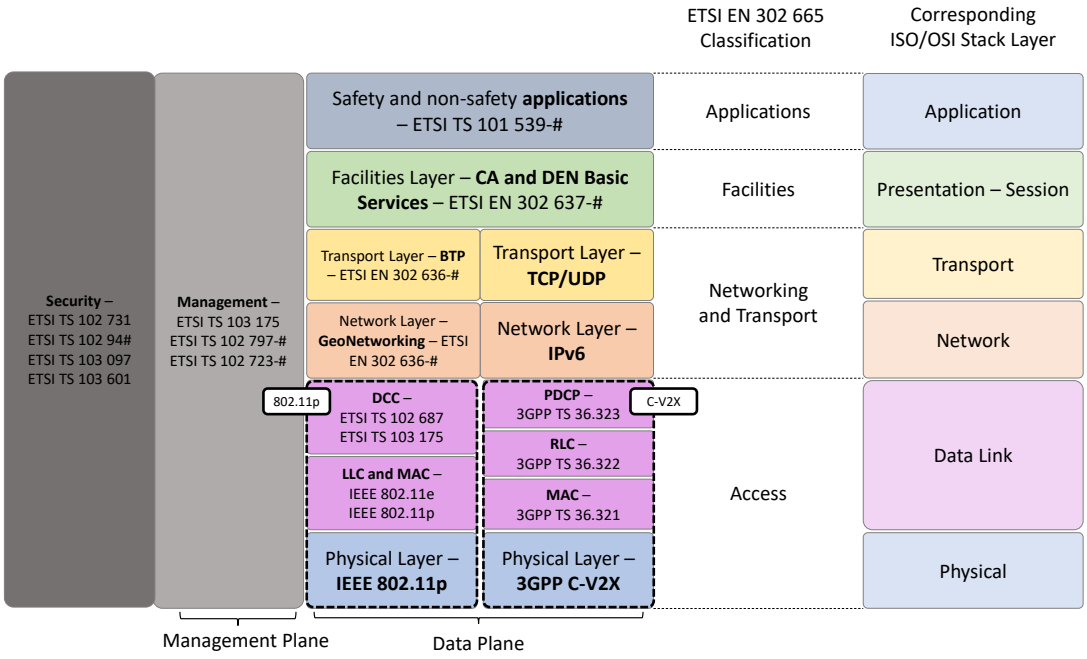


Figure 2.5: Standardized network stack for a connected vehicle, as foreseen by ETSI. Each layer corresponds to a given ISO/OSI layer, and all the relevant standards (by either ETSI, IEEE or 3GPP) are reported below each protocol.

is recommended for optimized V2X communications. However, other IP-based protocols are foreseen too, such as TCP/UDP over IP version 6.

The Transport and Networking layers include, respectively, Basic Transport Protocol (BTP) [59] and GeoNetworking [21]. The first is a lightweight connectionless transport protocol, comparable to UDP and containing port numbers which are used to determine the destination service (e.g., the one managing periodic messages, or the one managing event-based ones) when ITS messages are received by vehicles. According to ETSI, two types of BTP headers are foreseen: either BTP-A, for interactive sessions, in which the recipient is expected to provide a reply, or BTP-B, which is used for non-interactive message dissemination. BTP-A contains, in the header, a source port and a destination port, while BTP-B, instead, contains only a destination port and a *destination port info* field, which is normally unused and set to 0. The standard also defines a BTP port number for each vehicular message type [60].

GeoNetworking is instead a network layer protocol which defines several ways of performing geographical routing of messages, and describes different types of addressing schemes. Three main ways of performing geographical routing are foreseen [21]:

1. **GeoUnicast**. This mode is used for the transmission of unicast messages based on the destination location and on the so-called GeoNetworking address (*GN_ADDR*). The latter is a 64-bit address which combines the device MAC address with the ITS Station type (e.g., a passenger car has station type "5", while an RSU has its station type equal to "15"). Messages are forwarded by intermediate hops, if needed, until they reach the destination.

2. **GeoBroadcast (GBC)**. This mode is used for disseminating broadcast messages in a given area, specified as part of the GeoNetworking headers. If needed, messages can be forwarded until they reach the destination area.

3. **Topologically-Scoped Broadcast (TSB)**. An ETSI-compliant ITS Station relies on this mode when a message should be broadcasted to all the n nearest hops. A special case of TSB is called Single Hop Topologically-Scoped Broadcast (SHB), and it is leveraged to disseminate a message only to the nearest first hop. SHB does not require any forwarding, as opposed to TSB with $n > 1$.

As can be seen in Figure 2.5, above the Networking and Transport layers, ETSI focuses on the so-called Facilities Layer. Its main aim is to provide support to ITS Applications and Vehicular Networking services working at the Application Layer, including several crucial features to manage the transmission and reception of standardized vehicular messages [61]. An application is provided with ways to request the transmission of messages, and receives the decoded data from received packets from the Facilities Layer.

As this layer plays a pivotal role in the ETSI stack, it is further divided into three sub-layers [62]:

1. Application Support, which includes all the services providing message encoding/decoding functions and support to the overlying ITS applications.

2. Information Support, which includes all the services providing support for data management and collection. The ETSI Local Dynamic Map (LDM) [26] is part of this sub-layer.

3. Communication Support, which provides services for communication and session management [62]. For instance, this sub-layer manages, if needed, sessions between different ITS stations, or it implements, through the *Web service support* facility, protocols such as HTTP to enable the connectivity between the vehicle and Internet.

As for SAE J2735, ETSI ITS-G5 defines several standardized message types. The most important ones are described below:

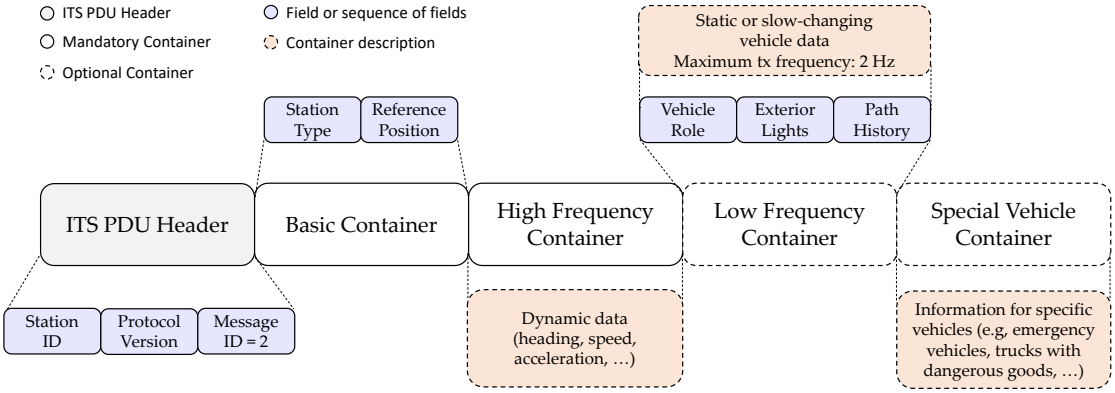


Figure 2.6: Structure of a CAM message, with the ITS PDU Header and all the mandatory and optional data containers [63]. For some containers, the included fields or sequences of fields (e.g., Path History is a sequence of latitude and longitude variations relative to the current vehicle position) are indicated, while for others only a short description is provided for readability reasons.

- **Cooperative Awareness Message (CAM)**: periodic messages transmitted with a frequency between 1 Hz and 10 Hz and storing kinematic and dynamic data such as vehicle position, heading, speed, acceleration, alongside additional information such as steering wheel angle and turn indicator status [63]. CAMs are transmitted via GeoNetworking with SHB routing and include both mandatory and optional data containers, as depicted in Figure 2.6. They correspond to the BSM WAVE messages and their frequency is managed dynamically depending on the last position variation, speed variation and heading variation. The default frequency is set to 1 Hz, but if any of these variations exceeds a given threshold (set respectively to 4 m, 0.5 m/s and 4°), it is increased up to 10 Hz [63]. The transmission and reception of CAMs is managed by the so-called Cooperative Awareness Basic Service (CA Basic Service), in the Facilities Layer.

- **Decentralized Environmental Notification Messages (DENM)**: event-based messages broadcasted to notify vehicles and infrastructure nodes about hazards and other events on the road [64]. DENMs are transmitted via GeoNetworking with GBC routing over a destination area. They partially correspond to the RSA WAVE messages. The fields of CAMs and DENMs are defined in the so-called ETSI Common Data Dictionary [65]. The transmission and reception of DENMs is managed by the so-called Decentralized Environmental Notification Basic Service (DEN Basic Service), in the Facilities Layer.

- **Infrastructure to Vehicle Information Message (IVIM)**: these messages are used in V2I communications to carry road signage information [66]. Each

road sign is encoded as a *category code*, defined by International Organization for Standardization (ISO) 14823 [67].

- **Radio Technical Commission for Maritime Services Extended Message (RTCMEM)**: this message has been defined to encapsulate RTCM (Radio Technical Commission for Maritime Services) data. RTCM is a proprietary protocol defining messages and techniques for supporting GNSS operations with one reference station, or a network of reference stations. RTCM can be used, for instance, to exchange raw GNSS data between different entities. RTCMEMs are thus foreseen as part of a GNSS Positioning Correction (GPC) service for improving vehicle localization thanks to correction data sent from RSUs to vehicles [66].
- **Services Announcement Essential Message (SAEM)**: even though less commonly used, this message acts as service announcement and provides information about available ITS services.
- **MAP (topology) Extended Message (MAPEM) and Signal Phase And Timing Extended Message (SPATEM)**: these messages carry information, respectively, about the current road topology (e.g., lane topology, dedicated lanes, allowed maneuvers) and the phase and timing of traffic lights at intersections to enable use cases such as GLOSA [66].
- **Vulnerable Road Users Awareness Message (VAM)**: this type of message has been recently standardized as part of a VRU awareness service. These messages work similarly to CAMs, but they are transmitted by VRUs (such as pedestrians), and they enable safety related use cases such as pedestrian collision avoidance. These messages are transmitted with a variable frequency between 0.2 Hz and 10 Hz, and they carry information such as position on the road, speed, exterior lights for cyclists and pedestrian profile (e.g., normal pedestrian, road worker, first responder) [68].
- **Cooperative Perception Message (CPM)**: this message, at the time of writing, is currently being standardized by ETSI, and it is expected to be released soon as part of a Cooperative Perception Service (CPS) [69]. These message enables collective perception and contain information about objects detected through on-board sensors such as cameras and radars. This data is shared in order to provide enhanced awareness about both connected and non-connected objects. These messages are expected to contain a list of detected objects, their position (and possibly classification), and information on the sensory capabilities of the sending ITS station (i.e., a vehicle with its sensors).

Each ITS message is composed of an ITS PDU (Protocol Data Unit) header, followed by the actual message content. The ITS PDU header is composed of a

mandatory *station ID*, as an identifier of the transmitting ITS entity (i.e., a vehicle, a pedestrian, an RSU, ...), the *protocol version*, which identifies the version of the message and protocol (for instance, two versions of CAMs are currently defined), and the *message ID*, which identifies the type of ITS message. The message IDs and BTP port numbers for the messages listed above are resumed in Table 2.3.

Message	BTP port number	Message ID
CAM	2001	2
DENM	2002	1
IVIM	2006	6
RTCMEM	2013	13
MAPEM	2003	5
SPATEM	2004	4
VAM	2018	14

Table 2.3: BTP port numbers and message IDs for the most common ETSI ITS-G5 messages.

ASN.1

All the messages standardized as part of the ETSI Facilities Layer are defined by means of the Abstract Syntax Notation One (ASN.1) interface description language. This language enables the definition of complex data structures in a platform-agnostic way, including the content of packets to be transmitted over a network.

Importantly, ASN.1 files can be used to automatically generate the source code of encoding and decoding functions, thanks to open source tools like *asn1c* [70].

After defining the content of a message with ASN.1, several encoding rules are made available to the user, which define how the message will be encoded and provide different advantages and disadvantages in terms of processing speed and message compactness (which in turn translates into a reduced load on the wireless channel) [71]. The definition of an ASN.1 message thus always comes with an agreement on a proper encoding option. Therefore, all ETSI standard-compliant messages, such as CAMs, make use of UPER (Unaligned Packed Encoding Rules), guaranteeing small message sizes [63].

Decentralized Congestion Control

One of the main functionalities of ETSI ITS-G5 is the so-called cross-layer Decentralized Congestion Control (DCC) [72], [73]. The idea behind DCC is to control several PHY and MAC parameters depending on the channel conditions, such as

transmission power and minimum packet interval, with the aim of reducing the channel load and guaranteeing a stable communication in case of high vehicle density.

DCC works by defining a State Machine for the so-called Channel Busy Ratio (CBR, i.e., how often the channel is sensed busy by a node over a given time span). A channel is considered busy when the measured received power (either signal or noise) is above a given carrier sense threshold.

In the most simple configuration, a vehicle is then classified into three states: relaxed, active or restricted. The transition between these states is regulated by the minimum and maximum channel load measured over a given time interval, as depicted in Figure 2.7.

The leftmost state represents a situation in which a vehicle can be more "aggressive" on the channel (i.e., use a higher transmission power or transmit messages more often), as a low CBR was detected (corresponding to a low connected vehicle density condition). The rightmost state, instead, correspond to a high load condition, in which limits should be posed to the maximum transmission power or message frequency.

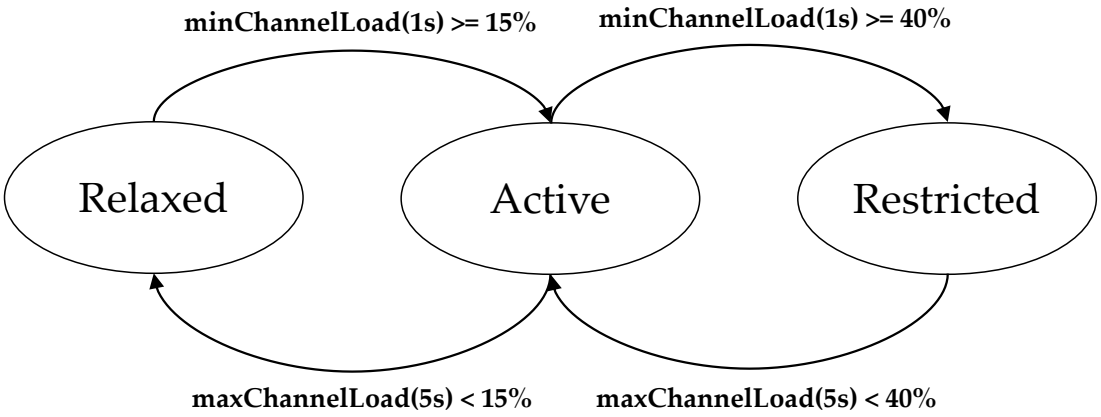


Figure 2.7: The state machine for Decentralized Congestion Control on the Control Channel with one "active" state. Notice how transitioning back from a restricted or active state takes more time than transitioning to the same state.

Each state is characterized by four main configurable parameters for each AC, corresponding to four control mechanisms, i.e., transmit power, minimum inter-packet time, transmit data rate, sensitivity of Clear Channel Assessment (CCA). The states are then configured in such a way that the load on the channel is reduced in case of high vehicle density (i.e., in case of high CBR, which can progressively lead to a restricted state). For instance, in a relaxed state a vehicle is allowed to use a transmission power up to 33 dBm for the CCH, while in an active state the maximum transmission power for an AC_BE traffic class is reduced to 20 dBm.

With the aim of providing a more fine-grained control and making the system more reactive, more than one "active" state can be defined. This is, for instance, the case of DCC for Service Channels in [72].

A new version of the DCC technical specifications, released by ETSI in 2018, is indeed proposing the parameter setting for a 3-active states configuration, together with a novel adaptive approach for congestion control [74].

2.3 Cellular-based protocols (C-V2X)

In recent years the application of cellular networks to both V2V and V2I communications has emerged as a very promising technology, which could represent an effective alternative to Wi-Fi-based protocols, possibly overcoming their limitations when the vehicle density increases.

The application of cellular networking to V2X appears to be promising under several aspects, including more resilience to channel congestion, the possibility to use both the so-called *Uu* interface [75] for communication with the network and the *PC5* one for direct V2V data exchange [76], [77], and possibly an improved radio range over IEEE 802.11p.

The first step towards C-V2X was defined by 3GPP in their Release 12 standard, where the Device-to-device (D2D) communication was first introduced for the so-called ProSe (Proximity Services), initially introduced for mobile devices in public safety scenarios. The first true C-V2X standard was however defined in Release 14, with the definition of LTE-V2X, based on 4G Long Term Evolution (LTE) and D2D communication [78].

This standard was further refined in the subsequent releases, up to Release 16, where New Radio (NR)-V2X has been standardized, based on 5G NR.

As opposed to IEEE WAVE, or ETSI ITS-G5 over IEEE 802.11p, 3GPP focuses on the integration of V2X communication in cellular networks, and details thus only the lower layers of the stack (i.e., PHY and data link layers).

Even though C-V2X appears to be very promising, the debate on which technology is the best is still open, and several works in literature highlight the advantages and disadvantages of both approaches [79]–[81]. In general, IEEE 802.11p, which represents a more mature and well-tested technology, appears to guarantee a lower latency and better performance with aperiodic and variable size messages, while C-V2X can provide increased robustness with respect to distance and channel congestion.

In addition, the automotive market in Europe is still looking at the technological evolution and standardization process to determine which technology could best satisfy the needs of the vehicle manufacturers. Currently, very few vehicles are being released with embedded V2X functionalities in Europe. In this context, while Volkswagen has released the new Golf 8 model with embedded DSRC communication [82], other Original Equipment Manufacturers (OEMs) are investigating the

integration of C-V2X as part of H2020 European Projects.

2.3.1 LTE-V2X

LTE-V2X was first defined in 3GPP Release 14, and foresees two different interfaces:

- The *PC5* interface is used for V2V communication in a one-to-many approach. This interface enables different User Equipment (UEs), which are represented in our case by vehicles, to communicate directly on the 5.9 GHz DSCR band through the *Sidelink* channel.
- The *Uu* interface is used for V2N communication towards a base station (E-UTRAN NodeB, eNB). It enables communication towards the infrastructure, over the standard licensed spectrum.

The V2V communication through *PC5* can happen following two modes: (i) **Mode 3** is used under network coverage, and the infrastructure manages the sidelink D2D communication, including the selection and configuration of the resources; (ii) **Mode 4** is instead exploited out of coverage, and vehicles autonomously select and configure the resources and sub-channels. Mode 4 indeed does not require any SIM card nor MNO subscription.

Mode 3 can provide a superior performance if compared to Mode 4, as resources are centrally managed, but it requires the node to be always under coverage [78], which is often a strong assumption in vehicular networks. This thesis will thus focus on C-V2X Mode 4, which is directly comparable with IEEE 802.11p for V2V and direct V2I (i.e., OBU to RSU) communications.

Concerning the physical layer, LTE-V2X uses Single-Carrier Frequency-Division Multiple Access (SC-FDMA). Two channel bandwidths are supported, 10 MHz and 20 MHz, and, in the frequency domain, each channel is subdivided into 180 kHz-wide Resource Blocks (RBs). Each RB is composed by twelve 15 kHz *subcarriers* [78]. In the time domain, instead, the channel is divided into 1 ms subframes, each composed by two 0.5 ms slots and comprising 14 OFDM symbols, of which four are used to transmit reference signals, which can help to counteract Doppler effect. This time-frequency channel division is schematized in Figure 2.8.

RBs are itself grouped into sub-channels, composed by multiple RB all within the same subframe (the actual number is pre-configured and variable) and used to transmit both data and control information. The transmission of data is organized in the so-called Transport Blocks (TBs), represented by groups of resource blocks able to contain a full packet (e.g., a full CAM or DENM if ETSI ITS-G5 is used for the higher layers). The size of the frame, together with the number of RBs per sub-channel and the selected Modulation and Coding Scheme (MCS), determine the size of the TB, that can span over a single or multiple sub-channels. As an

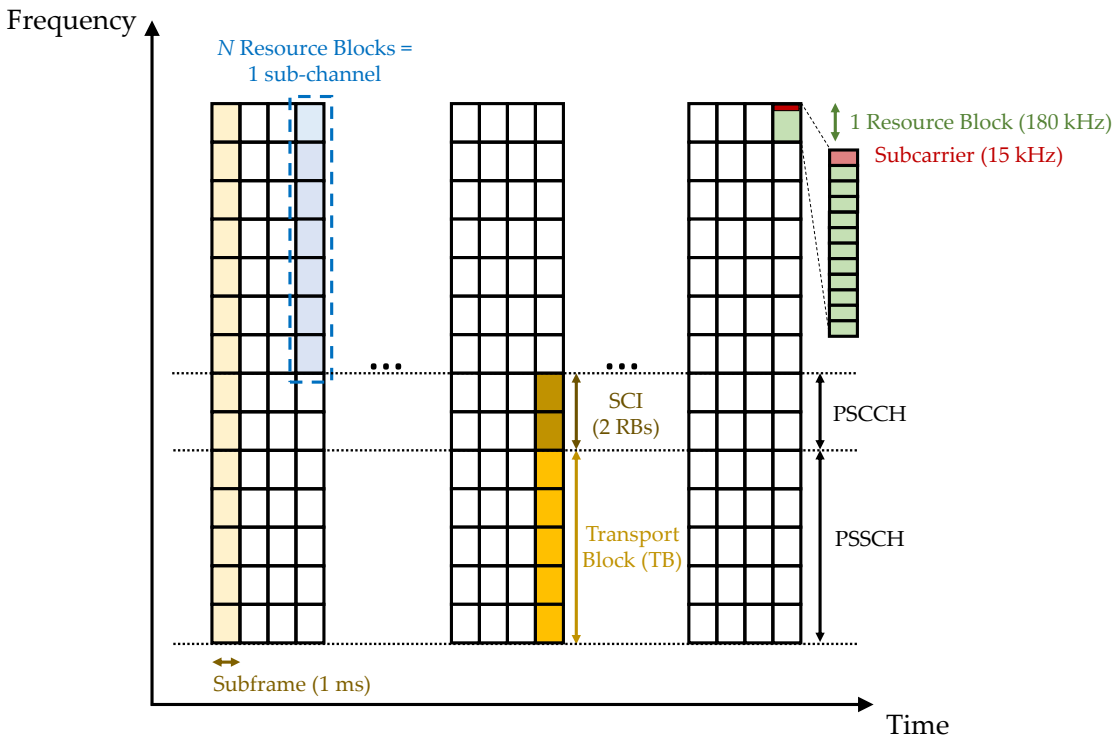


Figure 2.8: Time-frequency division of LTE-V2X channels, with adjacent Physical Sidelink Shared Channel and Physical Sidelink Control Channel.

example, a low MCS will cause a packet to require more RBs than when a high MCS is selected. Resource allocation thus happens, in LTE-V2X, over subframes (of 1 ms) and sub-channels.

Furthermore, each TB is associated with the so-called Sidelink Control Information (SCI), which spans over 2 RBs, and contains control information such as MCS, priority and the selected Resource Reservation Interval (RRI), a pivotal parameter for Mode 4 resource allocation. As the SCI contains crucial information for the decoding of the corresponding TB, it is always transmitted in the same subframe, and it can be transmitted in RBs adjacent to the ones reserved for the TB, as in Figure 2.8, or in non-adjacent ones.

In the frequency domain, two main channels are defined: the Physical Sidelink Shared Channel (PSSCH), used to carry TBs, and the Physical Sidelink Control Channel (PSCCH), which instead is used to carry the SCI control information. As mentioned earlier, these channels can be adjacent or non-adjacent [78].

Concerning the MAC layer, as depicted in Figure 2.5, instead of LLC and MAC as in IEEE 802.11p, three main sub-layers are defined: PDCP (Packet Data Convergence Protocol), RLC (Radio Link Control) and MAC. The respective 3GPP standards are reported in Figure 2.5.

C-V2X Mode 4 Semi-Persistent Scheduling

As LTE-V2X Mode 4 requires the vehicle to independently allocate and manage the needed resources (i.e., proper sub-channels in time), the sensing-based Semi-Persistent Scheduling (SPS) scheme is adopted.

According to the Release 14 3GPP standard, every vehicle should sense the channel for a period of 1000 ms (called *Sensing Window* and equivalent to 1000 subframes), in order to identify the Candidate Single-Subframe Resources (CSRs, i.e., adjacent sub-channels within the same subframe where to transmit the next TB and SCI) which should not be selected for transmission, as they are either sensed or detected (thanks to the reception of SCI from other vehicles) as already occupied.

Among the free resources, the vehicle randomly chooses between the ones with the lowest average RSSI. The selection is then maintained for a number of transmissions, determined by the so-called *Reselection Counter*, which typically assumes values from 5 to 15 [79], even though it depends on the RRI value.

The transmission of the TB and SCI for *Reselection Counter* times is expected to occur at a periodicity defined by the RRI. Indeed, the RRI, transmitted in the SCI, is used to inform other vehicles about the intention of using the same selected resources for the next transmission, which will occur in RRI time. This is done to avoid other nodes selecting the same resources and causing packet collisions [78]. The RRI can assume a value of 20 ms, 50 ms, 100 ms or any multiple of 100 ms up to 1 second. A special value of 0 ms is reserved for the case in which the vehicle will not reserve the same resources for the next TB and SCI transmission.

As the RRI can impact the overall performance of C-V2X, it must be adapted as much as possible to the characteristics of the desired communication. For instance, when transmitting CAMs at 10 Hz, an RRI of 100 ms would probably be the best choice.

When the *Reselection Counter* reaches zero, the vehicle should reselect the resources with a probability $1 - P$, with $P \in [0,0.8]$. As this value is not specified by 3GPP, it can be configured depending on the scenario, and it represents one important parameter when evaluating the performance of C-V2X [78].

Finally, it is worth mentioning how reselections also occur in two additional cases: (i) if a new message is too big for the resources which have already been reserved for the corresponding TB, (ii) if a new message has a latency deadline lower than the current RRI, and must be transmitted before RRI time (e.g., if RRI is equal to 200 ms, and the CAM frequency is increased to 10 Hz due to the ETSI dynamic frequency management [63]).

2.3.2 NR-V2X

With the advent of 5G, starting from Release 16, 3GPP standardized NR-V2X, as an evolution to LTE-V2X based on 5G New Radio. This new communication

standard for V2X extends Releases 14 and 15 with the following main features and peculiar characteristics [83]:

- Mode 3 and Mode 4 are respectively replaced by Mode 1 and Mode 2, being Mode 2 the new infrastructure-less communication mode.
- The numerology of the PHY layer is no more fixed, as in LTE-V2X, and it is now a scalable parameter. The numerology n is directly linked to the subcarrier spacing according to the following formula: $\Delta f = 15\,kHz \cdot 2^n$. Consequently, subcarrier spacing in NR-V2X is no longer limited to 15 kHz, and can be increased up to 30, 60 or 120 kHz. This also determines a different OFDM symbol duration per numerology, and a variable number of RBs in each channel, given a fixed channel bandwidth (e.g., 10 MHz). Therefore, the number of sub-channels in a channel varies with the numerology.
- While LTE-V2X Mode 4, due to the RRI mechanism described above, focuses on periodic transmissions, NR-V2X Mode 2 focuses on supporting aperiodic transmissions too, as part of the new specifications. As a consequence, sensing-based SPS is now complemented by a dynamic allocation scheme, in which resources are reselected at every transmission.
- Cyclic Prefix OFDM is adopted for Sidelink too, while in Release 14 it was reserved for uplink and downlink only.
- The minimum allocation unit in time domain is called *slot* and consists of either 12 or 14 OFDM symbols (depending on the cyclic prefix setting). As the OFDM symbol time depends on the numerology, the slot time varies depending on the sub-carrier spacing. In particular, with $n = 1$ the minimum allocation unit is the subframe (1 ms), while with $n = 3$ this value is reduced to 0.125 ms, making each subframe composed by 8 slots. As in LTE-V2X, the minimum allocation unit in frequency domain is the sub-channel.
- NR-V2X introduces the possibility of using two frequency ranges: sub-6 GHz, as in LTE-V2X, and mmWave from 24.25 GHz to 52.6 GHz [84].
- NR-V2X Mode 2 now supports unicast and multicast transmission, alongside broadcast.
- NR-V2X introduces a novel Feedback Channel to improve the overall reliability of the communication.
- NR-V2X Mode 2 enables the selection of additional RRI values, i.e., any multiple of 1 ms up to 99 ms, and any multiple of 100 ms up to 1 second.

2.3.3 ETSI ITS-G5 over C-V2X

As mentioned earlier, even though ETSI ITS-G5 has been designed with IEEE 802.11p as PHY and lower MAC layers, it is flexible enough to be used in conjunction with C-V2X (either Rel. 14 LTE-V2X or Rel. 16 NR-V2X). All the messages described in Section 2.2.4 can thus be exchanged through *PC5* or *Uu*, leveraging cellular networking instead of IEEE 802.11p.

Indeed, ETSI foresees in standards such as [57] the possibility of using the ETSI Networking and Transport and Facilities layers over C-V2X, and provide requirements for additional access layer functionality for the *PC5* interface. ETSI also defines a congestion control mechanism for communication over PC5 in LTE-V2X [85], based on the ProSe Per Packet Priority (PPPP), which defines the traffic priority as in IEEE 802.11p with Access Categories.

2.4 Future directions

As the wireless technologies evolve, novel demanding use cases are emerging, such as on-board intelligence enabled by AI/ML and task offloading. The requirements for safety critical use cases and higher levels of automation, towards SAE L4 and L5 are also becoming increasingly stringent in terms of latency and required throughput. As an example, the 5G-CARMEN project has defined a maximum end-to-end latency of 10 ms for centrally managing highly automated maneuvers with 5G, which is more stringent than the latency requirements proposed by ETSI [8]–[10].

This has led to several future directions, both related to the access technologies and to the higher layers of the stack.

Concerning the first point, the need for increasingly high throughput and low latency has brought the automotive world to research the usage of mmWave technologies, which are expected to become part of the upcoming IEEE 802.11bd protocol [42].

Alongside mmWave, Terahertz communication [18] are also being investigated.

Together with high throughput and low latency technologies, long range low power technologies are also gaining interest for specific V2X use cases that do not require transmission of large quantities of data. As energy consumption is becoming a very relevant topic in recent years, technologies such as LoRa could enable the transmission of messages in V2X scenarios over long ranges with a minimal environmental footprint [86].

Concerning instead the second point, the usage of publish-subscribe messaging protocols such as Message Queue Telemetry Transport (MQTT) or Advanced Message Queuing Protocol (AMQP), is currently being experimented by several European projects, as a way to collect messages from vehicles and selectively send

them to centralized services, and, at the same time, to let the MEC services efficiently target the proper vehicles and/or geographical areas.

2.4.1 ETSI ITS-G5 over AMQP 1.0

As mentioned earlier, several V2X European Projects, such as 5G-CARMEN, are investigating the usage of messaging protocols for the efficient transmission of messages from/to centralized MEC services. These protocols can be used over Wi-Fi-based or cellular-based V2I/V2N communication, which is the focus of these projects.

In particular, the C-Roads[2] project defined a standard architecture for the transmission of ETSI ITS-G5 messages over the AMQP 1.0 protocol. It is worth mentioning how the specifications consider the 1.0 version, since AMQP 0.9 represents a radically different messaging protocol.

The architecture foreseen for the transmission of standardized messages over AMQP is schematized in Figure 2.9, taking as reference the deployment in 5G-CARMEN, which is itself based on the C-Roads specifications.

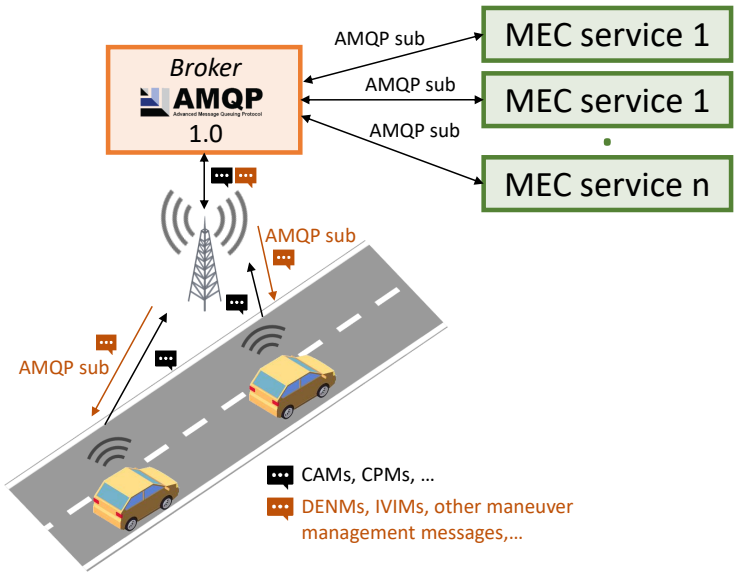


Figure 2.9: Architecture envisioned for the transmission and dissemination of ETSI ITS-G5 vehicular messages over the AMQP 1.0 messaging protocol.

As can be seen, a central entity, called *message broker*, acts as message collector and dispatcher, and collects messages based on *queues* (or *topics*), which are addressed through their name (which can be hierarchical). Vehicles transmits their

[2] https://www.c-roads.eu/platform.html

messages (mainly, CAMs) to the broker, and they subscribe to specific queues or topics for the reception of warning, road signage and maneuver management messages. On the other hand, MEC services can subscribe to the broker to receive CAMs from the vehicles of interest, based on the queue or topic they subscribe too, or on the message properties, as detailed later. These services are also able to send messages such as DENMs and IVIMs to the broker for dissemination to relevant road users (or to all users travelling in a given geographical area) subscribed to the same broker. Communication between different MEC services through the broker is also possible.

Both *queues* and *topics* act as message collection entities. The difference between them is not defined in the AMQP 1.0 specifications. However, taking as reference the definition by Apache for their ActiveMQ broker [87], in this thesis we will refer to:

- **Queue:** queue working according to load balancer semantics. Each single message can be de-queued and consumed by one consumer only. In case no consumers are available, messages will be queued and sent to the first available consumer, as soon as it subscribes.

- **Topic:** message collection entity working like a queue, but according to publish and subscribe semantics. This is the actual message collection entity foreseen by C-Roads for V2X communications. When a message is sent to a topic, it is then disseminated to all the available subscribers. If no subscribers are active, the message is not queued, and it is lost. Thus, zero to many subscribers receive each published message.

The transmission and reception of ETSI ITS-G5 messages over AMQP happens by encapsulating the ITS messages into the AMQP protocol. The latter is then encapsulated inside TCP over IP, as foreseen by the AMQP standard [88].

Two modes of transmission are possible, as analyzed in 5G-CARMEN and depicted in Figure 2.10.

The first preferred mode (mode 1) involves the transmission of the full BTP + GeoNetworking + ITS packet over AMQP, while the second mode (mode 2) involves the transmission of the "pure" ITS packet without the ETSI Transport and Networking layers. Even if AMQP can address geographical routing and protocol multiplexing through message properties, the first mode is always preferred as it allows standard-compliant MEC application to decode the packet as if it was received through a direct V2V or V2I communication.

Each AMQP 1.0 message can embed, in the header, user-defined properties, called *application properties*. These message properties are represented by *key-value* couples, in which the key should be a string defining the property name, and the value can be any simple type (mainly string or number). The main properties defines by C-Roads are listed in Figure 2.10. Thanks to the `quadkeys` property

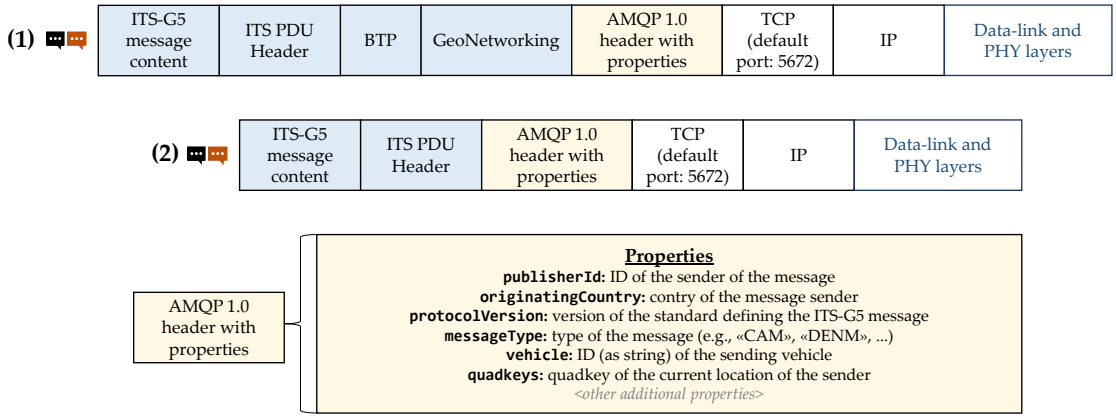


Figure 2.10: Message format for the transmission of ETSI ITS-G5 over AMQP 1.0, with the application properties defined by the C-Roads project. It should be mentioned that the `quadkeys` property is now called `quadTree` in the latest C-Roads specifications (version 2.0).

geographical routing is made possible, and the `messageType` and `protocolVersion` ones enable protocol multiplexing without the need of BTP and GeoNetworking, in case mode 2 is adopted. The C-Roads properties also include additional information which is normally not available in messages such as CAMs, e.g., the originating country or sender ID (which is typically the string ID of the OEM).

One interesting feature enabled by properties is represented by the possibility of filtering the messages at an AMQP broker level. Indeed, other than selecting a proper topic, a client subscribed to the broker can choose to receive only the messages with certain property values (e.g., only DENMs with `messageType="DENM"`). The broker will then forward to the client only the messages matching an SQL-like filter on the properties provided by the client itself. This filter can be arbitrarily complex and include both logical operators and wildcards.

The Quadkeys for vehicle localisation

Among the most useful properties defined by C-Roads, it is worth mentioning the `quadkeys`. Quadkeys are a way to identify rectangular areas on a map, and can be used to represent the geographical position of vehicles as compact strings with base 4 digits. These strings are then transmitted as message properties and can be used for geographical message filtering.

The concept of Quadkey has been first defined by Microsoft for their Bing Maps service [89], and it is based on a bi-dimensional Mercator projection of the Earth.

The map can be divided into rectangular tiles, corresponding to rectangular areas. Each of these tiles can then be uniquely identified by a Quadkey, with the

following logic: the map is divided into four parts, which are assigned, from top-left to bottom-right, four digits (0, 1, 2 and 3). These digits represent the level 1 Quadkeys, which correspond to very large areas. Each rectangular tile can be further divided in four parts, defining the level 2 Quadkeys. Each of these parts will keep as first digit the one of the parent tile (i.e., the tile from which it originates) and add a second digit again from top-left to bottom-right. The process can be interactively repeated defining progressively higher level Quadkeys. For instance, the area defined by Quadkey `2` contains the one defined by `21`, which contains the one defined by `212`, and so on.

Quadkeys are thus hierarchical strings which define progressively smaller areas as the level increases.

This mechanism is represented in Figure 2.11.

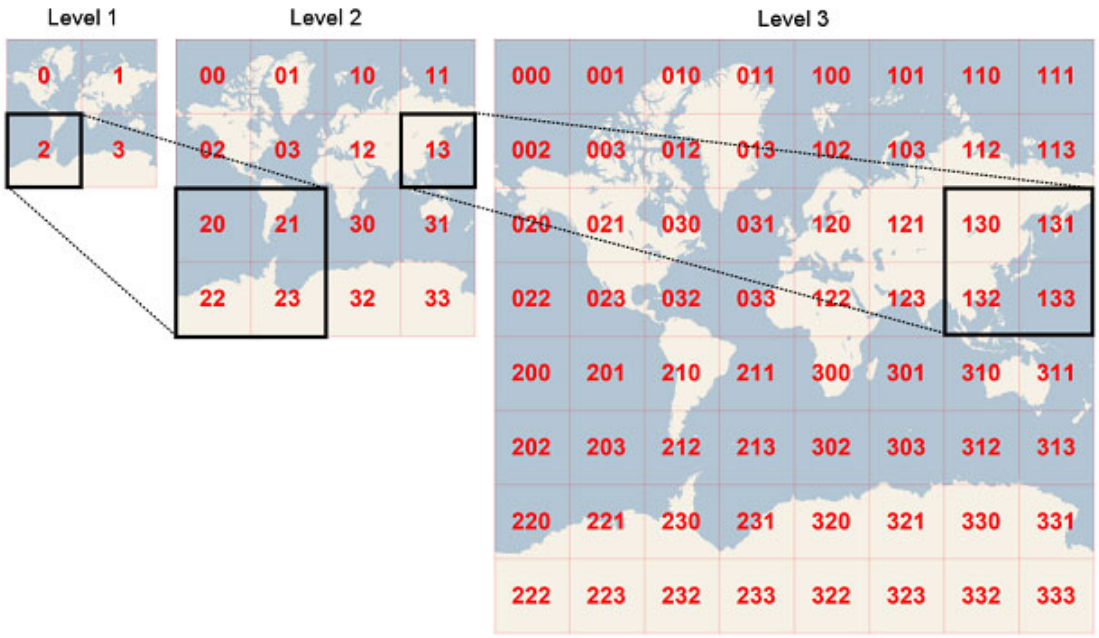


Figure 2.11: Logic for the definition of Quadkeys on a Mercator projection of the world. Copyright: Microsoft [89].

Microsoft also defines a convenient algorithm for translating a latitude and longitude value into a Quadkey [89]. Given as input the level of detail (L) and the coordinates of a point, the algorithm returns the level L Quadkey of the area which contains that point.

The algorithm first converts latitude and longitude into tile X and Y coordinates [89]:

$sinlat \leftarrow sin(latitude \cdot \frac{\pi}{180})$

$pixelX \leftarrow (\frac{longitude+180}{360}) \cdot 256 \cdot 2^L$

$pixelY \leftarrow (0.5 - \log(\frac{1+sinlat}{1-sinlat}) \cdot \frac{1}{4 \cdot \pi}) \cdot 256 \cdot 2^L$
$X \leftarrow \lfloor \frac{pixelX}{256} \rfloor$
$Y \leftarrow \lfloor \frac{pixelY}{256} \rfloor$

Then, it computes the corresponding Quadkey by taking the binary bits of X and Y, interleaving them (starting from Y), and interpreting the resulting number as a base 4 number. For instance, with $L = 3$, if X is equal to 2 and Y to 7, the corresponding binary bits are 010 and 111. Interleaving the bits brings to 101110, which corresponds to 232 in base 4. The corresponding Quadkey will be thus "232".

As can be seen, the algorithm is composed by few simple steps, which can be efficiently executed by the OBU of a vehicle to compute, given L, the Quadkey of the area comprising the current location. The size of this area (directly linked to how precise will be the localisation of the vehicle) is defined by L. This value can then be transmitted via AMQP 1.0 as a message property.

The advantage of the Quadkey approach is twofold. First, it allows representing coordinates in a compact and efficient way. Second, thanks to their hierarchical structure, Quadkeys can be leveraged for efficient message filtering and geographical routing [90].

For instance, a MEC service can subscribe to receive all messages from a certain large area by generating a filter on the `quadkeys` property with one or more low level Quadkeys (with wildcards) corresponding to the desired area. For instance, if the filter contains the string "321%" (% is just a wildcard meaning "zero or any number of other digits"), the service will receive all messages from vehicles sending (higher level) Quadkeys such as "**3211**1210012" and "**3210**012111" (i.e., all messages from vehicles travelling inside the area corresponding to the level 3 Quadkey "321"). The vehicles can thus send a high level, more precise, Quadkey, and the MEC service can easily receive all messages from a large area by specifying a low level Quadkey.

On the other hand, vehicles can subscribe to receive only warning messages containing the Quadkey corresponding to the area they are located in (with a certain L), and a MEC service can thus target vehicles located only in a desired rectangular area by tuning the value of the `quadkeys` property.

2.5 Intra-vehicular communications

Even though this thesis is mostly focuses on V2X communications, intra-vehicular communications plays a pivotal role in autonomous vehicles, as they represent the means of communication between the different on-board Electronic Control Units (ECUs). For instance, if an alert message is received by the OBU via V2X communication, the actuation command for the brakes will be transmitted to the braking system ECU by means of intra-vehicular communications.

Among the available protocols, the most widely used nowadays is Controller Area Network (CAN), a robust bus standard for the exchange of data between the

different vehicle ECUs.

At the PHY layer, CAN is a two-wire bus, terminated by a 120 Ω resistance. The signals are transmitted by means of two logic states on the bus: the dominant state for `0` (with a voltage difference between the two wires) and the recessive state for `1` (with the same voltage on the two wires). To guarantee synchronization on the bus, Non-Return-To-Zero (NRZ) coding is used.

The structure of a standard CAN data frame is composed by a header with a few flags, an 11 bit identifier (which can be extended to 29 bits if a proper *Extended Flag* is set in the header) and the Data Length Code (DLC), containing the number of bytes in the CAN payload. The payload is then composed of up to 8 bytes of data, encoding one or multiple values (e.g., engine RPMs, steering wheel angle, ABS status and control, ...).

Each ECU is assigned an identifier (ID), with the rule that lower IDs have higher priorities than higher IDs in case of multiple ECUs transmitting on the common bus at the same time.

The ID assignment, together with the format of the encoded data (i.e., which bits correspond to what values, and how each value is scaled to fit in the right number of bits), are defined by the OEM, and usually not disclosed to the public.

In modern vehicles, CAN is often used as a transport protocol for the so-called On-Board Diagnostics (OBD)-II messages. These messages follow a public standard and can be transmitted/received to/from the vehicle ECUs thanks to a dedicated OBD-II port, usually located behind the steering wheel.

The OBD-II messages follow a request-reply paradigm and can be used to retrieve from the vehicle several useful values, as a subset of the proprietary data which is normally exchanged inside the CAN bus. These values mostly include information related to engine and emissions, such as odometer speed, engine RPM, and lambda sensor status.

OBD-II messages are transmitted over CAN with a set of standardized, dedicated, low priority IDs, i.e., `0x7DF` for requests, and `0x7E8-0x7EF` for replies from the ECUs. The CAN payload is then used to encode the OBD-II frame, which is composed of: (i) the number of bytes in the OBD-II frame itself, (ii) a *Mode* flag determining if the message is a request or a reply and the category of data (e.g., real-time data, diagnostic code, ...), (iii) a standard Parameter ID (PID) to identify the kind of information requested or carried in the message (e.g., ID `0x0D` in mode `0x01`/`0x41` corresponds to the vehicle speed in km/h), (iv) four data bytes which can be either unused, such as in requests, or carrying the requested information in reply messages, and (v) one final unused data byte [91].

Chapter 3

Open source embedded solutions for V2X

As mentioned in the previous chapters, there is currently a strong standardization effort by both IEEE and 3GPP for the definition of access technologies suitable for V2X communication. The availability of open source solutions for the development, assessment and deployment of V2X systems, however, is very scarce, despite protocols such as IEEE 802.11p are at a very mature stage, being standardized almost ten years ago.

This thesis thus described several open source solutions for V2X, aimed at filling this gap and providing researchers with open solutions for connected vehicles, providing all the fundamental advantages described in Chapter 1. These solutions are also used to gather and showcase important insights on the capabilities of access technologies such as IEEE 802.11p to provide reliable low-latency connectivity to vehicles, as detailed in Chapter 4.

3.1 State-of-the-art of embedded solutions for V2X communications

Before delving into the first low-cost and open hardware and software platform for DSRC-based vehicular communications, this Section presents a brief overview of available solutions for V2X communications, taking as reference the commercial world.

With the awareness that this list will not be exhaustive, the most well-know embedded solutions for V2X available nowadays, listed by company, include:

- **Cohda Wireless** is producing dedicated hardware and software for V2X communication, and it is currently one of the most well-known company among OEMs. They propose both RSU and OBU hardware, embedding a commercial

ETSI ITS-G5 and IEEE WAVE stack, with a dedicated API for the transmission and reception of messages. They provide some support to customers, but full support is made available through a paid subscription. Among the OBU models, it is possible to mention the *MK5* IEEE 802.11p OBUs [92] and the *MK6C* C-V2X Mode 4 OBUs [93]. Cohda Wireless also provides to customers a Linux Virtual Machine to emulate one of their hardware boards, which can also be relied on for interoperability tests between different platforms.

- **Commsignia** is a company from Santa Clara, California, USA, developing Cooperative Intelligent Transport System (C-ITS) systems, focused both on V2I and V2V. Their offer includes both OBUs and dual-channel RSUs, able to support both DSRC and C-V2X Mode 4 over the *PC5* interface. Their ITS-OB4 unit [94] is a vehicular OBU implementing multiple technologies, i.e., Wi-Fi, LTE, DSRC and C-V2X. Even though it is proposed as an aftermarket solution, their stack is commercial and can be accessed through dedicated APIs (for instance, no core modifications are possible for research purposes).

- **Chemtronics** is a company partnering with NXP, Qualcomm, and other key V2X players, proposing two hybrid-technology OBUs (i.e., DSRC + LTE or DSRC + C-V2X Mode 4), plus an IEEE WAVE RSU based on IEEE 802.11p-2010.

- **Danlaw** is a telecommunication and IoT company based in the USA, proposing the AutoLink V2X Aftermarket Safety Device. This device is a dedicated OBUs which can be equipped with either DSRC or C-V2X, supporting both dual-channel (i.e., listening and transmitting on two ITS channels at the same time) and single-channel operations. According to the product presentation [95], the focus is on providing readily available commercial applications, such as Emergency Vehicle Management.

- **Lacroix** proposes the Neavia V2V Unit. Their unit is dedicated to public transportation and special vehicles and enables DSRC communication. They also provide, if needed, a proprietary ETSI or IEEE WAVE stack and, optionally, an interface to CAN bus via OBD-II [96].

- **Harman**, a company belonging to Samsung, proposes the Savari MobiWAVE dual-mode OBU. The access to the stack is provided through facility APIs, which are also needed for the development and cross-compilation of custom application. This commercial board, based on a dual core 800 MHz IMX.6 processor, is advertised to support both IEEE 802.11p and C-V2X at 5.8-5.9 GHz [97].

- Companies such as **Qualcomm** and **Autotalks** produce dedicated chipsets for vehicular communications. As an example, Qualcomm produces the C-V2X

9150 chipset, an Application Specific Integrated Circuit enabling native C-V2X *PC5* direct communication. Autotalks is instead the first producers of dual-access technology chips. Their CRATON2 automotive grade communication processor supports both C-V2X and DSRC and includes a dual ARM Cortex A7 CPU. Combined with their PLUTON2 Radio Frequency transceiver, it can provide dual-mode embedded V2X communication. It is worth mentioning how other companies propose their products based on these chipsets. For instance, **Quectel** proposes the AG-15 module [98] based on Qualcomm 9150, together with a dedicated Evaluation Board and software stack to build a full embedded board for C-Mode Mode 4 V2X communication.

- Finally, **Unex**, a Taiwan-based company producing solutions for V2X communications. other than several standard Wi-Fi modules, proposes the UNEX SOM-301 mini PCI express System-on-Modules, integrating the Autotalks CRATON2 and PLUTON2 chips. These modules can be installed on any supported board running Linux with a mPCIe slot and 5V auxiliary power supply, and can be interfaced through a proprietary V2X solution and stack developed by Unex [99]. These chips are sold in conjunction with dedicated Unex evaluation boards.

It is important to take into account that these solutions enable V2X communication, but usually at high costs (which can be over 2000 dollars per unit) and without disclosing the source code. The customization opportunities for both the academic and industrial world are thus often very little or null. Open source solutions are thus always preferred while doing research on V2X, since they have the advantages of being much more customizable and accessible to testing, performance assessment and deployment purposes.

3.2 Open source testbed for DSRC-based vehicular communications

As C-V2X is still emerging as a market solution for production vehicles in Europe, and since OEMs such as Volkswagen are investing in integrating IEEE 802.11p-based solutions, it appears that a first step has been made towards the deployment of DSRC for connected vehicles. Toyota and Lexus also announced the intention of deploying DSRC on vehicles sold in the United States starting from 2021 [100].

While IEEE 802.11bd is currently being standardized, it appears likely that IEEE 802.11p will be the protocol of choice for DSRC-based communications. Furthermore, the market of commercial devices also confirms this trend, as all the companies listed in the previous Section produce at least one product supporting at least IEEE 802.11p. Indeed, legacy IEEE 802.11p is likely to be the technology

of choice for car manufacturers seeking to equip their latest models with either ETSI ITS-G5 or IEEE WAVE communication capabilities.

This is what drove us to the creation of an up-to-date, Linux based, open source platform that can be used with any IEEE 802.11p-compatible wireless NIC and that provides all the hardware and software elements needed to deploy vehicular testbeds like the one presented in this Section. Being it open source, it is open to any improvement or derivative work and it can be used to experiment new and existing features, sharing the results with the R&D community working on vehicular communications.

Effective, low-cost, custom open source solutions for V2X can be built starting from customizable off-the-shelf hardware. Two key companies in this context are *PC Engines* and *Unex*.

Unex produces several wireless modules which can be used in the field of vehicular communications, including the Unex DHXA-222 MIMO 2x2 card, that can be used to transmit over the 5.8/5.9 GHz frequency band and which is supported by the *ath9k* Linux driver. Even though it is designed to provide standard Wi-Fi connectivity over IEEE 802.11a/b/g/n, it has been proved to work very well in the DSRC frequency range too. The UNEX DHXA-222 are indeed based on the AR9462 chip, which has been testbed as compatible with DSRC by the Rail2X project [101]. As part of the development of our platform, we also performed several additional tests to validate the capability of this wireless NIC to provide reliable and complete IEEE 802.11p connectivity.

PC Engines produces instead a series of embedded boards targeted at networking applications and supporting any compatible wireless NIC, such as the Unex cards. Among their products, PC Engines sells the APU and APU2 boards. Each of these boards provide three Gigabit Ethernet ports, one mSATA slot for the installation of a fast mSATA SSD drive, two mPCIe slots supporting Wi-Fi and cellular modules, a DB9 serial port, a push button, three status LEDs, and several internal I/O pins. These boards are all based on the x86_64 architecture and embed multi-core AMD CPUs.

There are several works in literature currently proposing the combination of off-the-shelf hardware and open-source software to build working solutions based on IEEE 802.11p. The combination between PC Engines' ALIX and Unex's DCMA-86P2 has been used by [102], [103] and [104] to develop 802.11p working solutions. The authors of [103], focused their study on the packet loss rate and latency under different speeds and distances. They proved how their devices can communicate up to a distance of 300 m. Their results are however in contrast with the work by Agafonovs *et al.*, in which the authors claim to reach up to 1.2 km almost with the same hardware setup.

A third related work by Kamat *et al.*, instead, validates their system only through DSRC spectrum analysis, while in our work we develop a full-fledged platform for basic IEEE 802.11p communication, which is then used to perform several

static and dynamic field tests, described more in detail in Sections 4.2 and 4.3.

V2X-dedicated software framework are also validated by a number of past research works. Among them, it is worth mentioning OpenWrt 10.03 Backfire modified in the Grand Cooperative Driving Challenge (GCDC) challenge, the OpenC2X-embedded platform developed by the University of Paderborn, the CVIS Operating System and several other software patches for Linux-based systems, enabling a basic V2X communication [105]–[107].

The system we have developed proposes a low-cost, open source and fully-working hardware and software solution for IEEE 802.11p, based on Linux and on the latest kernel versions. To the best of our knowledge, at the time of development this solution was the first to rely on the latest LTS kernel (4.14.63). Indeed, our system can work with almost any recent Linux kernel, from kernel 4.14.63 up to more recent versions (such as version 5.4).

The selected hardware also keeps the cost of the overall solution low, especially if compared to commercial and closed source products. Considering the cost of the hardware at the time of development, a full embedded device could cost less than 350 euros.

3.2.1 Platform description: hardware

To develop the IEEE 802.11p testing framework, we selected *PC Engines* embedded boards (as shown in Figure 3.1) which are designed to be a flexible and customizable hardware for networking systems, with the capability of running many Linux distributions and representing an up-to-date solution with good performances when compared to other available embedded boards.

Among the available models, we selected the *APU1D* boards, supporting mini PCIe cards, which are now more and more frequently found on the market with respect to mini PCI ones [108]. As mentioned earlier, the APU1D provide two slots, allowing the installation of any compatible Wireless Local Area Network (WLAN) card and increasing the customizability of this solution.

These boards are equipped with a 1 GHz, dual-core AMD G series T40E processor (with 64 bit support), 2 GB of DDR3-1066 RAM, two USB 2.0 ports and 12V DC power supply, with a relatively low power consumption (6 to 12W depending on the load). As SSD, we installed a SATA III Transcend MSA370 MLC NAND Flash SSD for each board, with a capacity of 16 GB each. It is to be noted how, however, any mSATA SSD with a capacity of at least 8 GB could work very well in conjunction with our platform. Despite the availability of an additional SD card slot connected to the USB bus, the usage of an SSD is preferable, as, in case of mid to massive I/O, the SD card may not prove to be an ideal choice.

Even though the whole platform has been developed for the APU1D boards, it can be very easily ported, with no software modifications, to the newer PC Engines APU2E4 boards, providing a more powerful 1 GHz quad-core AMD Embedded G

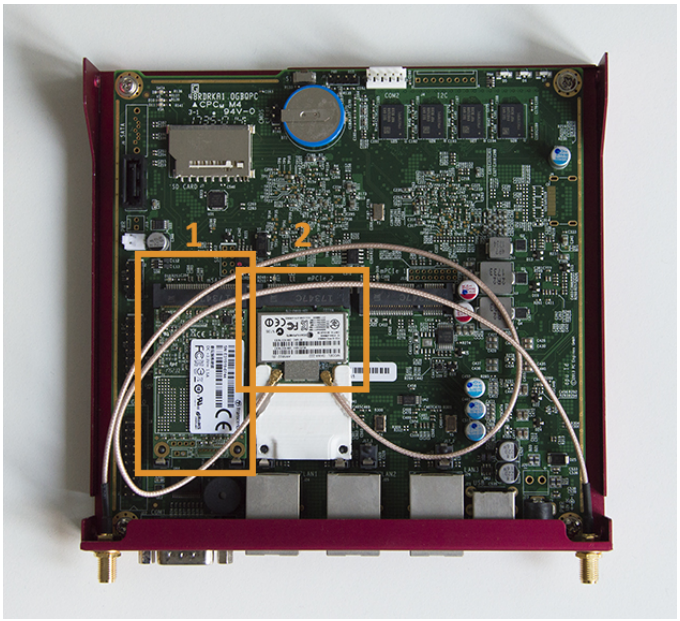


Figure 3.1: Top view of a PC Engines APU1D board, with two modules installed: (1) an mSATA 16 GB SSD, (2) a Unex DHXA-222 wireless card with two pigtails for the connection of two compatible RP-SMA antennas.

series GX-412TC CPU, 4 GB of RAM, USB 3.0 ports and better Intel Ethernet NICs with support for hardware clock synchronization [109].

As a compatible WNIC, we selected the Unex DHXA-222 modules, whose main characteristics are the following:

- Maximum power output: 18 dBm (up to 18 dBm could be selected in software for the DSRC channels, with two attached 5 dBi antennas);
- Support for spatial diversity thanks to MIMO 2x2;
- Compatible with IEEE 802.11a/b/g/n and Bluetooth 4.0, but efficiently supporting also operations in the DSRC frequency range;
- Based on the Atheros AR9462 chipset and supported by the open source Linux *ath9k* driver (which makes this platform almost automatically compatible with other modules based on the same AR9462 chipset);
- Half-size mini PCI express compact format.

The AR9462 chipset represents one of the main characteristics that drove our choice towards these modules, since chipsets supported by *ath9k* have been successfully used in a number of past research works, including the OpenC2X embedded platform [105].

3.2.2 Platform description: OpenWrt-V2X

The software, which represents the core of our solution, is based on a special version of the OpenWrt embedded Linux distribution. In particular, we selected for development the latest stable version of OpenWrt (18.06.1), at the time, with Linux kernel 4.14.63.

The choice of OpenWrt, among the available embedded Linux distributions, was led by several factors, including:

- It is supported by an active and wide community, thanks to services like a complete up-to-date documentation and an official forum with a large number of active users[1].
- It supports several architectures, besides the x86_64 one of the APU boards, including ARM for Raspberry Pi and NXP IMX.6-based devices.
- Its source code is fully open source and available in a dedicated GitHub repository[2].
- It represents a highly customizable solution, thanks to thousands of readily available packages and a flexible build system. When compiling a new image, the latter enables the selection only of the needed modules and packages, making OpenWrt particularly suitable also for low memory and cheap devices, such as network routers.
- It is up-to-date: the latest version was released few months before the time of writing. Moreover, both the Linux kernel and OpenWrt are being constantly updated with bug fixes and new features. This is quite important when working with vehicular network applications, as up-to-date software can introduce new features and patch existing ones, improving performance and possibly already include some important V2X features which already made their way inside the kernel (such as Outside Context of BSS mode, which is now selectable by means of the `iw` utility).
- Several works in the literature already adopt this Operating System for networking and V2X-related research, such as [110].

OpenWrt-V2X [20] is a patched version of OpenWrt, which includes (i) several kernel and driver patches to enable DSRC communication with supported NICs, (ii) a custom kernel and system configuration, thanks to a customized OpenWrt `.config` file (the `.config` file determines the architecture, configuration and

[1] https://forum.openwrt.org/

[2] https://github.com/openwrt/openwrt

included packages when building a new OS image), (iii) custom scripts for the automatic IEEE 802.11p OCB mode and frequency configuration, and (iv) basic broadcast transmissions C programs, thanks to the newly developed Rawsock library [111], i.e., a user-space library aimed at simplifying the usage of Linux raw sockets, needed for the transmission of broadcast packets over IEEE 802.11p.

Despite being initially developed starting from OpenWrt 18.06.1, the project has been recently upgraded to OpenWrt-V2X 21.02.1 [20], which provides updated patches for DSRC and the following additional advantages:

- Most up-to-date version at the time of upgrade, with Linux kernel 5.4.154;
- Integrated support for Docker containers[3];
- Bug fixes and updated packages;
- Updated `.config` file with selection of packages for USB GNSS receivers;
- Initial firmware support for Intel AX200 IEEE 802.11ax, alongside supported IEEE 802.11p chips. This enables the development of multi-technology hardware (e.g., RSUs), supporting both DSRC and standard Wi-Fi.

Physical layer

Two types of solutions are available for the implementation of PHY layer protocols. The first is represented by Software-Defined Radio (SDR) solutions, which implement the PHY layer algorithms in software. At least one SDR implementation has been tested in the past, using a General Purpose Processor (GPP) approach and GNU radio, as investigated in [112]. The second instead leverages the implementation in hardware by dedicated WLAN modules that can provide IEEE 802.11 connectivity over the desired frequency range. It should be noted how the first approach (i.e., using SDRs) does not natively provide MAC functions, as opposed to the second solution.

Our approach follows the second solution, as the physical layer is managed in hardware by the UNEX DHXA-222 boards.

To enable the selection of the DSRC channels (with IEEE numbering from 172 to 184), in the dedicated ITS frequency band, the *ath9k* driver needs a patch for supporting the 5.9 GHz 10 MHz-wide channels, when in Outside Context of BSS (OCB) mode[4]. For this purpose, we started from the patches provided with the

[3]Docker is a platform for the management and execution of software in self-contained packages called containers. These packages are isolated from each other and from the main operating system (even though the share the same kernel), and include all the libraries and files needed to run a desired software tool or framework.

[4]It should be noted how OCB mode is already supported in the Linux kernel and in the vanilla *ath9k* driver, which, however, normally lacks support for the DSRC frequency band.

OpenC2X embedded platform, developed by the *CCS Labs* group at the University of Paderborn [105]. These patches, developed for Linux kernel 3.18, were ported, with some modifications, to the kernel tree available in OpenWrt 18.06.1, to effectively enable the selection of the ITS channels on a modern Linux kernel. This enabled the usage of the channels with IEEE numbering from 172 to 184 [20].

This patch was integrated with an ad-hoc patch for the Italian (IT) and German (DE) regulatory database flags. Any frequency and channel bandwidth must be allowed in the regulatory database in order to be selected. Typically, the valid frequencies are calculated considering 20 MHz wide channels, hence the need to allow additional 10 MHz frequency bands below 5850 MHz and above 5925 MHz. After patching the database for the IT flag, the DE flag was included in order to maintain full compatibility with the OpenC2X ETSI compliant platform [105].

With the aim of making the system able to work with the *ath5k* driver too, we patched it in order to enable two channels (182 and 184) which seemed to be, otherwise, unselectable due to the implementation of a specific kernel function used by this driver. Thanks to this patch, and to the integration of an additional patch developed by CCS Labs for OpenC2X [105], we extended software support to older modules supported by *ath5k*, alongside the ones managed by *ath9k*.

Thanks to the now built-in capability of the `iw` Linux wireless configuration tool to manage both 10 MHz-wide channels and OCB mode, we were able to successfully transmit and receive data on any of the IEEE 802.11p channels, as required by the standard.

Furthermore, a dedicated script has been developed to run at startup, automatically setting up the board for communication in the DSRC frequency range and OCB mode. This script mainly leverages the latest version of the `iw` tool.

Together with this script, named "`iw_startup`", a set of configuration scripts and `iw` commands are made available to tune the transmission power, the selection of the channel and the physical data rate. According to our tests, among the IEEE 802.11p modulations, the UNEX DHXA-222 cards support the selection of the mandatory ones (corresponding to data rates of 3, 6 and 12 Mbit/s) through the `iw dev <interface_name> set bitrates` command.

A set of patches for the *ath10k* driver are currently work in progress for the next version of OpenWrt-V2X. These patches, after proper testing, could extend DSRC support to several additional modules based on chipsets like Qualcomm QCA-9880.

MAC layer

Concerning the MAC layer, the platform focuses on enabling the EDCA functionality, as it represents one the main features of IEEE 802.11p [46].

The Linux wireless subsystem includes a layer called *mac80211*, which enables the development of the so-called Soft-MAC drivers, in which the MAC subLayer Management Entity (MLME) [46] is implemented in software. The Linux kernel

supports, since few years, the EDCA Access Categories. However, they are normally selected by looking at the Type of Service (ToS) field inside the IP header, which may not work as expected when transmitting non-IP based packets (such as ETSI ITS-G5 packets based on BTP and GeoNetworking). We thus patched the *mac80211* layer of the subsystem with the aim of enabling the selection, by applications, of the EDCA Access Categories, both for non-IP traffic and when in OCB mode. The patch has been developed for OpenWrt 18.06.1 starting from a patch initially included in the OpenC2X project, and it enables the direct selection of an AC through a call to the `setsockopt()` system call.

Concerning instead the slot time, no patch was required, as *ath9k* already supports operations with 10 MHz-wide channel, setting a proper value of 13 microseconds. This is evident when looking at the code of the driver. The function `ath9k_hw_init_global_settings()`, in `hw.c`, indeed sets the `slottime` to `13` (microseconds) when working with 10 MHz channels (i.e., when `IS_CHAN_HALF_RATE()` returns true). The same applies to the latest versions of *ath5k*, which define the `AR5K_INIT_SLOT_TIME_HALF_RATE`, in `ath5k.h`, as 13 microseconds.

In addition, a tool for efficiently measuring the network throughput, supporting at the same time the selection of different ACs, was missing. OpenWrt, however, fully supports iPerf, which has become a de-facto standard software for throughput measurements. We thus developed an ad-hoc patch for *iPerf 2.0.12*, coding an additional command line option (i.e., `-A`) to select any of the available ACs (BK, BE, VI or VO) when generating test traffic.

Finally, it should be mentioned that, due to an update to recent Linux kernel versions, OpenWrt-V2X can properly select an AC when transmitting unicast packets, but only one priority queue can be used as far as broadcast communication is concerned. Indeed, when broadcast packets are transmitted, only AC_BE can be currently used. This is due to the introduction of the so-called *intermediate software queues*[5] inside *mac80211*, which are currently used by ath9k and several other drivers.

This feature was introduced to shift the queuing implementation more towards the software side of the wireless subsystem, allowing the hardware to keep only short queues. This also enables more fairness between different communicating devices, since these queues, when transmitting unicast data, should be kept for each station or Virtual Interface (VIF).

However, only one intermediate queue is actually allocated for broadcast and all supported drivers are coded in order to manage a single queue. An additional, fairly complex, and multi-module patch is thus needed to enable queuing of broadcast packets over AC_BK, AC_VI or AC_VO. At the time of writing, this patch is currently under development.

[5]More details are available here: https://patchwork.kernel.org/patch/6111801/

It is worth highlighting how this does not affect the reception of prioritized messages, the transmission of broadcast packets over AC_BE, and the transmission of unicast packets with different priorities.

Figure 3.2 depicts the different components and patches which were integrated into OpenWrt.

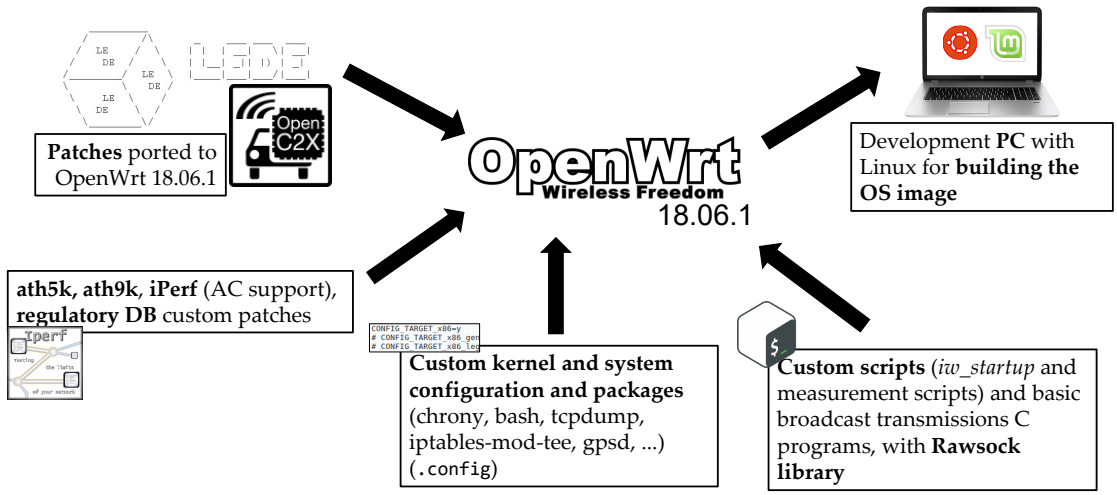


Figure 3.2: Scheme showing, at a high level, the different patches and components which were integrated into OpenWrt 18.06.1 during the development of OpenWrt-V2X.

Higher layers

With the aim of creating applications running on top of the OS, and, in general, on top of the MAC and PHY layers of the network stack, we wrote a C library enabling the use of raw sockets to send packets according to any protocol over the wireless medium. The whole library, named "Rawsock library", has been developed to simplify the usage of raw sockets, providing functions to create headers and populate payloads, through a modular design.

Currently, it supports a user-space, customizable, implementation of UDP and IP version 4, other than the direct transmission of data over the wireless medium, but it has been built in such a way that other protocols, such as WSMP (WAVE Short Message Protocol [52]), could be implemented by expanding the library without modifying most of the existing functions and without touching the Linux kernel structure, which may be subject to updates.

The library also provides two functions for injecting errors inside IP and UDP packets, for research and testing purposes, and it is available on GitHub under an open source license [111].

3.2.3 Validation through spectrum analysis

As mentioned earlier, the proposed framework has been exploited to perform several measurements on a real IEEE 802.11p channel, which are presented more in detail in Chapter 4. These measurements focused on packet loss and reachable throughput under different conditions, together with the validation of the EDCA functionality provided by the platform.

Before starting any on-field measurement campaign, we validated our platform through spectrum analysis, with the aim of proving its compliance to the standard. In particular, we selected a MetaGeek Wi-Spy DBx 3 USB spectrum analyzer, coupled with the open source Kismet Spectrum-Tools. Since there was no support for the DSRC spectrum at 5.9 GHz, we patched the tool, in order to add the possibility to analyze it, together with the implementation of few additional features which were initially not available. This patched version is available on GitHub [113], under the GPLv2 license.

We were able to verify the correct 10 MHz-wide channel usage (as shown in Figure 3.3), together with the absence of interference in the lab in which the tests took place and in which the maximum collected background signal level, between 5.850 and 5.925 GHz, was -92 dBm, which can be interpreted as pure noise.

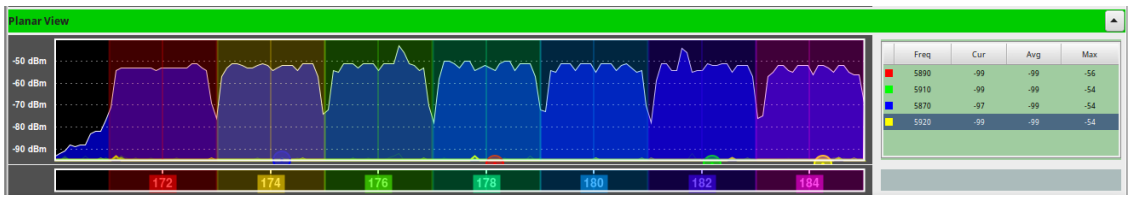


Figure 3.3: Planar view of the patched version of Kismet Spectrum-Tools, showing the spectrum captured during a test in which iPerf was set to send traffic on the seven 10 MHz DSRC channels, one at a time.

3.2.4 GUI tools

With the aim of showcasing our platform and easily performing tests through a Graphical User Interface (GUI), we have developed a set of open source GUI tools. These tools can be executed on a Linux PC connected to the APU1D boards through Ethernet. They can then control the DSRC channel and physical data rate (thanks to `iw`) thanks to the `ssh` protocol, directly from a GUI running on the PC. One of them can also be exploited to run iPerf tests with different ACs, and graphically visualize in real-time the results.

Figure 3.4 shows the graphical interface of the channel selection tool.

All the aforementioned tools are available on GitHub under the GPLv2 license [114].

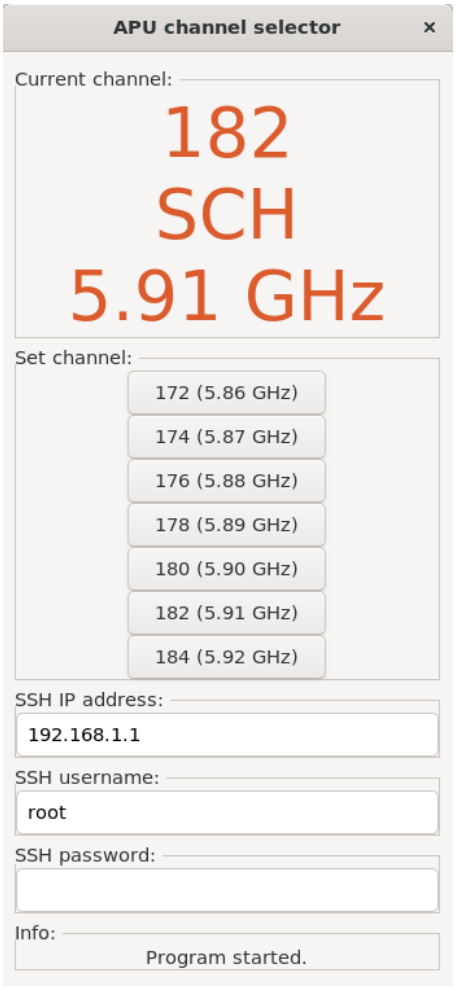


Figure 3.4: DSRC platform channel selector GUI tool. This tool can be used to easily select any DSRC channel on an APU board, connected to the PC running the GUI via `ssh`.

3.3 The LaMP protocol for precise latency measurements

Latency is one of the crucial requirements for V2X applications, especially as the concept of Ultra-Reliable Low Latency Communication has emerged with 5G networks. Indeed, as autonomous vehicles shift towards high levels of automation, latency requirements are becoming more and more stringent. ETSI has defined a 50 ms maximum end-to-end latency for certain types of applications [7], but projects such as 5G-CARMEN have reduced this value to 10 ms to provide centralized automated maneuver management.

It is thus clear how latency measurements play a pivotal role when testing,

evaluating and deploying a vehicular network and a set of V2X applications. Linked with this, the availability of reliable tools and protocols to measure latency in automotive scenarios become a crucial point.

These tools need to provide a clear understanding of the performance of the underlying system, at different layers of the stack, in order to determine the best technology (or combination of technologies), understand the performance of the algorithms deployed within a V2X application, and possibly optimize the whole system.

In general, latency can be measured at different layers of the stack: it is possible to measure a transmission and propagation delay at the physical layer, a processing delay at the transport or network layer due to the underlying OS kernel networking stack, a combination of the two (i.e., transmission, propagation and kernel processing time), and an overall delay, which is of particular interest, which can be defined as *application layer delay*. This latency represents the real latency experienced by an *application* (i.e., a V2X application, or any other network service), and comprises the transmission and propagation delay, including the often non-negligible delay due to channel access (CSMA/CA or C-V2X SPS), the kernel processing time, and the time needed to pass the packet to the application.

There is thus the need for an open protocol flexible enough to measure all these types of latency, which can be potentially encapsulated into any lower-layer protocol. Furthermore, this *lower layer agnostic* protocol should be able to provide a baseline tool for measuring the realistic *application layer* latency of V2X applications, with up to microsecond accuracy.

We thus propose a novel application-layer protocol for precise latency measurements, named LaMP (Latency Measurement Protocol). One of its core characteristics is its openness. Indeed, all the protocol specifications are public and available on a dedicated GitHub page [115].

LaMP has been designed to provide an efficient and lightweight protocol, based on a client-server paradigm, flexible enough to be encapsulated into any lower layer protocol, such as UDP, TCP, IP version 4 and 6, ETSI GeoNetworking [21] and even inside raw Ethernet or IEEE 802.11 packets. It contains all the data needed for latency measurements and supports the encoding of timestamps with a microsecond accuracy.

With the aim of enabling the encapsulation of additional data and other custom measurements, LaMP also supports a custom payload (up to 65535 bytes), and can thus act as intermediate layer for other application layer protocols.

Even though it has been designed for automotive applications, LaMP can be exploited for the performance assessment of any kind of network architecture, including 5G URLLC industrial systems. Furthermore, even if the core idea behind LaMP is the measurement of the real application layer latency, it also supports other kind of measurements. For instance, through the (optional) transmission of special *follow-up* packets, LaMP can estimate the processing delay and then return

the application layer latency minus the processing time, providing a measurement of the transmission + propagation delay, or transmission + propagation + kernel delay.

3.3.1 Review of existing latency measurement protocols

In the literature, there exist several examples of latency measurement protocols. Among these, the most notable one is the usage of the Internet Control Message Protocol (ICMP) as part of the `ping` tool. `ping` relies on two types of ICMP messages, i.e., "Echo request" and "Echo reply". Every system implementing a standard-compliant IP stack should be able to reply to "Echo requests" coming from other nodes in the network, with proper "Echo reply" packets. `ping` makes use of this mechanism by sending request packets and storing, for each transmitted packet, a transmission timestamp. This timestamp can be stored either inside the application or encoded inside the ICMP request just before sending the packet. When the corresponding reply is received, latency can be computed through a timestamp difference operation.

Rossi Mafioletti *et al.* [116] propose instead Metherxis, a system relying on Virtualized Network Measurements Functions (VNMF) to measure latency with an accuracy up to a microsecond. Linux containers (LXC) are leveraged to create a set of VNMFs under a single Linux host; some of these VMNFs are dedicated to traffic generation, while others are used as packet analyzers.

Other protocols for latency measurement include the recently proposed Two-Way Active Measurement Protocol (TWAMP) [117], supporting both one-way and two-way measurements thanks to a client-server paradigm, and a dedicated IP Timestamp Option. The latter, in particular, foresees the encoding of an optional timestamp in the IP header, to measure latency between single links [118].

These solutions, which represent the most common for latency measurements, have, however, two main drawbacks. First, `ping` relies on ICMP. Even if it can be very convenient to rely on something that is implemented inside almost every networking system and OS kernel (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. Furthermore, ICMP allows the user to more precisely compute the network latency, but tools like `ping` do not provide a clear estimate on the latency experienced at the application level. The replies are indeed often generated by the kernel, and not by a user-space third-party application [119].

Secondly, Metherxis, TWAMP, and other specific protocols, even though very precise in giving the desired measurement values, all require the tested device to be compliant either to specific standards (e.g., to provide LXC support) or to specific protocol options, such as in the IP case. They may also need the implementation of additional capabilities and features, which may not be the case for all the network nodes.

We thus introduce LaMP, a lower layer agnostic application layer protocol, and the first LaMP-compliant tool, named LaTe (*La*tency *Te*ster). LaTe is a lightweight and flexible open source application supporting LaMP over UDP and IPv4 (and, optionally, LaMP over AMQP 1.0), targeted at automotive measurements, but capable of performing latency measurements in a wide range of conditions and with a plethora of different network architectures (industrial, LAN, cellular, and so on).

As described in the next Sections, LaTe has been validated through several tests deploying different communication protocols and physical media, such as Ethernet, IEEE 802.11a and IEEE 802.11p in OCB mode.

3.3.2 LaMP protocol description

LaMP is a flexible application layer protocol, designed to be as much self-contained as possible. The protocol foresees a client-server paradigm: in a typical scenario, a LaMP client sends request packets towards a LaMP server, which replies back with LaMP reply packets. The latency is then computed as the difference between a transmission and a reception timestamp for each corresponding pair of request-reply packets. These timestamps are managed by the applications participating in the measurement session.

In the standard configuration for the computation of the Round-Trip Time (RTT), the microsecond-accurate transmission timestamp is inserted in the LaMP request packets by the client, while the reception timestamp is gathered by the same client when the corresponding reply has been received from the server and properly parsed. The reply contains a copy of the transmission timestamp embedded by the client in the request, allowing it to compute the timestamp difference as soon as each reply is parsed.

Before starting any measurement session, the client sends a connection initialization (INIT) packet, which should be properly acknowledged by the server. This is done to ensure that the measurements can start and to exchange a few test parameters (e.g., as detailed later, the type of connection to be established, either unidirectional or bidirectional).

LaMP also supports two additional main modes, called respectively *unidirectional* mode and *follow-up* mode. The first is used to compute unidirectional latency, and the client sends only requests to the server, which is responsible for computing the latency and then returning the results to the client at the end of the test. Unidirectional mode, as of now, requires the clocks of the devices running the client and the server to be precisely synchronized through third-party protocols, such as Network Time Protocol (NTP) or Precision Time Protocol (PTP). The design of a possible internal clock synchronization mechanism is currently ongoing for a future, back-compatible, version of LaMP. The second mode, instead, can be applied to both unidirectional and standard bidirectional (i.e., RTT) measurements,

and involves the transmission of additional follow-up packets after each LaMP reply (in bidirectional mode) or LaMP request (in unidirectional mode). These follow-up packets contain additional timestamp data (which could not be embedded in the original packets) to improve the initial measurement, which is normally based only on the time of transmission and reception of the request and reply packets. Focusing on RTT measurements, after each LaMP reply the server can send a follow-up packet with the best estimate of the server-side processing time. This time is then leveraged by the client to provide a more precise transmission + propagation latency (or transmission + propagation + kernel latency, depending on how the timestamps for the follow-up packets are gathered), by subtracting the server processing time from the application layer measurement. The combination of the standard and follow-up modes allows the user to compute latency at different layers of the networking stack.

Figure 3.5 shows the full structure of a LaMP packet, including the header, the optional payload, and a short description for each header field.

As can be seen, LaMP foresees a 24 B header, accounting for the need of a lightweight protocol but including all the necessary data for precise latency measurements. In order to let each node identify if the application layer data is a LaMP SDU, in an agnostic way from the lower layer protocol, the LaMP packet starts with 12 reserved bits, which are always set to alternating `0` and `1` (i.e., in hexadecimal, `0xAAA`). These bits span over a reserved 1 B field, and half of the so-called *control* field. The second-half of the control field is used to encode the packet type, accounting for up to 16 different packet types (e.g., bidirectional request and reply, unidirectional request, connection initialization, acknowledgment, ...). The choice of reserving 12 bits and leaving only 4 bits to encode the actual packet type come from the need of making the LaMP packet detection more robust. Furthermore, 16 packet types proved to be enough for all the measurement modes foreseen by LaMP.

Then, 16 bits are used to encode the LaMP ID, i.e., an identification number to uniquely identify each client-server measurement session, which is particularly important when multiple clients and servers are in execution. The header also carries a Sequence Number (16 bits), from 0 to 65535, enabling the association between each request and reply, together with the detection of lost and out-of-order packets.

The Sequence Number field is followed by 16 bits encoding either the payload length (if an optional payload can be carried in the current packet type) or an additional sub-packet type. The latter is used in INIT packets to store the type of connection which should be established (bidirectional or unidirectional), and in the so-called *follow-up control* packets, which are exchanged before starting a measurement session in follow-up mode. The follow-up control packets, which are sent by both the client and the server, encode in this field the type of timestamps which should be used by the server to estimate its processing time, and other useful

	Byte		0								1								2								3								4								5								6								7							
		Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Header	0	0	Reserved								Control 0xA | Pkt Type								LaMP ID																Sequence No.																Length or packet type																			
	8	64	Sec Timestamp																																																																		
	16	128	uSec Timestamp																																																																		
Payload (optional)	24	172	Payload (optional) – up to 65535B																																																																	
	...	...																																																																		
	65552	524416																																																																		

Fields Description

Byte	Length (B)	Name	Description	Value	
0	1	Reserved	Reserved field: it is used to determine if the packet is a LaMP packet	0xAA	
1	1	Control	Control field: the first 4 bits should always be set to 0xA, while the second 4 bits are used to encode the packet type	0xA	0x0: PINGLIKE_REQ
					0x1: PINGLIKE_REPLY
					0x2: PINGLIKE_ENDREQ
					0x3: PINGLIKE_ENDREPLY
					0x4: UNIDIR_CONTINUE
					0x5: UNIDIR_STOP
					0x6: REPORT
					0x7: ACK
					0x8: INIT
					0x9: PINGLIKE_REQ_TLESS
					0xA: PINGLIKE_REPLY_TLESS
					0xB: PINGLIKE_ENDREQ_TLESS
					0xC: PINGLIKE_ENDREPLY_TLESS
					0xD: FOLLOWUP_CTRL
					0xE: FOLLOWUP_DATA
					0xF: *RESERVED (Future use)*
[2,3]	2	LaMP ID	Identification field: it is used to uniquely identify the LaMP session	Uniform(0x0000,0xFFFF)	
[4,5]	2	Sequence No.	Sequence field: it is used to store cyclically increasing sequence numbers, up to 0xFFFF (65535). It is used to identify packets loss, out-of-order packets and to associate replies with requests	Increasing from 0x0000 to 0xFFFF	
[6,7]	2	Length or packet type	Length or packet type field: it is used to store the (optional) payload length, up to 65535B. If the packet type is INIT, it stores the type of connection that should be established (pinglike or unidirectional), if it is FOLLOWUP_CTRL, it stores the type of follow-up control packet	If packet type == INIT	INIT type (*0x0001* or *0x0002*)
				If packet type == FOLLOWUP_CTRL	Follow-up control type
				If packet type == *any other*	Maximum value: 0xFFFF
[8,15]	8	Sec Timestamp	Sec Timestamp field: it is used to store the seconds of the current timestamp	Maximum value: 0xFFFFFFFFFFFFFFFF In case of TLESS it is set to 0	
[16,23]	8	uSec Timestamp	uSec Timestamp field: it is used to store the microseconds of the current timestamp	Maximum value: 0xFFFFFFFFFFFFFFFF In case of TLESS it is set to 0	
[24,65559]	0 - 65535	Payload (optional)	Payload field: it is used to store the payload of the packet. This field is optional	User defined payload	

Figure 3.5: Structure of a LaMP packet according to the 2.0 specifications [120]. The bidirectional mode, in the open specifications, is also referred to as *pinglike* mode.

information such as if the server accepted to use the follow-up mode or denied its usage (a server can deny the usage of the follow-up mode if, for instance, it does not support the kind of timestamp requested by the client).

Finally, the LaMP packet includes two 64-bits seconds and microseconds timestamps, followed by an optional payload up to 65535 B. This payload can carry, if desired, other data and possibly other user-defined protocols, which can be thus further encapsulated inside LaMP.

After the completion of each bidirectional measurement session, the client should gather at least the following metrics [120]:

- Average latency
- Minimum latency during the whole test session
- Maximum latency during the whole test session
- Number of lost packets
- Number of packets with errors (i.e., packets which were not lost during the test, but which contained or caused errors, making the latency or RTT computation impossible)[6]

Alongside these metrics, additional user-defined statistics can be computed too (e.g., out-of-order and duplicate packets, confidence intervals and standard deviation). These statistics are then reported to the user and can be either visualized or saved to a file. In unidirectional mode, instead, the server is responsible for the computation of these metrics and for returning the report to the client, thanks to a dedicated REPORT packet (as listed in Figure 3.5).

3.3.3 The Latency Tester (LaTe) LaMP-compliant tool

After the definition of the LaMP protocol, we developed a lightweight latency measurement tool called LaTe. This tool has been written in plain C language and it works as a command line application for Linux. Furthermore, it does not require any external library, in order to make it compatible with as many architectures as possible. Indeed, it can be compiled and executed both in several Linux distributions (including Ubuntu and OpenWrt) and in Android smartphones, thanks to apps such as *Termux*, which provide a fully working command line without the need of root permissions.

LaTe is able to measure the latency between devices running Linux, and belonging either to a wireless or wired network. All the LaTe source code is available on GitHub under the GPLv2 license [121].

LaTe follows the LaMP specifications and currently supports tests with LaMP over UDP/IPv4 (since UDP is usually the protocol of choice for vehicular networks

[6] An example is the presence of an inconsistent timestamp value, which could be caused by an error when the client gathers the transmission timestamp.

when leveraging IP-based stacks). If the Apache Qpid Proton[7] AMQP library is installed, LaTe can also be compiled with a special option to enable testing with LaMP over AMQP 1.0.

Despite supporting, as of now, only UDP and AMQP 1.0, we plan to extend LaTe to support testing over additional protocols (such as ETSI GeoNetworking), and add more advanced features as well, such as a way to measure latency in broadcast flooding situations, which are often of particular interest in V2X scenarios.

By gathering the timestamps at different layers of the Linux networking stack, LaTe supports several types of latency. In particular, the current version supports four types of RTT (or unidirectional) latency measurements:

- **User-to-user**. This mode (which is the default one) measures the full application layer latency. LaTe gathers the reception timestamp, to compute the latency, as the receiving entity (typically, a client receiving a reply) has completely processed the LaMP packet. The transmission timestamp is instead gathered just before sending the request, and embedded into the LaMP packet.

- **Kernel Reception Timestamp (KRT)**. While leveraging this mode, LaTe obtains the reception timestamp as the time when the LaMP packet is passed from the hardware to the kernel stack. This is done thanks to the Linux kernel capability of generating packet timestamps. The transmission timestamp, as in the previous case, is gathered just before sending the request, and embedded into the LaMP packet.

- **Software transmit and receive timestamps**. This advanced mode is available only if supported by the NIC and by the driver. Both the transmission (sender) and the reception (receiver) timestamps are obtained from the Linux kernel, as every request is sent and every reply is received. The reception timestamp is gathered as in KRT mode, while the transmission timestamp is obtained directly from the kernel each time a request is sent, thanks to Linux ancillary data and CSMG [122].

- **Hardware transmit and receive timestamps**. This advanced mode is available only if supported by the NIC and by the driver. Usually, only Ethernet NICs with PTP support can provide hardware timestamps (such as Intel I219-V or Intel i210AT). Both the transmission and the reception timestamps (which are then compared to compute the latency) are obtained from the network adapter, as every request is sent and every reply is received. This mode

[7] Apache Qpid Proton is one of the most complete and efficient implementations of the AMQP 1.0 protocol, packed as a messaging library and available for different programming languages, including C, C++ and Python. LaTe supports AMQP 1.0 through this library.

allows the user to compute the actual transmission + propagation delay, neglecting as much as possible the kernel and application contributions from the measured latency.

LaTe exploits the philosophy on which LaMP is based, offering a flexible Linux tool supporting several user-defined options. These options include:

- Choose a transport protocol (as of now, UDP, or AMQP 1.0 if Qpid Proton is installed on the system);
- Test over a specific interface (either wired, wireless or loopback) or bind to any available interface;
- Select the amount and frequency of request packets;
- Specify a custom LaMP payload size and compare between different packet sizes;
- Select the latency type (User-to-user, KRT, Software or Hardware timestamps) and the mode (either bidirectional for RTT or unidirectional).
- Enable or disable the follow-up mode (which, at the time of writing, is still partially experimental);
- Choose and compare the usage of both non-raw and raw sockets for supported transport protocols (i.e., UDP);
- Save both the per-packet latency data and final statistics to CSV files for further manipulation, and/or send this data to a socket (thanks to a simple UDP-based protocol) for further processing by a third party application;
- Send and store the measured latency data in a Graphite database [123], and visualize it with the Grafana platform [124], as depicted in Figure 3.6;
- Several advanced options such as specifying a given end time for a test, or setting the packet periodicity according to a random distribution;
- Select any of the EDCA traffic classes [46] when testing over IEEE 802.11p; this feature is supported only when a patched kernel is available, such as in OpenWrt-V2X or in OpenC2X-Embedded [105].

The last feature is particularly relevant when assessing the performance of DSRC in vehicular scenarios and, to the best of our knowledge, it is not available in any other latency testing platform.

Moreover, our tool can compute the 90%, 95% and 99% confidence intervals, according to the Student's t-distribution, around the point average. This statistic is computed with a lookup table approach for the distribution values, to increase

Figure 3.6: Example of a simple Grafana dashboard, showing average latency data from a LaTe test over the loopback interface. Both bidirectional and unidirectional latency, coming from two parallel LaTe client-server sessions, are displayed.

the performance in low-end devices, and is reported over the packets that are sent and received in a single test session.

As detailed below, our tests have proved, on the one hand, how LaTe can be used to measure the latency in distributed embedded Linux systems. On the other, that LaTe can be used as a basic measurement tool to characterize vehicular networking systems.

Before exploiting LaTe to test the performance and characterize our open DSRC platform, described in Section 3.2, we have validated our tool by performing several measurements with different technologies. Our test setup involves two Linux laptops, with two different wireless NICs, i.e., an Intel Dual Band Wireless-AC NIC and a Qualcomm Atheros AR9460 NIC. Both laptops were running Linux Mint 19.1.

The connectivity between the two laptops has been realized with the following options:

1. Directly connected through an Ethernet crossover cable, realizing a point-to-point connection;
2. Connected through Ethernet with an 8-port 10BASE-2/T hub in between;
3. Connected through a 100 Mbit/s switch;
4. Communicating over Wi-Fi with a 5 GHz IEEE 802.11a access point in between, using a 20 MHz wide channel, after verifying the absence of interference; the Wi-Fi tests have been performed both by leaving the Linux rate adaptation algorithm as is (i.e., not trying to force any specific bitrate) and by selecting a data rate of 6 Mbit/s with the `iw` tool.

Every single test lasted for 60 seconds, over which we collected the average latency values. The same measurement session was then repeated for different values of LaMP payload sizes, up to 1448 B, which is the maximum currently supported by LaTe, in order to never exceed the Ethernet MTU. Request packets

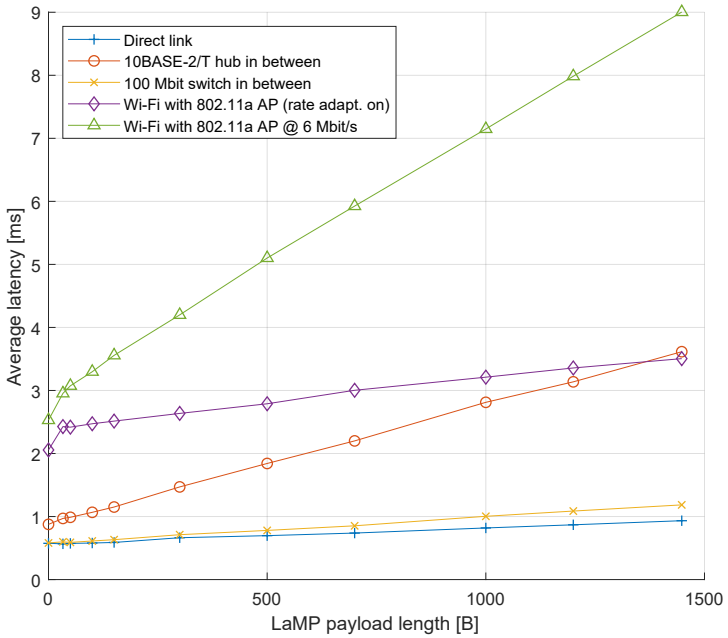


Figure 3.7: Average user-to-user latency (RTT), using a LaMP/UDP/IPv4 stack, over direct Ethernet link, 10BASE-2/T hub, 100 Mbit switch and 802.11a, on the two laptops.

were sent every 100 ms for the whole test duration (thus with a total of 600 packets for each test), which is also the maximum CAM frequency foreseen by the ETSI [63].

The most relevant results are reported in Figure 3.7, showing the average User-to-user latency as a function of the LaMP payload length (i.e., as a function of the overall packet size).

The values reported on the x axis represent the LaMP payload size: therefore, in order to obtain the full UDP payload size, 24 B have to be added, which represent the length of the LaMP protocol header.

The obtained values allowed us to fully validate our tool. Indeed, as expected, a higher payload corresponds to a higher observed RTT for all configurations, due to the increase in the transmission delay. On one hand, this latency increase is quite low when a fast channel is involved, i.e., when testing over the point-to-point Gigabit Ethernet connection, or when connecting the devices with a 100 Mbit/s switch. On the other hand, it becomes more evident, for example, with a 10BASE-2/T hub, which limits the maximum data rate to 10 Mbit/s.

Finally, the results show how communicating over a contention-based wireless medium causes the latency to be, on average, higher. The obtained values also

reveal how enabling the rate adaption algorithm normally leads to better performance as opposed to the selection of the base physical data rate (6 Mbit/s). This is due to the rate adaptation algorithm always selecting higher data rates than 6 Mbit/s, as the tests have been performed in a controlled laboratory environment without interference.

3.3.4 Latency characterization of DSRC with open V2X embedded devices

LaTe and LaMP have been used, after their design, development and validation, to assess the latency performance of a DSRC platform, taking as reference the open IEEE 802.11p framework, based on V2X embedded devices, presented in Section 3.2.

As described in the previous Sections, we used two PC Engines APU1D embedded boards, mounting UNEX DHXA-222 WNICs and Realtek RTL8111 Ethernet Controller Cards, and considered the following configurations:

1. Communicating over Wi-Fi with a 5 GHz IEEE 802.11a access point in between, using a 20 MHz wide channel; the Wi-Fi tests have been performed both by leaving the Linux rate adaptation algorithm as is and by selecting a data rate of 6 Mbit/s (i.e., the minimum IEEE 802.11a data rate);

2. Directly communicating with IEEE 802.11p, over a 10 MHz wide channel and using OCB mode, as required by the standard [46]. The three mandatory physical data rates have been selected: 3 Mbit/s, 6 Mbit/s and 12 Mbit/s. Tests have been performed at fixed distance, inside a laboratory environment, and outdoors, by varying the distance between the boards, from 0 m (i.e. the boards being placed very close to each other) to 190 m. We used a transmission power of 18 dBm, set in OpenWrt-V2X thanks to the `iw` tool.

3. Communicating with IEEE 802.11p, in OCB mode, selecting a LaMP payload size of 1448 B (to generate quite large packets) and using different EDCA Access Categories, with a parallel interfering traffic, generated by means of the patched version of the iPerf tool [20]. As detailed in Section 3.2.2, this version is capable of sending packets over different ACs. iPerf was set to generate a sizable interfering traffic, pushing data at 60% of the selected physical data rate.

The outdoor tests have been performed on the roof of Politecnico di Torino (as shown in Figure 3.8, where the absence of interference on the selected channel (i.e., channel 178) has been verified prior to the beginning of the measurements.

The LaTe configuration has been kept as in the validation tests, with each measurement session lasting 60 seconds, and packets being sent every 100 ms.

Figure 3.8: Picture showing one APU1D board with the IEEE 802.11p platform, and one laptop interfaced with the board through Gigabit Ethernet, in the place where the outdoor tests have been performed.

The first set of results is related to the laboratory tests, considering configurations (1) and (2).

Figure 3.9 compares the average user-to-user latency (RTT) measured with the APU1D boards under different configurations, when communicating over IEEE 802.11a and IEEE 802.11p[8].

Two scenarios have been considered. The first scenario is infrastructured IEEE 802.11a, in which the boards act as clients stations, with an additional Access Point (AP) to which they are connected. The second scenario involves instead IEEE 802.11p, in which the boards directly exchange packets in OCB mode. The LaMP payload size is shown on the x axis and the measured average latency values, in milliseconds, are depicted on the y axis.

The results are in line with expectations. Indeed, latency values are quite similar for low payload sizes, and they linearly increase with the payload length. This increase is proportional to the data rate associated with each modulation, and the higher the data rate, the less latency increases with the payload size.

The obtained results also show the greater latency performance of IEEE 802.11p with respect to IEEE 802.11a, when selecting similar data rates. This is particularly evident when comparing the IEEE 802.11a curve at 6 Mbit/s with the IEEE 802.11p

[8]The use of different hardware with respect to results in Figure 3.7 explain why homologous curves do not exactly match

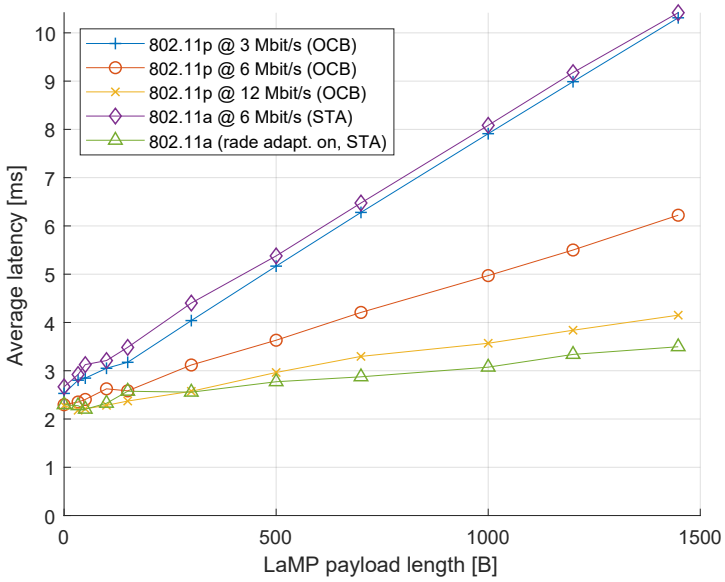


Figure 3.9: Average user-to-user latency (RTT), using a LaMP/UDP/IPv4 stack, over IEEE 802.11p and IEEE 802.11a, on the two APU1D boards.

curve at the same data rate. Indeed, the direct communication thanks to OCB mode noticeably reduces the measured latency, since each packet can directly reach its destination, without the necessity of performing an additional hop through the Access Point.

Furthermore, it is worth noticing how the curve for IEEE 802.11a at 6 Mbit/s proves to be very similar to the one for IEEE 802.11p at half of the data rate, i.e., 3 Mbit/s. This can be explained as each request and reply, when leveraging infrastructured IEEE 802.11a, need to pass through the AP, which doubles the device-to-device transmission. Indeed, one may think that the difference could be more noticeable because the delay parameters of IEEE 802.11p are doubled, but IEEE 802.11a must also account for the two-hop communication through the AP.

These results not only validate our DSRC open platform from a latency point of view, proving how it can provide a low RTT even with large payload sizes and low data rates, but also show the advantages of IEEE 802.11p.

Figure 3.10 depicts instead the results of the tests for configuration (3), showing the behaviour of IEEE 802.11p when selecting different ACs. As a reference physical data rate, we selected 12 Mbit/s, corresponding to a 16-QAM modulation, and iPerf was set to generate traffic with 1470 B-long UDP datagrams.

The plot shows on the y axis the average user-to-user application-layer latency (RTT), in milliseconds, for each AC used by LaTe. Every section of the plot is further divided into four histogram bars, showing the traffic class of the parallel iPerf interfering traffic. Each test was performed 15 times, and provides an average

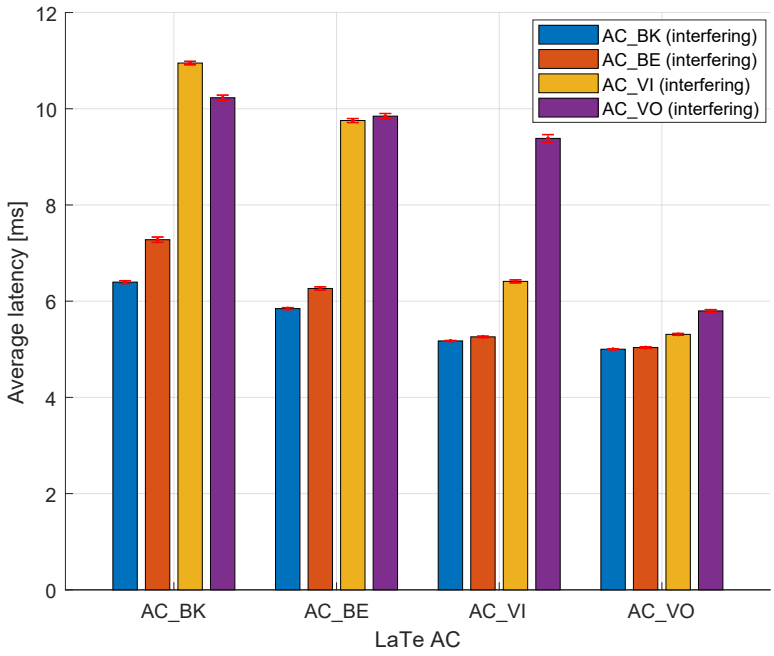


Figure 3.10: Average user-to-user latency (RTT), using a LaMP/UDP/IPv4 stack, over IEEE 802.11p and different Access Categories, with a parallel interfering traffic, over a certain AC, generated by iPerf.

value over 600 packets. We then considered as reference the average among all these tests, with 95% confidence intervals depicted as error bars. Note that the confidence intervals around the average value are quite small since all the tests, for each configuration, yielded similar results.

The obtained values are coherent with the working principles of a standard-compliant implementation of EDCA. Indeed, increasing the priority of the interfering traffic (going from AC_BK to AC_VO) leads to an increase in the latency values, for each LaTe traffic AC. On the other hand, a lower end-end latency is experienced if a higher priority is selected for the "useful" data packets transmitted by LaTe, such as AC_VI or AC_VO. However, a small decreasing trend between the AC_VI and AC_VO interfering traffic condition can be observed when LaTe leverages the lowest priority AC, i.e., AC_BK. This situation can be explained as the channel usage is quite unbalanced towards iPerf, as it generates a higher amount of prioritized traffic with respect to LaTe.

Figure 3.11 shows the results we obtained for the outdoor tests, when using LaTe to test the RTT (i.e., the "user-to-user" bidirectional latency) between the

two PC Engines boards, as they were placed at increasing distances, keeping them in Line-of-Sight. A LaMP payload size of 1448 B has been selected as a reference, to better highlight the effect of different physical data rates.

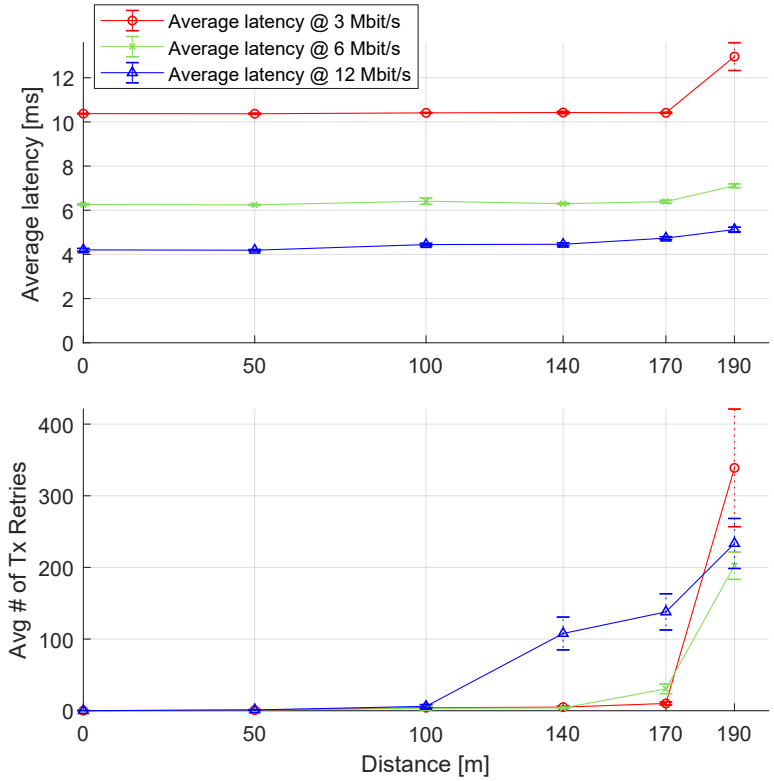


Figure 3.11: Average user-to-user latency (RTT) and number of retransmissions, measured on the *PC Engines* boards, averaged over 10 tests, using LaMP/UDP/IPv4 over IEEE 802.11p and at different distances between them (outdoor tests). 90% confidence intervals are represented too. The real-world outdoor test environment is the one depicted in Figure 3.8.

Each single 1 minute-long test was repeated 10 times, with a pause of 10 seconds between each test, and the resulting values were then averaged, computing also the 90% confidence intervals. The results show how the embedded boards can communicate at least until 190 m, with all the selected data rates, and with nearly a 0% packet loss in all the cases, as reported by LaTe. As the distance is increased, however, it is possible to notice an increase in the measured values, due to more frequent retransmissions caused by a lower received signal level. This is evident when looking at the *Tx Retries* curves, showing a correlation between the average number of retransmissions and the increased latency. Furthermore, the obtained results show how selecting a higher data rate can reduce the overall latency for large packets, but causes a higher number of retransmissions as the distance increases,

since more complex modulations are used.

Finally, an additional test on the DSRC platform has been conducted to experimentally compare LaMP and LaTe with other available latency measurement solutions, specifically, `ping` and a cross-compiled version of `twping`, an open source implementation of TWAMP as part of the perfSONAR project [125].

The results are depicted in Figure 3.12, for an IEEE 802.11p physical data rate of 3 Mbit/s.

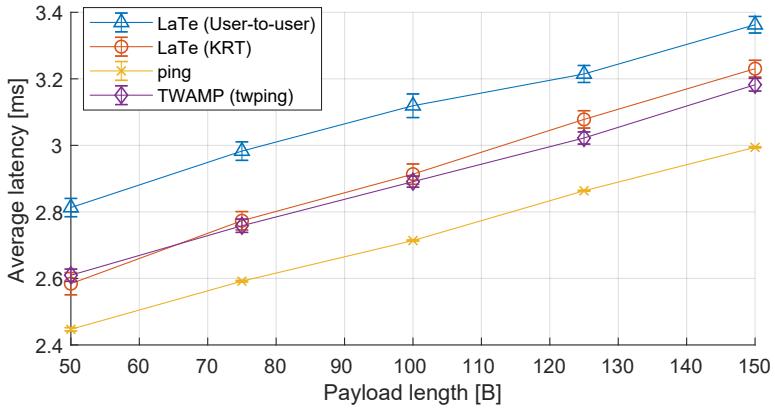


Figure 3.12: Average user-to-user latency (RTT), measured on the *PC Engines* boards, averaged over 20 tests, over IEEE 802.11p at 3 Mbit/s. 95% confidence intervals are represented too.

As mentioned earlier, it is possible to highlight how `ping` always measures a lower latency than LaTe, under the same network configuration. Indeed, this ICMP-based tool can provide a better estimate of the network latency, but not of the application layer one, which is the real latency experienced by a user when leveraging UDP to exchange vehicular messages between different V2X applications.

Furthermore, the plots shows the difference between the two main types of latency measurements supported by LaTe, i.e., User-to-user and KRT. When leveraging KRT, the measured values are normally lower than the User-to-user latency, as KRT does not consider the client-side kernel and application handling time for each reply, in the latency computation. Software and hardware timestamps were not considered for this analysis as they are not supported by most wireless NICs (including our UNEX DHXA-222), as opposed to Ethernet NICs which often support software timestamps, and, in some cases, also hardware timestamps.

Finally, it compares TWAMP with LaTe, showing how it provides comparable results, when the KRT latency is selected. Note that this comparison is made possible thanks to TWAMP leveraging UDP to encapsulate the test packets, as in the case of LaTe with LaMP over UDP.

3.4 The LTNT long term testing framework

A need that may emerge when evaluating the performance of an automotive system, especially concerning MEC platforms, is represented by the long-term monitoring of metrics such as latency and throughput. This kind of measurement can help in more reliably determining the system performance, and in monitoring its parameters to detect possible failure and/or find out where optimization should be performed.

This led us to the development of LTNT, i.e., Long Term Network Tester, a specialized open source framework for latency and throughput monitoring and long-term testing. LTNT relies on LaTe for latency, packet loss and network reliability, and iPerf for throughput measurements. Their execution is managed thanks to an ad-hoc orchestrator, which executes the proper combination of LaTe and iPerf client and servers according to the user needs, configuring them to provide, as output, CSV files which can be easily parsed by tools such as MATLAB.

LTNT designed to run on Linux (more specifically, OpenWrt, but also other Linux distributions such as Ubuntu are supported) and it is available on GitHub under the GPLv2 license [22]. The open source nature of the testing platform enables an easy deployment to any kind of test site, without any licensing issue and with the possibility of freely extending the platform according to the future research needs.

At the time of writing, the framework is capable of measuring the following metrics, by alternating iPerf and LaTe tests thanks to the ad-hoc, lightweight orchestrator, called *LTNT Test Manager* and written in plain C language:

- Latency over UDP (unidirectional and bidirectional);
- Packet loss;
- Number of duplicated packets over time;
- Number of out-of-order packets over time;
- Standard deviation of latency over UDP;
- Minimum and maximum latency in a given time interval;
- Throughput, over UDP and TCP;
- Optionally, all the other metrics supported by LaTe.

The LTNT framework is thus composed by four main software components: *(i)* LaTe, *(ii)* iPerf 2.0.13+, (iii) the LTNT Test Manager [22], and *(iv)* a tool for clock synchronization, as better detailed below.

LTNT requires two hardware boards and has been designed to support, as preferred hardware, PC Engines APU2 boards, described in Section 3.2. The two

boards should be connected via Ethernet (or, optionally, via wireless/cellular network with the installation of proper mPCIe WNICs) to the two ends of a network under test, and act respectively as *master* and *slave*. These embedded boards represent the ideal hardware since they provide three high-quality Ethernet interfaces, i.e., Intel I210-AT.

The *master* contains the tests' configuration file and executes the orchestrator in *master mode*. This instance of the orchestrator will then control the instance on the slave through a dedicated Ethernet connection (preferably, point-to-point, but also be through one or more switches). In case a cabled link is not feasible, it is also possible to leverage any wireless connection, after installing proper wireless NICs (either cellular or IEEE 802.11-based). This interface, called *control* interface, should also be used to perform clock synchronization between the two devices for all the unidirectional latency measurements. Clock synchronization in LTNT should happen with tools available within Linux, and should preferably occur thanks to the IEEE 1588 PTP protocol. The latter, in particular, can provide a sub-microsecond clock synchronization accuracy, when NICs supporting hardware timestamps are used (such as the ones available in the APU2 boards). The same link used to send control data can thus be leveraged for clock synchronization thanks to tools like `linuxptp`. It should be mentioned that, in case an Ethernet connection is not feasible for the control interface, LTNT supports the installation of a USB GNSS receiver, to provide better clock synchronization. Indeed, using a GNSS receiver should provide a much better accuracy rather than relying on PTP or NTP over a wireless link, since wireless NICs typically do not support hardware timestamps.

The configuration file contains several configuration parameters. Among the most important ones, it is possible to mention:

- IP address to reach the slave board on the dedicated *control* interface;
- IP address to reach the other end of the network under test, to which the slave device is connected;
- Name of the interface connected to the network under test (e.g., `eth0` is the left-most Ethernet port is used on the APU2 boards, when running OpenWrt-V2X);
- Name of the interface for control data and clock synchronization;
- Duration, in seconds, of each alternating LaTe and iPerf test;
- LaTe payload size and packet periodicity;
- Directories where to save the output CSV files;
- Path where the LTNT Test Manager should look for LaTe and iPerf on both the master and slave device.

The user directly controls the master board, while the slave is entirely managed by the master. The orchestrator then manages the execution of LaTe and iPerf client and servers to perform latency and throughput measurements, in both directions (i.e., master to slave, slave to master, and master to slave to master for RTT), between the two ends of the network under test, providing as output a set of CSV files. Several mechanisms are implemented to automatically resume tests and log errors in case of network issues or power failures. Furthermore, the LTNT orchestrator can be configured to run as a service (both in the master and in the slave), and its execution can be scheduled to perform automated tests.

A third Ethernet interface on the APU2 boards is instead leveraged to control the boards via SSH and gather the logs without "disturbing" any currently running measurement session. Figure 3.13 shows the preferred Ethernet port configuration on two APU2 boards, running OpenWrt and LTNT.

Figure 3.13: Suggested Ethernet port configuration and connection for LTNT, on the APU2 boards. The names of the interfaces (i.e., `eth0`, `eth1` and `eth2`) represent the default names under OpenWrt.

The LTNT GitHub page also provides a step-by-step guide for preparing and deploying LTNT on two APU2 embedded boards [22], together with information on how to replace the APU2 boards with other supported hardware, if necessary.

3.4.1 Performance assessment of an automotive edge system

LTNT has been used to assess the performance of a real-world 5G Non-Standalone (NSA) edge system in the context of the 5Growth project[9], funded by the European Commission. This edge system was composed of hardware from Nokia (i.e., the User Equipment) and Ericsson (core network and MEC servers) and was extensively tested thanks to LTNT over several months, detecting possible issues and optimizing the system during the same time span.

One device, acting as master, was connected to the UE (a Nokia Customer Premises Equipment), and the other, acting as slave, to the Evolved Packet Core (EPC). Figure 3.14 shows this setup. "SW" refers to an industrial and automotive grade switch, to let our device connect directly to the EPC and properly measure both downlink (DL) and uplink (UL) metrics.

Among the collected data, the most relevant results are shown in Figure 3.15. This data has been collected towards the end of the project, after several tuning steps, thanks to the data gathered by LTNT. Each point represents the average result of a 15 minutes-long test, concerning iPerf, or 30 minutes-long test, concerning LaTe.

As can be seen, the measurements collected over a 5-day window demonstrate an average DL latency of around 5.38 ms and an average uplink UL latency of 6.07 ms, which represent good results for the tested cellular network setup, as they are lower than what could be reached with a standard 4G LTE setup. Furthermore,

[9]https://5growth.eu/

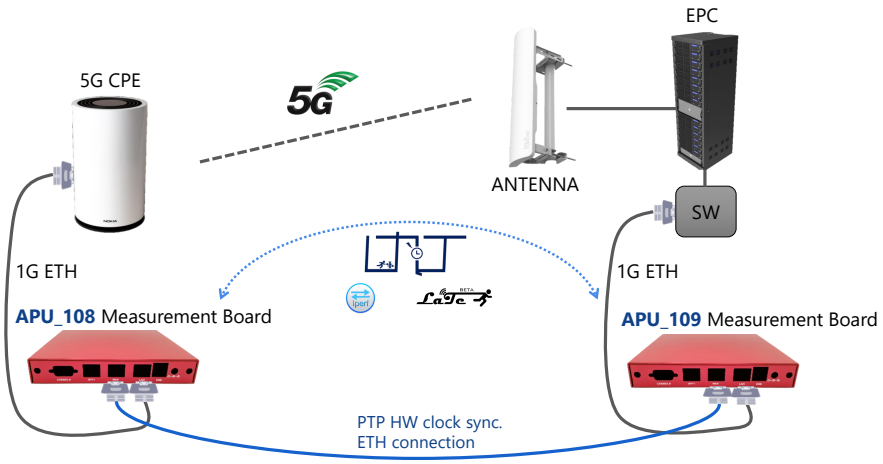
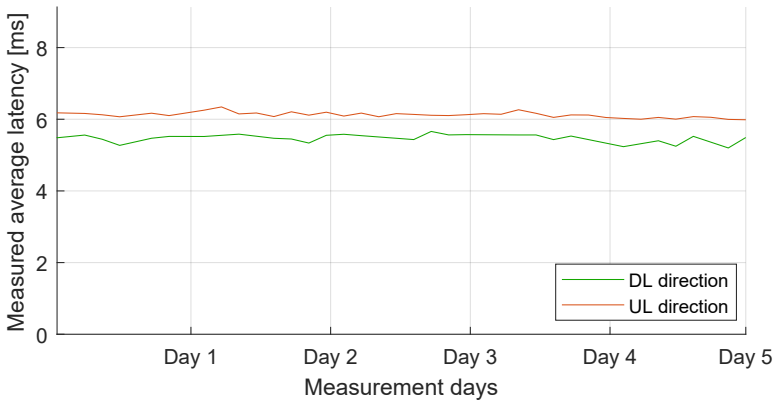
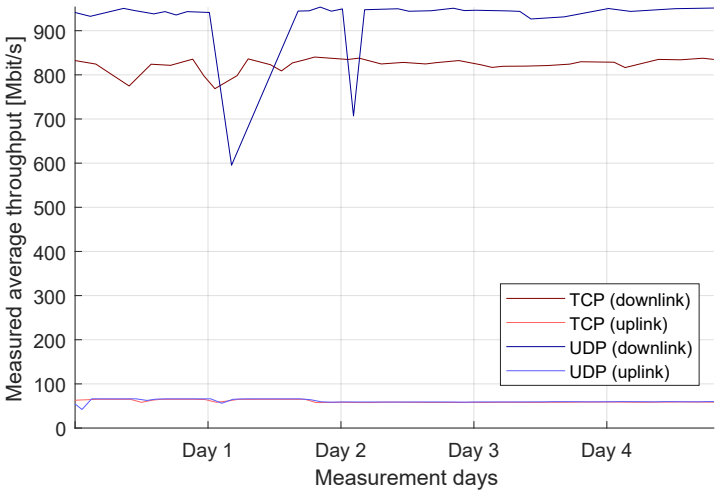


Figure 3.14: Setup of the LTNT-based testbed for the 5G NSA Radio Access Network (RAN) and edge system.



(a) Measured average latency in milliseconds.



(b) Measured average throughput in Mbit/s.

Figure 3.15: Measured average latency and throughput over a 5 days-long validation test of the 5G NSA platform, after filtering a few outliers.

the average throughput for UDP was measured to be around 922 Mbit/s in DL and 61.5 Mbit/s in UL, while the one for TCP was around 874 Mbit/s in DL and 60.6 Mbit/s. While the UL results are very similar, no matter the choice of the transport protocol, UDP appears to provide a better DL throughput overall, even through it is more prone to oscillations due to, for instance, other interfering devices connected to the same 5G NSA network, as it happened between Day 1 and Day 2.

It is worth mentioning that the trade-off between throughput and latency is tuneable to serve each use case with the most appropriate performance. Specifically, a higher throughput requires that the radio scheduler handles a greater rate of packets, resulting in a higher latency experienced on the radio link. Unfortunately,

no measurements are available concerning the configurations with a maximized throughput or minimized latency, as the focus was put on optimizing the edge system presented earlier for the requirements of 5Growth, focusing on the best trade-off between latency and required throughput.

The evaluation of the metrics provided by LTNT also provided a baseline to let the MNO and the 5G infrastructure provider understand the criticalities of an automotive 5G installation and optimize it to meet the strict requirements of the use cases mentioned in Chapter 2.1.1.

3.5 The OBU project

This currently ongoing project comes from an evolution of the open DSRC platform described in Section 3.2. The idea is to start from our platform, upgrade it to use the newer APU2 boards with 4 GB of RAM, and enrich it with support to additional technologies, provision of services and a fully open source ETSI ITS-G5 stack.

This project currently goes under the name "OBU project", as the goal is to develop a plug and play, fully functional, On-Board Unit which can be easily installed in existing vehicles for research and experimentation purposes.

As mentioned earlier, the project hinges upon the open DSRC platform, and aims at integrating other technologies, all in a single embedded board running OpenWrt-V2X. These technologies include:

- IEEE 802.11p, which is already available;
- Standard 4G connectivity through the *Uu* interface for V2I/V2N connectivity, with the plan to upgrade to 5G *Uu* as soon as 5G mPCIe modules will start to be available;
- Release 14 LTE-V2X Mode 4, leveraging hardware solutions like the Quectel AG-15 cards or the UNEX SOM-301 modules, with the plan to upgrade to NR-V2X as soon as the proper hardware will be available;
- Standard Wi-Fi at 5 GHz, which can be promising both for letting devices inside the vehicle connect to the Internet via 4G/5G, and for communication with a properly-equipped RSU, as shown in Section 4.3.

Furthermore, our OBU includes additional services, such as the transmission of CAM messages to a selected AMQP 1.0 broker via 4G (and, in the future, 5G), in line with the growing interest towards messaging protocols for V2X. This service is able to send CAM messages to a broker according to the C-Roads specifications, and it thus includes in each message, as a property, the Quadkey of the area the vehicle is currently located in, with a configurable level of detail. The transmission of ETSI-compliant messages via AMQP is performed thanks to an open source relayer

tool, namely the *UDP-AMQP relayer* that we developed, available on GitHub[10] and provided as a containerized service. This tool is able to receive any message (with or without the ETSI BTP and GeoNetworking layers) from any application running on the OBU, through a simple UDP-based loopback interface, and relay it to any AMQP broker of choice, after the insertion of the `quadkeys` property based on the current vehicle position. The open source components implementing the ETSI ITS-G5 stack, such as the CA Basic Service, will then just need to send the encoded messages (e.g., CAMs) to the relayer, which will take care of managing the full communication with the broker. It is also planned to make the open OBU receive messages via AMQP, such as event-based DENMs and IVIMs coming from an infrastructure provider.

The whole solution is based on four main design principles:

- **Plug-and-play:** to enable its easy installation into existing non-connected vehicles (which represent the great majority nowadays in Europe), of different models and brands;
- **Multi-stack:** to integrate several access technologies as part of the same device, making it future proof and able to communicate to different V2X deployments for research purposes;
- **Low cost:** as most commercial solutions have high costs which cannot be easily afforded by end users and researchers, we propose a low-cost solution based on the same hardware as our DSRC platform;
- **Open stack:** our OBU comes with an open source, lightweight, C++ implementation of the ETSI ITS-G5 stack, making our software customizable and easily extensible for research or experimentation.

Among these, the first represents the most important. Looking at today's automotive market, the penetration rate of the V2X technology is still low. However, most of the use cases introduced in Section 2.1.1 require at least a certain percentage of vehicles to be equipped with V2X connectivity, in order to be effective in practical deployment scenarios. For instance, as demonstrated by Avino *et al.* [15], a Collision Avoidance use case may require a connected vehicle percentage between 50% and 100%, in order to be effective in practical urban deployments.

As already mentioned in Section 2.3, the European market currently foresees very few vehicles already including V2X functionalities (among these, the new Volkswagen Golf 8, equipped with DSRC), and the process of equipping one for research, or, possibly, future mass deployment, is still complex and expensive. Our "OBU project" thus aims at developing an *aftermarket*, open, low-cost and plug-and-play

[10] https://github.com/francescoraves483/UDP-AMQP-relayer

OBU aimed at equipping already circulating vehicles, which do not provide any V2X functionality. This OBU can be exploited, thanks to its openness, to perform research on multiple V2X technologies and applications on the field with a full ETSI ITS-G5 stack. Furthermore, it could also provide, in the future, a useful solution for automotive manufacturers willing to easily integrate V2X functionalities in vehicles, to foster the sought-after increase in the penetration rate, needed to enable the most advanced use cases in practice.

The open OBU is based on PC Engines APU2 boards (more specifically, APU2x4, where x can be any hardware revision from C to E), running the latest version of OpenWrt-V2X to enable all the DSRC functionalities [20]. As mentioned earlier, these boards provide the same functionalities and architecture as the APU1D, but with improved hardware and performance.

The OBU also integrates a lane-level accurate Multi GNSS receiver, interfaced with the APU2 boards through USB 3.0. Positioning is indeed crucial for a large amount of V2X applications, and often lane-level accuracy is required. However, most commercial GNSS receivers, without any positioning correction service, can only provide an accuracy of 2 to 3 meters Circular Error Probable (CEP), especially in urban scenario where the satellite visibility is often sub-optimal. While this can be accepted in some cases, sometime this accuracy is not enough for safety critical collision avoidance and maneuver management services.

This has led to the birth of several differential correction schemes, to improve localization accuracy, such as Real-Time Kinematic (RTK). This approach enables the GNSS device to receive real-time corrections from a fixed reference station, which position is well known. Positioning accuracy can be thus noticeably improved, up to a centimeter-level. Corrections can be received in several ways, including through packets received via 4G or 5G cellular connectivity.

Therefore, each one of the APU2 boards is interfaced with an advanced ArduSimple simpleRTK2B-F9R GNSS RTK and Inertial Navigation System (INS) device. This device is based on the u-box ZED-F9R dual-band GNSS chipset, providing configurable update rates up to 30 Hz and RTK corrections thanks to the installation of a 4G Network Transport of RTCM via Internet Protocol (NTRIP) Client module (more details on NTRIP are available in Section 5.1.3). This module has been mounted on the simpleRTK2B-F9R main receiver, and can provide RTK corrections through 4G, when a SIM card with an active subscription is installed. The ZED-F9R chipset does not only provide lane-level accurate positioning, but it also includes an Inertial Measurement Unit (IMU). This sensor can provide additional information such as vehicle acceleration, which can be encoded in CAMs, and enable dead reckoning thanks to sensor fusion. The latter is particularly useful when the GNSS signal is lost, for instance when entering tunnels. In such conditions, the GNSS receiver is nevertheless able to provide an accurate-enough position estimation thanks to the last valid position and to the integration of the

speed, heading and acceleration measurements from the IMU. Finally, the ArduSimple devices, other than providing standard localization data (latitude, longitude, speed, heading), also disclose the raw GNSS measurements, possibly enabling advanced cooperative positioning applications, such as the ones analyzed later on in Section 5.1.

3.5.1 The first "Alpha" prototype

A first fully working prototype of our OBU has already been developed and tested on the field. This prototype is depicted in Figure 3.16, which also shows the ArduSimple GNSS receiver connected to the APU2 board through one of the available USB ports.

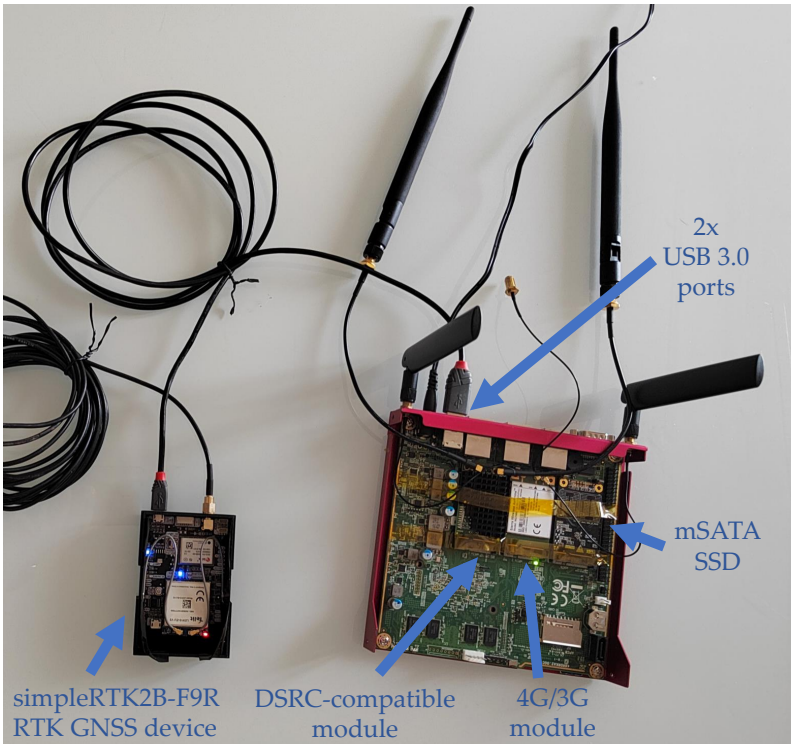


Figure 3.16: "Alpha" version of the open On-Board Unit. Notice the APU2 board with both the DSRC-compatible IEEE 802.11 mPCIe module (supported by the *ath9k* driver), and the one for 4G cellular connectivity, and the ArduSimple simpleRTK2B-F9R RTK GNSS receiver, connected to the OBU through USB.

This first prototype includes both IEEE 802.11p and 4G, while support for LTE-V2X Mode 4 and standard Wi-Fi is currently planned for the next version of the open OBU, namely the future "Beta" prototype. This prototype will also include a larger enclosure for the APU2 board, to enable the integration of an additional

"slave board" supporting LTE-V2X, which is needed due to the limited number of mPCIe slots available on a single APU board. The ArduSimple GNSS receiver will instead be kept separate from the main OBU, as it includes an IMU sensor which needs to be carefully placed inside the car to provide meaningful and accurate measurements.

Support for IEEE 802.11p is provided thanks to the open DSRC platform described earlier in this Chapter, together with proper DSRC-compatible IEEE 802.11 mPCIe modules. Although the development started by leveraging the UNEX DHXA-222 cards, we recently switched to more powerful MikroTik R11e-5HnD mPCIe modules, based on the Athreos AR9580 chipset and officially supporting IEEE 802.11a/n. These Wi-Fi modules are able to reach a maximum transmission power of 27 dBm, which is considerably higher than what can be achieved by the UNEX cards (limited to a maximum of 18 dBm). Furthermore, being supported by the patched *ath9k* driver, they fully enable communication on the seven DSRC channels at 5.8/5.9 GHz. We are still investigating the selectable data rates, but we expect these modules to support at least the mandatory modulations, corresponding to 3 Mbit/s, 6 Mbit/s and 12 Mbit/s.

4G LTE connectivity is instead provided with the installation of a Sierra Wireless MC7455 LTE mPCIe module. This module supports both 4G and, when 4G is not available, 3G, together with data operations on all the most common frequency bands leveraged by MNOs in Europe and North America (i.e., bands 1, 2, 3, 4, 5, 7, 8, 12, 13, 20, 25, 26, 29, 30, 41).

The prototype requires only a single 12 V DC power supply, with a maximum estimated power consumption of around 12 W. It can be thus easily connected, as of now, to the car cigarette lighter. Power supply, during experiments, can also be provided through a power bank supporting 12 V DC output[11], which should last long enough thanks to the relatively low power consumption.

Finally, our prototype can be optionally interfaced with the CAN bus of most modern vehicles through the OBD-II port. We were able, in particular, to successfully receive CAN frames from a Range Rover Evoque. In the presence of a CAN Database (i.e., a database storing how to decode the data inside each CAN data frame), the content of CAN frames can be directly leveraged to populate optional fields in messages, such as the turn indicator status in CAMs. In absence of a valid CAN database (which is often the case for end users and researchers), the connection to the OBD-II port can be nevertheless beneficial, as it allows the OBU to access several useful data thanks to the OBD-II protocol. The interface to the CAN bus through the OBD-II port is enabled thanks to a low-cost OBD-II to DB9 cable, and to a Kvaser Leaf Light HS v2 device[12]. The latter can be connected

[11] An example is the Litionite Tanker 90 W: https://www.litionite.com/product/tanker/

[12] https://www.kvaser.com/product/kvaser-leaf-light-hs-v2/

to the CAN bus through a DB9 connector and provides a USB interface enabling reception and transmission of CAN frames from/to the APU2 boards.

Open software components

Together with the development of the hardware solution for the first "Alpha" prototype, several open source software components are already available. They have been all designed to run on OpenWrt-V2X, and they include, first of all, the AMQP relayer container to support the transmission of CAMs to an AMQP 1.0 broker.

Then, two main software components are currently available:

- **Open CA Basic Service (OCABS):** a fully open source implementation of the CA Basic Service, including the ETSI BTP and GeoNetworking layers. It is able to send CAMs based on the data coming from the ArduSimple GNSS receiver and following the ETSI specifications on dynamic frequency management [63]. It currently supports non-secured CAMs, while support for security is currently planned for the next "Beta" prototype. OCABS is able to transmit CAMs over the DSRC interface, or to an AMQP 1.0 broker through 4G/3G thanks to the AMQP relayer tool.
- **Automotive Integrated Map (AIM):** an open source implementation of a vehicle Local Dynamic Map [26]. This component is able to receive standard compliant CAMs (and DENMs) from other vehicles, and store them in an efficient local database. This information is then made available through a web-based interface, and can be easily gathered by other applications thanks to a JSON-over-TCP interface.

The development of an open DEN Basic service is currently ongoing, together with extending support to additional messages such as IVIMs, MAPEMs and SPATEMs.

Thanks to the available software components, several low penetration rate services can be already enabled, including, for instance, car and emission monitoring and reception of event-based messages from the first DSRC RSU deployments which are starting to be a reality in several cities (including around Turin, Italy, where the prototype has been developed and tested).

3.5.2 Preliminary evaluation of the open OBU

The "Alpha" prototype has been evaluated by performing several tests both in Turin, Italy, and on a stretch of motorway in Italy.

Evaluation in urban roads

The first set of tests was aimed at validating both the lane-level RTK GNSS precise positioning, and the transmission of CAMs according to the standard. Figure 3.17 shows the trace of a Fiat Panda equipped with our prototype while travelling in an urban area in Turin. Each point correspond to where a CAM has been transmitted.

Figure 3.17: Trace of a vehicle equipped with the open OBU prototype during a test in Turin, Italy. Each white dot correspond to the position where a CAM message was transmitted.

As can be seen, lane-level accuracy is working as expected, since the GNSS receiver was always able to accurately provide the latitude and longitude of our vehicle. Furthermore, the points become denser in correspondence to curves, which is consistent with the ETSI specification, which foresee an increase in the CAM frequency when the heading difference between subsequent messages is greater than 4°.

Furthermore, during the same test session, it was possible to compute some statistics on the frequency of CAMs and on the reasons which triggered the generation of CAMs at a higher frequency. Recall that a CAMs are normally transmitted at a frequency of 1 Hz, which can be increased up to a maximum of 10 Hz. In particular, according to the standard [63], CAMs are initially generated at 1 Hz.

Then, the validity of one of the following "dynamic" conditions, usually verified every 100 ms, will trigger the immediate generation of a new CAM:

- the absolute difference between the current heading and the one included in the previous CAM is greater than 4° (*Heading* condition);
- the distance between the current position and the one included in the previous CAM (computed by OCABS by leveraging the Haversine formula [126]) is greater than 4 m (*Distance* condition);
- or, the absolute difference between the current speed and the one included in the previous CAM is greater than 0.5 m/s (*Velocity* condition);

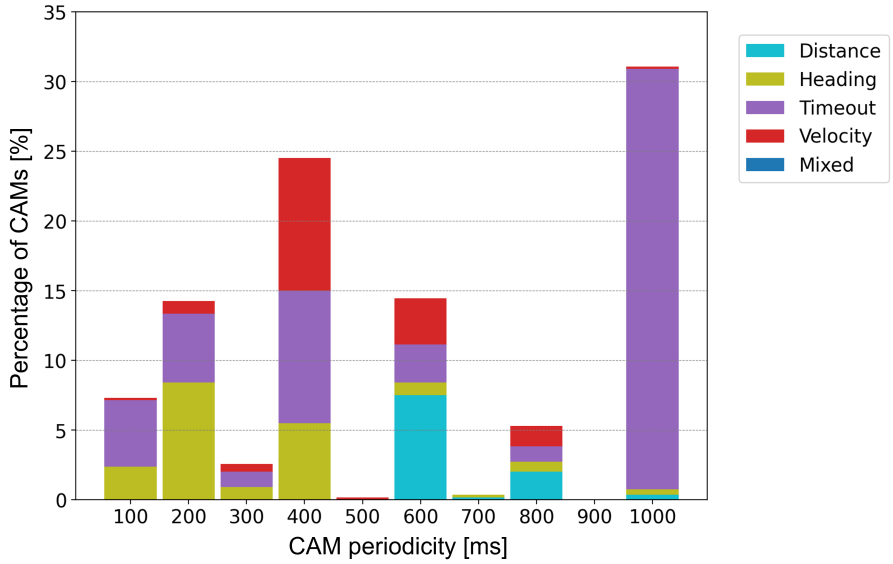
When a CAM is generated due to one of the above conditions, a parameter named *T_GenCam* is set *to the time elapsed since the last CAM generation* [63]. A fourth trigger condition is then introduced, i.e., a new CAM is triggered if the time elapsed since the last CAM generation is greater or equal than *T_GenCam* (*Timeout* condition). For instance, if a CAM is triggered in 200 ms after the previous one due to a *Heading* condition, the next CAMs will be transmitted at 5 Hz even if no dynamic condition is satisfied. *T_GenCam* is reset to 1 second (to "restore" a basic frequency of 1 Hz) after a certain number of consecutive CAMs, if no dynamic conditions are satisfied in the meantime. This number of consecutive transmission is defined by the *N_GenCam* parameter [63], which is set to 3 in OCABS.

The results are depicted in Figure 3.18.

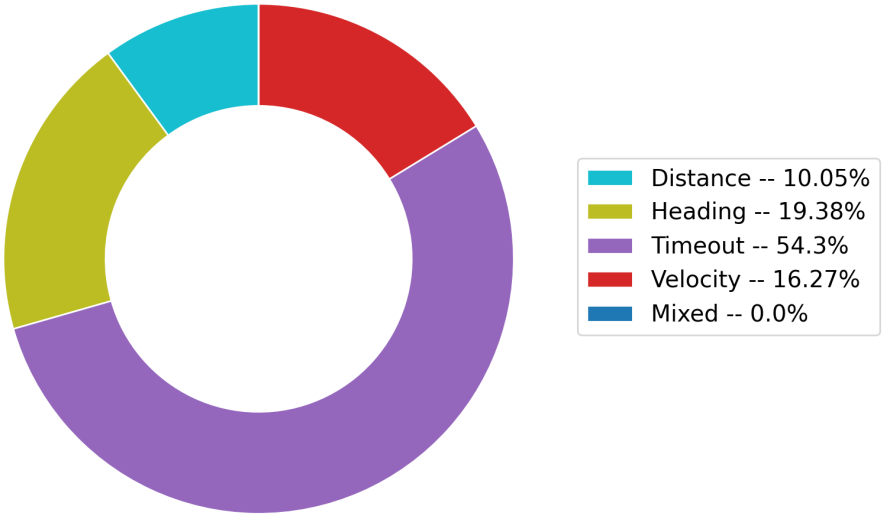
The top plot shows the percentage of CAM generations for each periodicity value, i.e., how many CAMs are generated every 100 ms, 200 ms, and so on up to 1 second, during the whole test duration. It also divides each bar according to the condition which caused the generation and transmission of each CAM. The bottom plot, instead, shows the overall percentage of CAMs triggered due to each condition mentioned earlier (i.e., either *Distance*, *Heading*, *Velocity* or *Timeout*). It should be mentioned that *Mixed* refers to the validity of more than one *dynamic* or *timeout* condition at the same time.

As can be seen from Figure 3.18a, driving in an urban scenario (with a speed limit of 50 km/h) caused most CAMs to be generated either every 100 to 400 ms, or at the lowest frequency of 1 Hz. This is due to the relatively long straight segments, which, due to the low speed, often did not cause any "dynamic" condition to be verified. Instead, when making quite narrow curves and moving on the roundabout, the heading and the speed are often subject to noticeable variations, causing CAMs to be generated much more often, especially due to the *Velocity* and *Heading* conditions.

This is also consistent with the results depicted in Figure 3.18b. Indeed, most CAMs are generated as a result of timeouts due to the low speed straight segments, and due to the moments when the vehicle was still because of the presence of other vehicles. The main reasons for increased generation frequency are instead due to



(a) Percentage of CAMs generated for each periodicity, corresponding to how many CAMs are generated every x ms during the test session.



(b) Percentage of CAMs triggered due to different conditions.

Figure 3.18: CAM generation frequency statistics during the test session depicted in Figure 3.17. The trigger conditions which determined each CAM generation are represented by different colors.

speed variations (i.e., acceleration and deceleration in curves and at intersections), or due to heading variations in curves and at the roundabout. Distance variation are instead less represented due to the overall low speed limits.

It should be mentioned that DCC, which can cause the CAM frequency to be

further reduced in case of channel congestion, is currently being implemented and it is not available yet. These results are thus intended to provide a baseline evaluation for relatively uncongested channels.

Thanks to the obtained results, it was possible to validate both the precise GNSS positioning and the proper generation and transmission of CAMs, according to the ETSI standards.

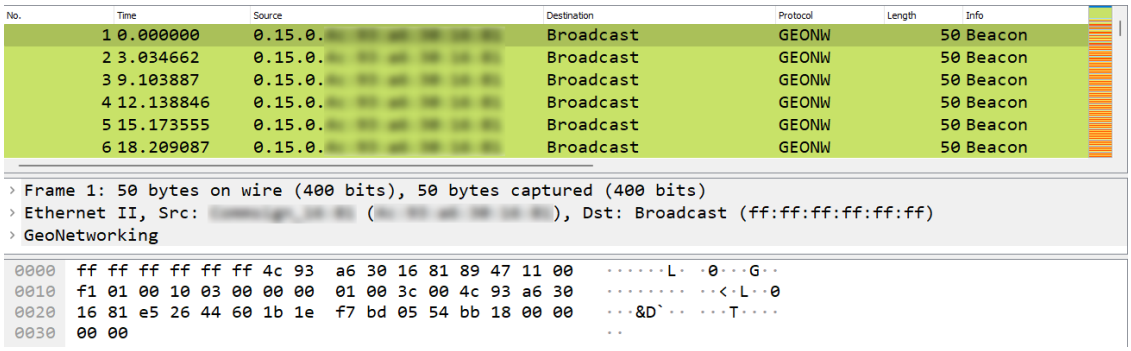
Interoperability tests with commercial RSU equipment

The second set of tests was instead aimed at testing the interoperability of our prototype with commercial RSUs, supporting IEEE 802.11p and deployed on a stretch of motorway in Italy.

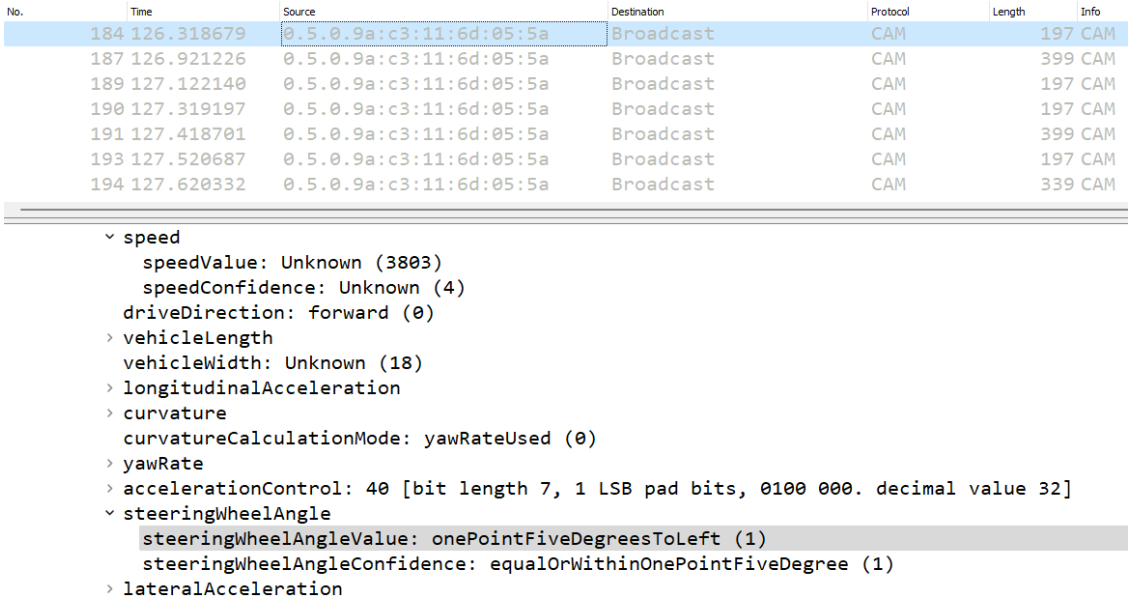
Figure 3.19: Antenna setup on the roof of our vehicle (Fiat Panda) for the interoperability and motorway tests with our OBU prototype.

We equipped the same Fiat Panda vehicle as before with our prototype, with the following antenna configuration (as shown in Figure 3.19):

- IEEE 802.11p: two automotive-grade, magnetic-mount MobileMark ECOM6-5900 6 dBi antennas, with a magnet certified up to 161 km/h;
- ArduSimple GNSS receiver: u-blox GNSS ANN-MB-00 Multiband antenna with IP67 outdoor certification;
- 4G: LINX ANT-5GWWS1-SMA 4G/5G SMA antennas, supporting a wide frequency range from 617 MHz to 4990 MHz.



(a) ETSI GeoNetworking beacons transmitted every 3 seconds by a commercial RSU. The MAC address of the RSU has been blurred as its model, address and position cannot be disclosed.



(b) CAMs transmitted by a Volkswagen Golf 8 and received by our OBU.

Figure 3.20: CAMs from other commercial devices captured by our OBU on channel 180 and displayed on the Wireshark GUI.

The first two were placed on the roof of the vehicle, while the 4G ones has been left attached to the APU2 board, inside the vehicle.

Thanks to our setup, we were able to successfully capture and decode several ETSI messages coming from the RSUs, by selecting the European DSRC Control Channel, i.e., channel 180 at 5.9 GHz. We were also able to receive and decode CAM messages coming from a Volkswagen Golf 8 (we met by chance), while it was overtaking our vehicle. This allowed us to fully validate our platform concerning the reception of standard-compliant messages from both the infrastructure and other commercial vehicles equipped with IEEE 802.11p. It should be remembered how

the Volkswagen Golf 8 is one of the few vehicles in the European market which already embeds DSRC.

Figure 3.20 shows some of the messages we were able to capture thanks to a `tcpdump` instance running in parallel to AIM and OCABS, on our OBU prototype. The messages are then displayed with the Wireshark Graphical User Interface (GUI).

We were also able to perform a set of tests on the IEEE 802.11p RSSI from the RSUs installed along the motorway, showing how the deployment is realized such that a vehicle can experience continuous coverage over a 4.5 km stretch of road. The RSUs are installed in an elevated position on one side of the road, to cover all lanes, and our vehicle was travelling in the opposite side of the motorway. The results are depicted in Figure 3.21. The plot shows the RSSI from four RSUs as a function of the distance from the point where we started the test, with our vehicle travelling on the motorway at a maximum speed of 130 km/h. It also shows the RSSI from a Volkswagen Golf 8, sending CAMs while overtaking our vehicle.

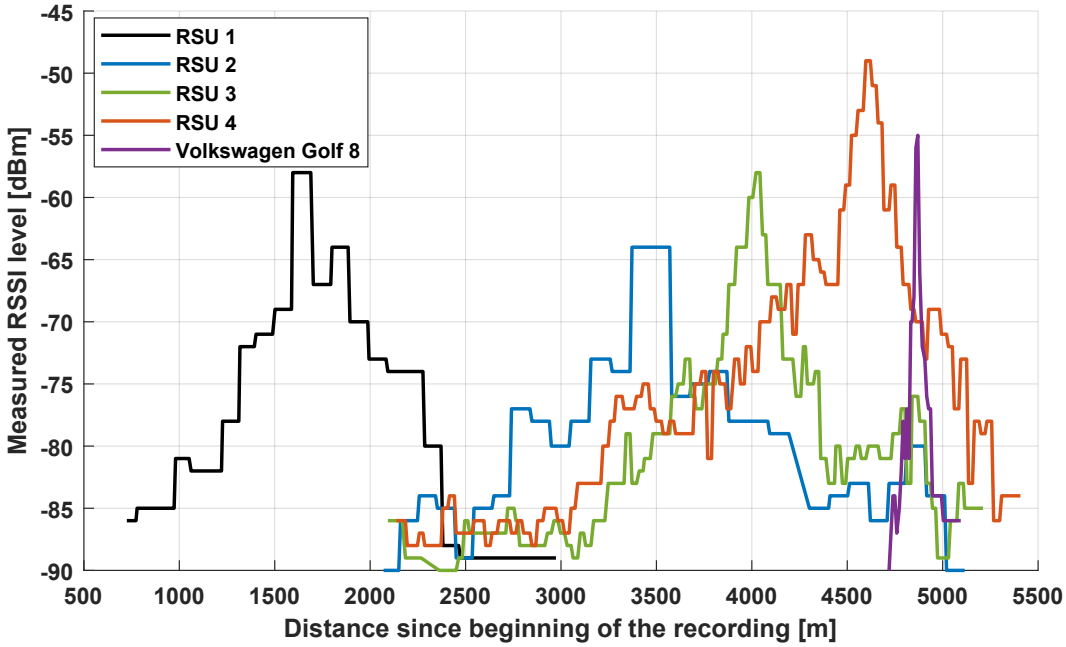


Figure 3.21: RSSI of the signal from the RSUs installed along a stretch of motorway in Italy, as measured by our open OBU. The plot also shows the RSSI from a Volkswagen Golf 8, overtaking our vehicle during the test.

The curves are characterized by several small "steps" due to the RSSI being updated only when a new message is received. Indeed, RSU 1 and RSU 2 were transmitting messages every 3 seconds, providing less fine-grained RSSI measurements along the way, while RSU 3 and RSU 4 provided finer results thanks to the transmission of messages at 1 Hz.

As can be seen, the RSUs are installed in such a way that an almost continuous coverage is guaranteed to vehicles travelling in a stretch of road of around 4.5 km. Furthermore, the measured RSSI values show *(i)* that thanks to the configured transmission power, the commercial RSUs can cover a relatively large portion of the motorway, up to around 1 km each and (ii) how our open OBU can provide reliable connectivity and receive information from properly deployed RSUs even at high speed and when the distance between the devices noticeably increases. Indeed, the measured RSSI values are overall good, considering that we were able to measured values up to -49 dBm, from the opposite side of the motorway.

Finally, the plot also shows the RSSI related to the Volkswagen Golf 8, which was continuously sending CAMs while overtaking. As expected, the RSSI first increases (as the vehicle is approaching), reaches a peak at -55 dBm, and then decreases with the same rate, as the relative speed between the two vehicles was almost constant during the overtake maneuver.

3.6 Edge-V: an open framework for Vehicular Edge Intelligence

Among the use cases enabled by vehicular networks, task offloading and Deep Leaning (DL) task execution are gaining increasing interest in the research community, thanks to the recent advances in wireless networking, edge computing and Machine Learning (ML). Indeed, as described in Section 2.1.1, a new generation of safer, smarter, and more autonomous vehicles will be enabled by the deployment of innovative use cases, which include real-time multimedia transmission and execution of complex DL tasks, such as *object detection* or *image segmentation* starting from a raw camera input.

Making vehicle smarter implies making them more aware of their surroundings, which is an often recurring topic of this thesis and of V2X communications in general. Other than the exchange of ETSI-compliant messages, vehicles may need to share their on-board sensor data, coming from LiDARs, radars and especially cameras, to enable technologies such as advanced *See Through* with high-quality video feedback, or real-time sharing of high-definition maps among self-driven vehicles for accurate localization [127], [128]. Furthermore, it is envisioned that the next generation of connected vehicles could support interactive entertainment systems for passengers based on V2V communication [129] and complex object detection tasks based on DL models and the exchange of camera outputs.

These use cases often require the execution of computationally expensive tasks, with strict latency requirements and the need for high throughput links. To this aim, task offloading can be deployed as a technique to make them practically feasible in vehicular networking scenarios. Vehicle can thus offload tasks to the infrastructure and/or to other vehicles that provide free resources, and then receive

a lightweight version of the results, containing only the relevant information (e.g., a vehicle can offload an entire frame from a camera output, and receive back the list of detected objects and their position in the frame). Task offloading is made possible thanks to V2V and V2I/V2N communication, and, as it requires ultra-low latency and very high throughput it faces several challenges. These challenges include the transmission of data from sensors, such as LIDARs that can generate up to Terabytes per hour of data [130]–[132], and the choice of a proper technology, or a combination of them, to guarantee at the same time high reliability, high throughput and low latency. Concerning task offloading, it would be possible to argue that the technological evolution in vehicles will make them able to execute all the needed tasks without the need of offloading, thanks to the deployment of proper CPU and GPU resources. However, if all vehicles would be equipped with proper resources to manage even the most computationally expensive tasks, most of these resources would be unused under normal driving conditions, leading to an expensive over-provisioning (OP) situation. Hence, task offloading is envisioned to avoid OP and provide additional resources to vehicles when their own are already used to perform several other safety-critical tasks. Indeed, several tasks are expected to be executed in parallel as vehicles become smarter and more autonomous.

With the aim of enabling such use cases, it may become pivotal to deploy the intelligence at the edge of vehicular networks (i.e., in the vehicles themselves and at infrastructure edge nodes), giving birth to the concept of Vehicular Edge Intelligence (VEI).

5G is often considered the solution to all these challenges, as its deployment is progressing across Europe and North America. However, relying solely on 5G for this kind of V2X use cases would significantly stress the already overloaded licensed spectrum bands, expected to support up to 64 billion subscribers by 2025 [133]. Furthermore, there is currently a lack of open frameworks combining different technologies for V2X to enable practical VEI and task offloading.

We thus propose Edge-V, the first open framework combining different technologies (DSRC, mmWave and standard Wi-Fi) to enable VEI and on-board AI/ML by leveraging only unlicensed spectrum bands. Edge-V is based on open source software and it provides several advantages over the usage of 5G only:

- It reduces the usage of the expensive 5G spectrum;
- It reduces the task offloading latency thanks to nearby vehicle offloading and V2V cooperation;
- It may prove essential where 5G is limited or absent [134], which, at the time of writing, is often the case in rural areas and small towns.

Existing technologies, when deployed alone, cannot guarantee the requirements of VEI. Indeed, technologies such as C-V2X Mode 4 [58] and IEEE 802.11p [46] are typically characterized by peak data rates below 30 Mbit/s [11]. As mentioned

earlier and demonstrated in Chapter 4, mmWave appears to be a very promising technology to tackle the low latency and high throughput requirements [11], [12], [42], [135]. However, it cannot be used as standalone, as it is well known that it suffers from relatively high path loss and loss of connectivity due to blockages [136]. It becomes thus critical to combine different technologies, to concretely realize reliable and effective VEI and provide, at the same time, reliable connectivity and high throughput.

Therefore, Edge-V is based on the combination on three different technologies, and has been extensively evaluated through in-lab and field tests, thanks to the development of a working Proof-Of-Concept (POC). These three technologies are, respectively, 60 GHz mmWave connectivity, V2X-specific ITS links using the 5.9 GHz DSRC band, and IEEE 802.11ac Wi-Fi at 5 GHz[13], and include the exchange of VEI-specific information among vehicles.

Furthermore, the POC described in Section 3.6.5 leverages a novel combination of low-cost off-the-shelf evaluation boards for IEEE 802.11p, IEEE 802.11ad and IEEE 802.11ac, including our open DSRC platform, and is controlled by Linux-based open-source code.

Finally, the evaluation of our framework is based on two main applications, which allow us to showcase the potentiality of Edge-V:

1. Direct data exchange between vehicles with high throughput and low latency;
2. DL-based object detection in a VEI scenario.

3.6.1 Review on VEI, mmWave and task offloading

Before presenting the Edge-V framework, it is worth providing the reader with a brief review on existing approaches and works on VEI and vehicular task offloading.

While V2X has been extensively researched in the past years, with the advent of access technologies such as IEEE 802.11p and C-V2X, VEI is still in its infancy, given the rapid development of DL-based algorithms specifically designed for vehicular applications [137]–[139].

As mentioned earlier, VEI is characterized by the critical need of establishing high-bandwidth and low latency communication among vehicles, which can be potentially moving at high speed. This makes the VEI field particularly challenging, together with task offloading in vehicular scenarios. It becomes thus crucial to combine different protocols and consider promising high throughput technologies such as mmWave. Even though, with the advent of IEEE 802.11ax, the throughput

[13]It should be noted how the choice of IEEE 802.11ac over IEEE 802.11ax was driven by the hardware and software availability at the time of development. IEEE 802.11ax mPCIe cards were indeed not so easily available, and an open source driver for Qualcomm-based IEEE 802.11ax chips was not available either.

gain achievable by mmWave may no longer be a decisive factor, mmWave provides several advantages. These include spatial reuse, as technologies working over the sub-6GHz spectrum do not provide a much higher aggregated throughput in a given location.

The application of mmWave to vehicular networks is currently being investigated by a number of research works, focusing also on IEEE 802.11ad, working in the 60 GHz unlicensed spectrum. As demonstrated in Section 4.3, this technology leverages beamforming and can provide ranges up to more than 100 m in Line-of-Sight conditions, and 30 to 40 m in Non-Line-of-Sight conditions, guaranteeing very low delays and high throughput, up to more than 1 Gbit/s [12]. Therefore, it appears to be very promising for the application in vehicular networks, although several challenges need to be faced, including beam management, impacting NLOS conditions and beam blockage recovery [140].

Among the most interesting works on mmWave and V2X, Molina-Galan *et al.* [135] proposed to decouple the mmWave data and control plane in V2X scenarios, and to use sub-6 GHz LTE-V2X Mode 4 as control plane to schedule the data transmission over mmWave. Their work is entirely based on simulations, as opposed to Edge-V, which leverages and evaluates mmWave on the field by equipping real vehicles. In [11], instead, Li *et al.* proposed a combination of Software Defined Networking (SDN) and mmWave links, with backhauling based on IEEE 802.11ad and LTE-V2X (for communication with a base station through the *Uu* interface), together with DSRC to carry control plane signals. Edge-V, as opposed to the work by Li *et al.*, does not involve LTE-V2X and leverages only unlicensed spectrum technologies.

Several works have focused instead on the development of algorithms for task offloading in the vehicular edge [134], [141], [142]. For instance, Tang *et al.* [143] proposed a cache-enabled task offloading system for vehicular networks. Their architecture focuses on offloading tasks to nearby RSUs or base station. Higuchi *et al.* [144] defined instead the concept of virtual edge servers composed by vehicles, to which tasks can be efficiently offloaded. It should be mentioned how all these works evaluate the algorithms only through simulations, without considering the limitations that may be brought by real hardware. Edge-V, instead, aims at providing a framework to develop, deploy and test these and other approaches in the field, with a strong focus on real hardware and on the combination of real-world wireless technologies. Furthermore, the design of Edge-V comes with the definition of a custom and effective solution for DL task offloading, based on a model of the overall system, as better detailed in the next Sections.

3.6.2 Modules and interfaces

The Edge-V framework, with its different modules, is represented in Figure 3.22. As can be seen, Edge-V is designed to be deployed, with different modules,

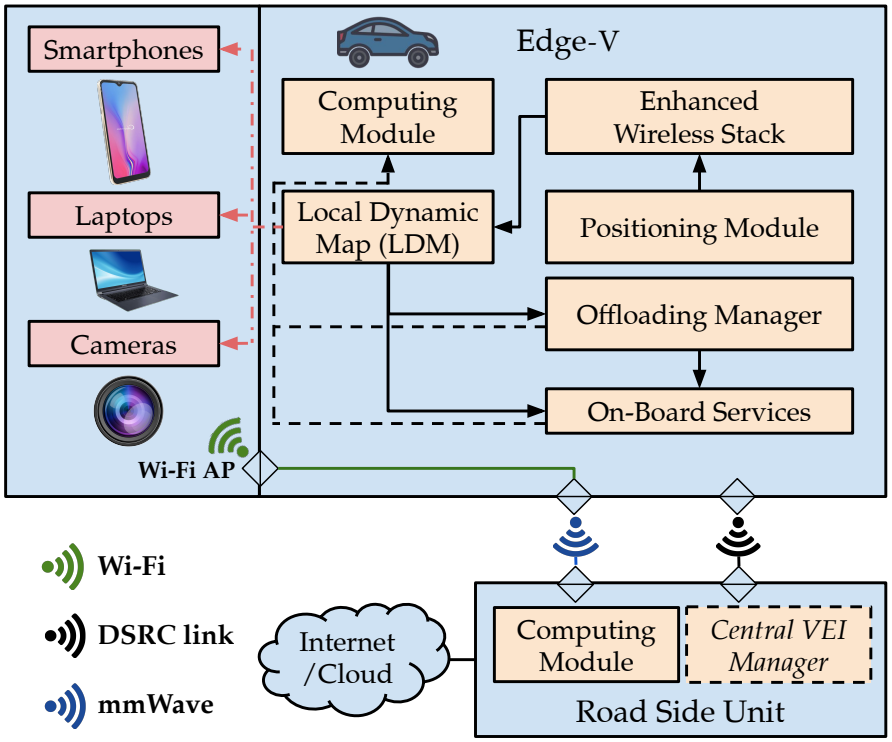


Figure 3.22: High-level architecture of the Edge-V framework.

on both connected vehicles and infrastructure nodes (i.e., RSUs, which can be leveraged for the connection to a MEC server or cloud data center). Edge-V, as a key advancement, combines computation and communication in unlicensed spectrum to deliver V2X and VEI capabilities to existing vehicular networks.

The main modules and components of the framework can be summarized as follows:

- **Radio interfaces.** Each vehicle is equipped with three radio interfaces. Two of them are external (DRSC and directional mmWave), and one of them is internal to the vehicle (5 GHz Wi-Fi). The latter comprises a Wi-Fi AP bridged with the mmWave interface (the green line represented in Figure 3.22), to provide the on-board devices with access to the Internet, through the RSU, and enable seamless communication between them as well as between different vehicles equipped with mmWave. These devices include cameras on-board of different vehicles, enabling use cases such as See Through with high quality video feedback. Furthermore, the presence of the RSU enables a data exchange between the on-board devices through the RSU itself, realizing, if needed, a mmWave V2I2V (Vehicle-to-Infrastructure-to-Vehicle) communication, ad defined in Section 2.1. This approach enables the exchange of data between vehicles even in case of noticeable NLOS blockage or distances higher than a

few hundreds of meters.

- **Computing module.** Each vehicle and RSU has computing resources in terms of RAM, CPU and GPU. These resources can be leveraged to execute tasks and on-board services locally, or can be exploited by other nodes to offload their tasks.

- **Local Dynamic Map.** Edge-V includes a standard-compliant LDM [26], which has been enhanced to store information about available resources on nearby vehicles (and, possibly, infrastructure nodes), together with other VEI-specific information. These include computational load and channel load metrics for each vehicle stored in the map. Vehicles running Edge-V can exploit such information to determine dangerous road conditions, or to optimize decisions such as where to offload data. The LDM is populated thanks to the exchange of standard-compliant vehicular messages, exchanged through the DSRC link, and it provides enhanced awareness to vehicles. Other than the direct reception via DSRC, the LDM can also receive messages from other vehicles through both multi-hop and from the RSU. This allows the Offloading Manager to have situational awareness, letting it compute the number of mmWave hops which may be needed to reach each node.

- **Enhanced Wireless Stack.** Each vehicle is equipped with a standard compliant ITS wireless stack, i.e., an ETSI ITS-G5 stack for Europe or an IEEE WAVE stack for the US. This stack is enhanced to include an optional container in periodic messages (e.g., CAMs or BSMs) for the transmission and reception of channel and node load data. The broadcast exchange of standard-compliant messages occurs through the DSRC link.

- **Positioning Module.** Positioning represent one of the key enablers of vehicular networks, as will be detailed in Section 5.1. Each vehicle running Edge-V is thus equipped with an external or embedded GNSS receiver, providing localisation and kinematic data to the ITS stack for the generation of standard-compliant messages, which are in turn used to populate the LDMs of the other vehicles. The positioning information is one of the key data leveraged by the Offloading Manager to perform the decision process. As it represents a quite critical information, RTK GNSS receivers can be used to reach lane-level accuracy, and correction can be received through the DSRC link (leveraging RTCMEM messages) or thanks to the mmWave link towards the RSU.

- **On-board Services.** They represent the core on-board services running on the vehicle OBU. They may include, for instance, collision avoidance, object detection through task offloading, See Through, automated maneuver management, indication on nearby parking spots and road works, and many more. As depicted in Figure 3.22, these services can retrieve data from the enhanced

LDM when needed, including dynamic, node and channel load and network data for each nearby connected vehicle. Furthermore, they can also retrieve useful data either from the ITS stack or from the high capacity mmWave link, and can use the on-board computing resources.

- **Offloading Manager (OM).** This component represents the core of Edge-V. It is in charge of selecting which are the best vehicles to perform task offloading and to communicate with, and deciding which is the best link to use. The decision is based on the information available in the LDM and on mathematical optimization or AI-based algorithms. This component is also in charge of managing the local computing resources and to determine the best route towards any destination in case multiple mmWave hops are required. Indeed, when performing high-speed data exchange between on-board devices, it can determine which is the best route and provide it to the related On-Board Services.
- **Road Side Unit (RSU).** As mention earlier, Edge-V can be deployed on an RSU with a smaller and different set of components. As for vehicles, the RSU is equipped with two external interfaces, i.e., mmWave and DSRC, while it does not include any internal Wi-Fi interface. The RSU includes a Computing Module, and, optionally, a Central VEI Manager, and it is connected to the Internet. Indeed, the RSUs are used by Edge-V for connection to the broader Internet, enabling further offloading of DL-based tasks to the cloud in case no vehicles have enough on-board resources to reach the desired results. They can also execute tasks locally, if needed, as they can host a MEC server and use the internal available resources (CPU, RAM and GPU). Furthermore, a Central VEI Manager can be optionally deployed, to centrally manage groups of vehicles in a centralized VEI approach. It should be highlighted that this component is optional, and Edge-V is designed to work independently from a centralized unit.

An Edge-V Walk-Through

After describing the modules and components of Edge-V, this Section presents a brief walk-through of the main operations performed by Edge-V, focusing on two real-world use cases.

The first, and probably most significant, is a low-latency, DL-task offloading use case in a VEI scenario:

1. In our example, multimedia sensors generate DL tasks (e.g., *object recognition* from the output of a HD camera), which are captured through the Wi-Fi interface.
2. At the same time, Edge-V is receiving enhanced ITS messages which are used to populate the enhanced LDM.

3. As DL tasks are being generated, Edge-V checks the available on-board resources thanks to the OM, which directly gathers this information from the Computing Module.

4. If the on-board resources, on the vehicle, are not enough, the OM selects the best remote vehicles to offload the task to. If no vehicles with enough resources and stable enough connection are available, task offloading will occur in the MEC/cloud thanks to the communication to one or more RSUs. To compute the needed information, the OM uses the information available in the LDM, solves an optimization problem and provide the results to the On-board service which needs the results of the DL task execution.

5. The On-board service can then offload the tasks thanks to the mmWave links.

The second use case is instead related to high-speed multimedia streaming between connected vehicles through mmWave, with a Human-Machine Interface (HMI) connected to Edge-V thanks to the internal Wi-Fi AP:

1. The vehicle requests to receive a multimedia stream from another remote vehicle.

2. The vehicle is equipped with an on-board HMI, which needs to receive the video stream from the remote vehicle. The HMI can make use of the content of the LDM to understand where the video sources are and subscribe the destination vehicles. As soon as the server address and location are known, the HMI can send a subscription request to the server through Wi-Fi.

3. The request is then seamlessly routed to the mmWave links and forwarded to the destination.

4. The video stream will then be transmitted back over the same link.

5. Thanks to mmWave, which provides a high-throughput link (as shown in Section 3.6.6), and to Wi-Fi, multiple video streams could potentially be accommodated in parallel towards different on-board devices.

3.6.3 The VEIP problem formulation

To practically enable VEI, optimized task offloading to other vehicles and infrastructure nodes with free resources should be fully supported and integrated with DL-based and other innovative use cases.

For this reason, the OM can employ a mathematical model of the whole system to perform optimal decisions on where and how to offload tasks, at each time slot. This problem is called Vehicular Edge Intelligence Problem (VEIP).

The main features of our model for VEIP can be summarized as follows: first, our model prioritizes offloading to other vehicles to reduce the load on the remote edge or cloud servers. Second, it retains generality and does not make any strong assumption on the underlying channel model, as opposed, for instance, to [145] which focuses only on IEEE 802.11p. Third, it considers the possibility of multi-hop routing via mmWave thanks to graph modelling. Among the related works proposing VEI models and algorithms, it is worth mentioning again the one by Higuchi *et al.* [144]. The authors define a task completion probabilities which are shared between vehicles. Tasks are then offloaded based on deadlines. Although this represents a very valid approach, their model differs from VEIP, as our model attempts to explicitly minimize the task completion latency, keeping the available resources as constraints, considering that, in vehicular networks, most tasks are safety critical and should be executed as soon as possible.

For each DL task, the OM needs to perform three main decisions:

1. which location tasks should be offloaded to, i.e., either other vehicles or the cloud;
2. how many resources should be reserved on the same vehicles (or requested to the cloud);
3. only if the tasks are splittable, which fraction of each task should be assigned to each destination node.

We define as *sources* all vehicles that may offload tasks to other nodes, i.e., other vehicles or the cloud. Through the shared knowledge built thanks to the DSRC link, each source is supposed to be aware of the full mmWave network topology. At a given time slot, several tasks may need to be fulfilled by each source i.

We define as f_i the number of computations needed for the tasks of each source i with respect to a given reference system (e.g., a given platform with a certain CPU and GPU). Furthermore, we assume no constraints on the available RAM and disk in the destination nodes.

Several input quantities are then defined, all referring to a single time slot:

1. $N = \{1, ..., n, k\}, |N| = \nu$, set of all the available n connected nodes; beside the connected vehicles, k is a special node modelling the access to the cloud, which can be accessed from all the vehicles.
2. $S \subseteq N, |S| = \xi$, set of all the *source* vehicles generating (and possibly performing) tasks at a given rate.
3. $E = \{(w, z) : d_{wz} < d_{lim} \wedge RSSI_{wz} \geq RSSI_{lim}\} \cup \{w, k\}, 1 \leq w \leq n, 1 \leq z \leq n$, set of edges between nodes, i.e., active mmWave links with a good-enough signal. Furthermore, d_{lim} is the distance limit above which any mmWave link is considered unstable, while $RSSI_{lim}$ is the RSSI limit below which any mmWave is considered unstable.

4. $G = (N, E)$, graph of the current network topology, whose edges are represented by valid and stable mmWave links.

5. $C_c = \{c_1, ..., c_\nu\}$, set of the currently available computational capacity of each node, in terms of computations per second, with respect to a given reference system, as defined for f_i. The computational capacity is a function of the available CPU and GPU for each node j: $c_j = \alpha(CPU_j, GPU_j)$.

6. $\mathbb{R}_{ij} = \{(i, h_1, ..., h_z, j) : (i, h_1) \in E, (h_z, j) \in E, (h_i, h_{i+1}) \in E, 1 \leq i \leq z - 1\}$, set of all routes from each source i to any possible destination $j \in N$ in the network.

7. $L(i, j)$, latency of the route from source i to destination j. Assuming a symmetric channel, the Round Trip Time (RTT) can be represented by $2 \cdot L(i, j)$. This term depends on the amount of data D_{ij} which needs to be transmitted from each source vehicle to each destination node, and on the average wireless channel data rate b_{ij}. In turn, D_{ij} depends on the actual task fraction assignment to each destination node: $D_{ij} = \beta(\tau_{ij})$, where β maps the task fraction to each destination node to the amount of data which needs to be sent to that node.

8. o_j, overhead time of each destination node j. This time accounts for the overhead due to the message reception in the target operating system, data encoding and decoding and all the operations which do not depend on the size of the actual task.

We also define a set of output quantities for each time slot, coming from the solution to the optimization problem:

1. $A_i^* = \{a_{i1}, ..., a_{i\nu}\}$, set of optimal resource assignment to each node in the network for the current source i. All the nodes j such that $\exists(a_{ij} > 0)$ will be destination nodes towards which the tasks are offloaded. Each a_{ij} is defined in terms of computations per second.

2. $D \subseteq N$, set of selected destination nodes to which the tasks should be offloaded.

3. $T_i^* = \{\tau_{i1}, ..., \tau_{i\nu}\}$, set of optimal task subdivision to each node in the network for the current source i. Each τ_{ij} represents the fraction of task f_i assigned to the destination node j. Each destination node $j \in D$ should have at least one non-zero τ_{ij}, while each non-destination node j should have all its $\tau_{ij} = 0$.

4. R_{ij}^*, set of selected optimal routes from each source i to the chosen destinations j.

The VEIP problem is then modelled as follows:

$$\text{Find} \quad D \subseteq N : \{1 \leq j \leq \nu : \exists(a_{ij} > 0), 1 \leq i \leq \xi\} \tag{3.1a}$$
$$\mathbb{R}^*_{ij} \subseteq \mathbb{R}_{ij}, 1 \leq i \leq \xi \tag{3.1b}$$
$$\mathbb{A}^*_i, 1 \leq i \leq \xi \tag{3.1c}$$
$$\mathbb{T}^*_i, 1 \leq i \leq \xi \tag{3.1d}$$

such that:

$$\text{minimize } i, j \quad \sum_{i \in S} \sum_{j=1}^{\nu} \{2 \cdot L(i,j) + \frac{\tau_{ij}}{a_{ij}} + o_j\} \cdot y_{ij} \tag{3.1e}$$
$$\text{subject to:} \quad \sum_i a_{ij} \leq c_j, \forall j \tag{3.1f}$$
$$\sum_j a_{ij} > 0, \forall i \tag{3.1g}$$
$$y_{ij} = \begin{cases} 1 \text{ if } a_{ij} > 0 \\ 0 \text{ otherwise} \end{cases}, \forall i, j \tag{3.1h}$$
$$\sum_j \tau_{ij} = f_i, \forall i \tag{3.1i}$$
$$\tau_{ij} \leq M \cdot y_{ij}, \forall i \tag{3.1j}$$
$$\tau_{ij} \geq \tau_{min} \cdot y_{ij}, \forall i \tag{3.1k}$$

The selected destination nodes and routes from sources to destinations are defined by the sets (3.1a) and (3.1b). The objective function (3.1e), instead, aims at minimizing the overall latency and it is composed of multiple terms:

1. the RTT of the path towards the destination node;
2. the computation latency, defined as the ratio between the number of computations needed to fulfill tasks from source vehicle i and the resources (number of computations per second) assigned to the destination node j for the tasks of source node i;
3. the overhead time, as defined earlier.

Constraint (3.1f) is the capacity constraint, ensuring that we cannot assign to a node more capacity than its current availability, while Constraint (3.1g) forces each task to be executed.

Furthermore, the support variable y_{ij} is defined to take into account that a node j with all its $a_{ij} = 0$ is not being chosen as a destination node, and thus should not contribute to the overall latency.

Constraint (3.1i) forces instead each task from source i to be completely fulfilled by the selected destination nodes j. Finally, Constraints (3.1j) and (3.1k) make

each destination node j with $a_{ij} > 0$ perform at least a fraction of task τ_{ij}, and each non-destination node j with $a_{ij} = 0$ perform no computations for the current task f_i. In our notation, M represents an upper bound to the value of each τ_{ij}, while τ_{lim} is a lower bound to the value of each τ_{ij}.

After formulating VEIP and its mathematical optimization model, we demonstrate that it is NP-Hard and propose an efficient and lightweight greedy solution which can be employed by the OM thanks to the development of proper open source software.

Theorem 1. *The proposed problem (VEIP) is NP-Hard.*

Proof. The problem can be reduced to a Multiple Knapsacks problem under the following assumptions:

1. Task are allowed to be unfulfilled (this removes Constraint (3.1g)).
2. A single hop problem is considered, in which there is no propagation nor transmission and overhead time, i.e., $L(i,j) = 0, o_j = 0, \forall i,j$.
3. The problem is now supposed to be unsplittable, enabling us to replace τ_{ij} with f_i (only one destination node can be selected to fulfill the whole task) and remove constraints from (3.1i) to (3.1k).
4. The same computations per second are required to each destination node to fulfill tasks from each source i. Each a_{ij} can thus be rewritten as $x_{ij} \in \{0,1\}$, since the problem now becomes finding which destination could fulfill our task. x_{ij} will be thus equal to 1 if a given destination j is going to fulfill tasks from source i.

Given the four assumptions mentioned earlier, the problem can then be rewritten as $\max \sum_j^{\nu} \sum_i^{\xi} (-f_i \cdot x_{ij})$ subject to $\sum_i f_i \cdot x_{ij} \leq c_j, \forall j$. This is a Multiple Knapsacks Problem, which is NP-Hard, thus also VEIP is NP-Hard. □

3.6.4 DG-VEIP: a greedy solution to VEIP

As the VEIP problem proved to be NP-Hard, the OM can employ an efficient Greedy Algorithm to solve it. We thus proposed a distributed VEIP algorithm, here referred to as *DG-VEIP*.

As an assumption, we consider the cloud as always reachable and available. Furthermore, DG-VEIP requires $L(i,j)$ to be properly set depending on the scenario and on the set of tasks, and it is designed to be executed for each source vehicle i.

The algorithm pseudocode is reported in *Algorithm 1*.

At each time step t_i, each source vehicle i generates a number of tasks to be executed. The number of total computations needed f_i is used to represent these tasks.

Algorithm 1 Distributed Greedy VEIP Algorithm (DG-VEIP)

Define an empty list of vehicles V
Define the remaining task fraction to be assigned $\tau_{rem} \leftarrow f_i$
for $n \in N$: path between i and n exists in $\mathbb{R}_{ij}$ **do**
    if $n \neq k$ **then**
        Compute $\frac{d_n}{c_n}$
    else
        $\frac{d_n}{c_n} \leftarrow \infty$
    end if
    Add node to list V, ordered by ascending $\frac{d_n}{c_n}$
end for
for $v \in V$ and $\tau_{rem} > 0$ **do**
    if $c_v > 0$ **then**
        Request all available capacity to the node: $a_{iv} \leftarrow c_v$
        if $v \neq k$ **then**
            $c_v \leftarrow 0$
        end if
        Assign task fraction τ_{iv} such that it is completed before or in a t_i time
        $\tau_{rem} \leftarrow \tau_{rem} - \tau_{iv}$
    end if
end for

Thanks to the enhanced LDM, each vehicle can scan the list of the N connected nodes (which can include the vehicle itself at $d_n = 0$), and add them to a new list V, which is sorted in ascending order by the ratio of the distance and the available node capacity (which can be referred to as *cost*). The cloud node is artificially assigned the highest ratio among all other nodes (ideally ∞), so that it will be selected last, only if needed. The algorithm then loops over all the nodes in V until all the f_i computations have been offloaded (or performed by itself or by the cloud). The variable τ_{rem} represents the number of remaining computations which need to be offloaded. If a node in V is found with free computation resources, the source vehicle will require all its capacity, to try to minimize the latency, and assign a task fraction τ_{iv} such that the task is either completed in a t_i time, meeting the deadline of the next time frame, or before that time (if the node has more computation resources available). Finally, it should be mentioned that the cloud is supposed with no hard resource limits, and it can thus be always selected as last possibility when no other local vehicles can be selected.

DG-VEIP has been implemented in a dedicated MATLAB function, which has been exploited to evaluate its usage within Edge-V and which is going to be become publicly available, on GitHub, under the GPLv2 license.

3.6.5 Proof-of-Concept and prototype

Edge-V has been realized and tested in the field thanks to the design and developed of a full-fledged prototype. The high-level overview of our Proof-of-Concept is depicted in Figure 3.23.

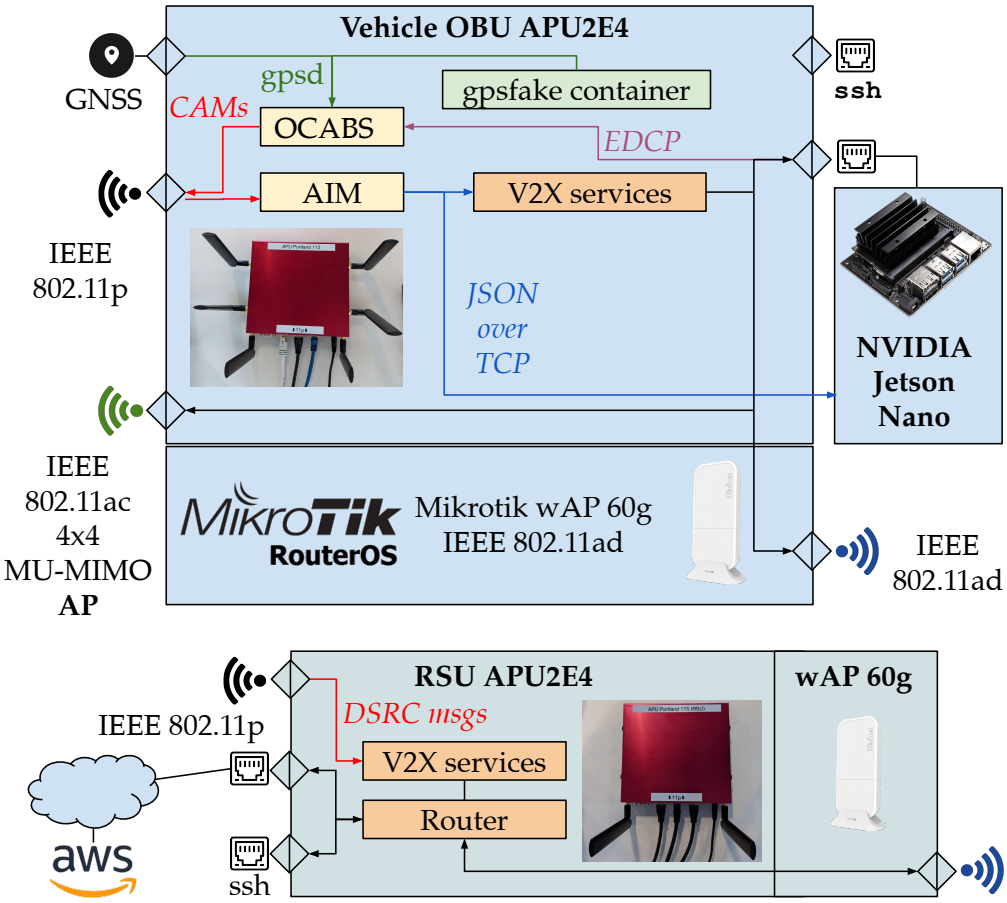


Figure 3.23: The proof-of-concept prototype of Edge-V.

As can be seen, we prototyped Edge-V inside an RSU Road Side Unit (RSU) and multiple OBUs. In particular, we realized three OBUs, to be deployed on up to three vehicles, and one RSU, which we then leveraged to evaluate Edge-V on the field and in a laboratory environment.

As mentioned, Edge-V is characterized by its openness. Therefore, all the key features of our framework have been realized in low-cost, commercially-available hardware, and with well-documented open source software components.

Concerning the unlicensed spectrum access technologies, we selected three IEEE 802.11-based standard, i.e., IEEE 802.11ac for the 5 GHz Wi-Fi internal connectivity, IEEE 802.11p for the DSRC link, and IEEE 802.11ad for mmWave at 60 GHz.

Hardware

Concerning the hardware, the prototype hinges upon the open DSRC platform described earlier in this Section, running the latest version of OpenWrt-V2X (i.e., 21.02) [20]. As in the OBU project, we upgraded the hardware to the APU2E4 boards. Each board has been enhanced with the installation of:

- An Atheros AR5B22 mPCIe wireless module for the IEEE 802.11p interface. This module relies on the same chipset (AR9462) as the UNEX DHXA-222 cards, and it is characterized by the same performance and maximum selectable transmission power (18 dBm). The choice of leveraging slightly different modules was due to the unavailability of the UNEX cards at the time of development.
- A Compex WLE1216V5-23 Multi-User Multiple-Input and Multiple-Output (MU-MIMO) 4x4 IEEE 802.11ac mPCIe card, providing a relatively high maximum transmission power (up to 29 dBm with MCS 0). As this module requires a 5V additional power supply, we used mini PCI express extender cards, which have been installed in one of the APU2E4 miniPCIe slots and on which we soldered a jumper cable. The cable has been soldered, in particular, to the reserved pins of the extender cards (45, 47, 49, 51), which can also be used to provide an additional power supply to the Compex chips. It has then been connected to the 5 V pin available on the APU boards. This setup is depicted in Figure 3.24.

Each APU2 board (both OBUs and RSU) is connected to a MikroTik wAP 60G [146], through one of the available Ethernet ports. These devices are IEEE 802.11ad routers working in the 60 GHz unlicensed spectrum and equipped with 6x6 planar phased antenna arrays, covering an angular range of 60 degrees. Since these devices are still unable to establish direct peer-to-peer links, we deployed an additional MikroTik wAP 60Gx3 Access Point, to which all the devices in the testbed are connected. This strategy makes the client devices appear as if they are directly connected together. Finally, since the APU2 boards do not embed a GPU for executing DL tasks, we interfaced each board (except the RSU one) with an Nvidia Jetson Nano Development Kit. The Nvidia Jetson Nano can be exploited for GPU computation and can be easily connected to an APU2 board through a Gigabit Ethernet point-to-point link.

Software

The European ETSI ITS-G5 set of standards has been taken as reference for the implementation of the enhanced wireless stack. Therefore, CAM messages are used to periodically broadcast, via IEEE 802.11p, kinematic and dynamic vehicle information such as speed, acceleration, position and heading. However, there

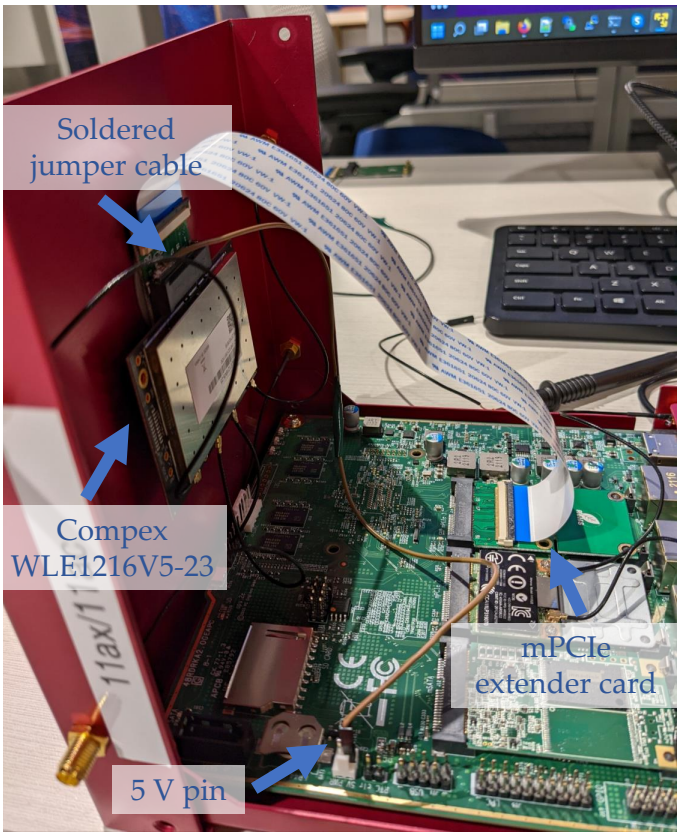


Figure 3.24: APU2 board hardware setup for the installation of a high-power Compex WLE1216V5-23 4x4 IEEE 802.11ac MU-MIMO mPCIe module.

are currently no standardized messages for the exchange of channel-related and load-related metrics, as required by Edge-V. This data is of utmost importance to enable the next generation edge intelligence use cases, which require offloading and local computation to reduce latency. Hence, we designed an additional optional container (the *Channel and Node Status Container*) which can be inserted inside standard CAM messages, enhancing them with the information needed to enable VEI and task offloading. Our proposed container has been defined by upgrading the standard CAM specifications, which are written in the ASN.1 description language [63]. Starting from the ASN.1 definition, it was then possible to generate the code for the encoding and decoding functions thanks to the *asn1c* tool [70].

The additional container is schematized in Figure 3.25. The grey text corresponds to optional fields, while the black text to mandatory fields. Each mandatory field can be set to a special value corresponding to *Unavailable*, if needed, to ensure the maximum flexibility from the usage of our container. The *Extra Device* load information is thought to carry the node load information of any additional computation device connected to the main computing unit, such as our NVIDIA

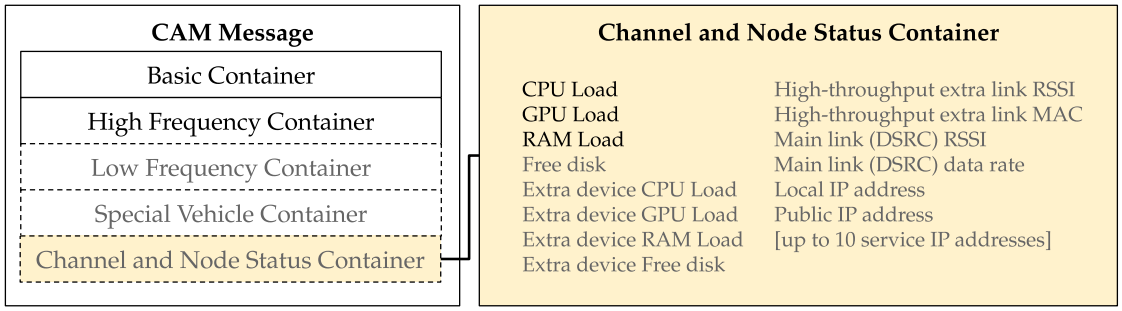


Figure 3.25: Enhanced CAMs to enable sharing of node and channel load information.

Jetson Nano. The *extra link* refers instead to any additional, high-throughput, radio channel available in the main computing unit, such as our mmWave link. The enhanced CAM ASN.1 specification file is going to be released with an open source license.

In addition to the definition of enhanced CAMs, we enhanced the open source software currently available as part of the OBU Project:

- The Open CA Basic Service (OCABS) has been upgraded to support and generate enhanced CAMs with the optional *Channel and Node Status Container*. In order to properly encode and disseminate CAMs, OCABS needs to be fed with GNSS data. This data can come either from a GNSS device connected to the APU2 boards via USB (as in our OBU "Alpha" prototype), or from pre-recorded traces, which are replayed in a dedicated container thanks to tools like `gpsfake`.

- The Automotive Integrated Map (AIM) has been upgraded to receive and decode enhanced CAMs, and store the information contained in the *Channel and Node Status Container* into the local database. The upgraded AIM thus implements the Edge-V enhanced LDM. As in the case of "vanilla" AIM, V2X services can request the needed information (including the VEI-specific data) through a JSON-over-TCP interface.

Finally, a novel lightweight protocol, encapsulated inside UDP at the Transport Layer, has been defined to transfer with low latency CPU, GPU and RAM usage of the Nvidia Jetson Nano to the upgraded OCABS. This custom protocol, called Extra Device Communication Protocol (EDCP) is based on a client-server paradigm. Each Nvidia Jetson Nano runs a server as a service, waiting for requests from OCABS (acting as EDCP client). Every time a request is received, it is parsed and a reply is immediately generated, containing the resource usage information. This reply is then sent to the EDCP client (i.e., OCABS) running on the APU2 board.

3.6.6 Performance evaluation

Thanks to realization of our prototype, it was possible to evaluate Edge-V both in a laboratory environment and on the road with two real vehicles and one RSU. As introduced earlier, two significant use cases have been investigated:

1. Direct data exchange between vehicles with high throughput and low latency;
2. DL-based object detection in a VEI scenario.

Before presenting the results of the tests with the prototype, we have also evaluated Edge-V through trace-based simulations, considering an OM implementing DG-VEIP.

Simulation with realistic vehicular traces

An ad-hoc MATLAB simulator, integrating a function with DG-VEIP, has been developed to simulate a vehicular communication system starting from the SAMARCANDA dataset [7]. This dataset, in its CSV version, comprises the traces of 19 real vehicles travelling in an area around Pinerolo, Italy. More details on this open dataset are available in Section 5.1.7. All vehicles are supposed to be able to reach the cloud through a proper deployment of RSUs in the simulated scenario.

Consistently with the results of an extensive field test campaign, involving mmWave at 60 GHz and presented in Section 4.3, the mmWave latency has been set to 0.7 ms, while the overhead time has been considered as negligible ($o_j \sim 0$). We also performed several measurements from our laboratory towards an Amazon AWS virtual machine, with the aim of configuring a realistic cloud latency in the simulator. This led to a Generalized Extreme Value latency distribution with $\sigma = 6.89932\,ms$, $\mu = 64.5928\,ms$ and $\xi = 0.11209$.

Furthermore, vehicles are connected in a stable way if the distance is lower than 140 meters (i.e., $d_{lim} = 140$), consistently with the Edge-V road tests described in Section 3.6.6. To analyze a high-load network scenario, each vehicle has been considered as both a source and possible destination node, where each source always offloads its own tasks, without fulfilling them by itself.

Figure 3.26 plots the comparison of our DG-VEIP algorithm with respect to a cloud-only baseline, with different values of task size f_i and by assigning a random remaining capacity to each vehicle at each time frame between 0 and a maximum capacity value. A time span of 1 second has been considered as maximum task deadline.

As can be seen, our VEI solution is much more effective when large tasks are being offloaded, such as images for object detection. For instance, a set of tasks of size $f_i = 50$ can be executed within the deadline by Edge-V, as opposed to a cloud-only approach when cloud capacity is 45 computations/s. Although the results could be improved with more advanced optimization techniques, it should be noted

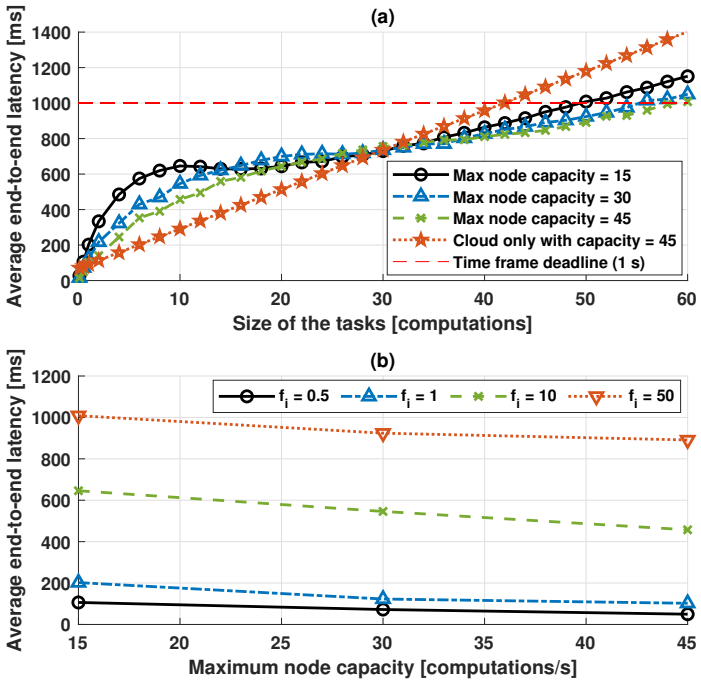


Figure 3.26: Average end-to-end latency of the DG-VEIP as a function of (a) task size, (b) maximum On-Board Unit computational capacity.

that the obtained trend should retain generality, with large tasks benefiting more from local computation than smaller tasks. Furthermore, the results show how an increase in the maximum local vehicle capacity can lead to an improved overall latency in a VEI approach.

Vehicular Data Exchange use case

The first use case generalizes several automotive applications requiring a low-latency direct exchange of data, such as video streaming, *See Through*, online gaming, and many more. To this end, two laptops are associated with the IEEE 802.11ac access point generated by the respective APU2 boards. We performed several latency and throughput tests to evaluate the performance of our framework, through the prototype hardware. Both indoor (laboratory) and outdoor (road) tests have been performed.

Concerning the indoor tests, they have been performed in our laboratory, with the LTNT framework software, after its deployment and proper configuration on the APU2 boards. Thanks to LTNT, it was possible to reliably measure RTT and throughput between two OBU boards (i.e., between two vehicles in a static scenario, with the aim of performing a baseline assessment). The results are depicted in

Figure 3.27.

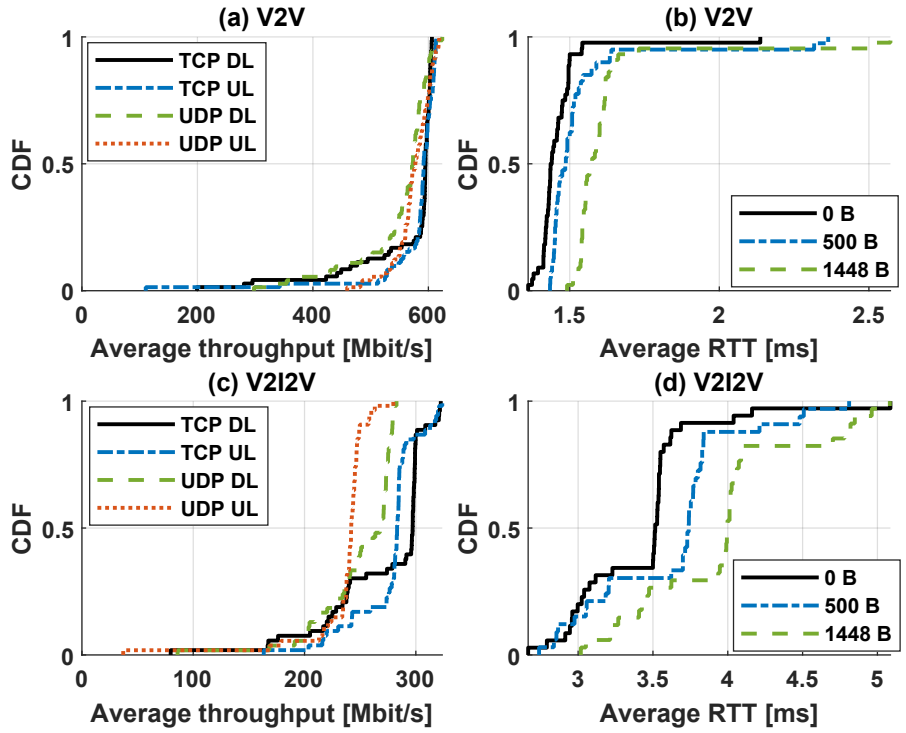


Figure 3.27: CDFs of the mmWave latency and throughput during a one-day-long test. (a) V2V throughput (b) V2V latency (b) V2I2V throughput (through RSU) (d) V2I2V latency.

The plots show the Cumulative Distribution Function (CDF) of throughput and RTT, measured between two APU boards, in a direct mmWave Vehicle-to-Vehicle communication (top plots), and between the same boards through the RSU, emulating a longer-range V2I2V scenario, in which the infrastructure node acts as message relay (bottom plots).

As can be seen, the overall RTT always remains below 5 ms, with an average of around 1.4 ms for the direct communication case (V2V) and 3.7 ms for the relayed communication (V2I2V). This proves that mmWave is suitable for a very low-latency high-throughput scenario. Indeed, it reaches more than 500 Mbit/s when using both TCP and UDP. This value, even though it could actually increase without the extra step through the mmWave AP, shows once more the advantages mmWave could bring to the next generation of automotive use cases.

Concerning instead the outdoor tests, with two real vehicles and one RSU, they have been performed on a straight stretch of road near Scarborough, Maine, USA, allowing us to test the effect of distance on the mmWave links up to 240 m. To this end, we developed specialized Python scripts which are able to output synchronized network performance (i.e., latency and throughput) and relative distance

information, thanks to the reception of data from a GNSS receiver. These scripts make use of both our LaTe tool and iPerf 3. The main aim of the road tests was to evaluate our POC on the field, where real environmental conditions are expected to affect the communication.

As mentioned earlier, we equipped two vehicles with the needed OBU hardware and a GNSS receiver (with 10 Hz update rate) and set up a fixed RSU, as depicted in Figure 3.28.

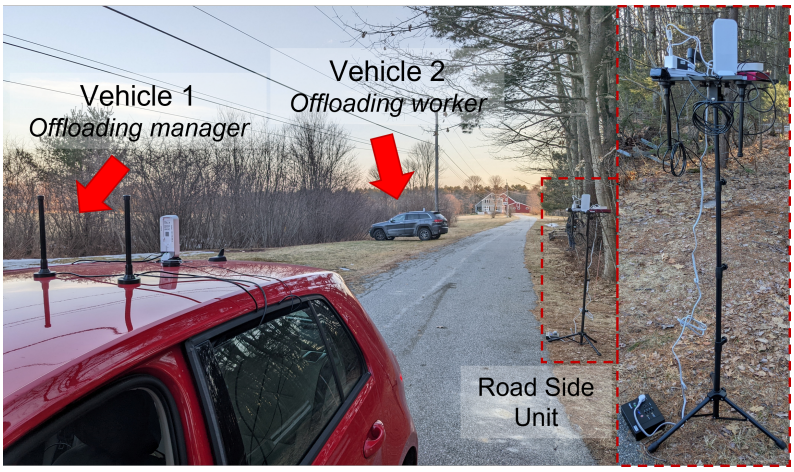


Figure 3.28: Road tests experimental setup.

One vehicle (*Vehicle 1*) was moving along the road, and one vehicle (*Vehicle 2*) was parked off the road, and while the RSU was placed 32.6 meters in front of *Vehicle 2*. We measured the effect of distance on the capability of Edge-V to provide a high-throughput, ultra low-latency communication between two on-board devices. As mentioned earlier, the laptops located in two different vehicles are able to directly communicate thanks to the IEEE 802.11ac access points located inside each vehicle, and to the mmWave links established between the different nodes (i.e., vehicles and RSU). We focused our analysis on the UDP throughput, as UDP appears to be more suitable than TCP to vehicular scenarios, and on the RTT with a UDP payload of 524 B (corresponding to a 500 B LaMP payload).

Both a direct V2V communication, and communication through the RSU (realizing a V2I2V communication) has been tested. The first scenario (V2V) has been realized by mounting the mmWave AP on *Vehicle 1*, while the second scenario (V2I2V) was configured by moving the mmWave AP from *Vehicle 1* (which was consequently equipped with a client) to the RSU installation. The V2I2V communication reflects now a real relayed communication, as opposed to the indoor tests, thanks to the AP being now directly connected to the RSU APU board.

The most significant results are depicted in Figure 3.29.

As can be seen, the obtained throughput and RTT results are in line with the

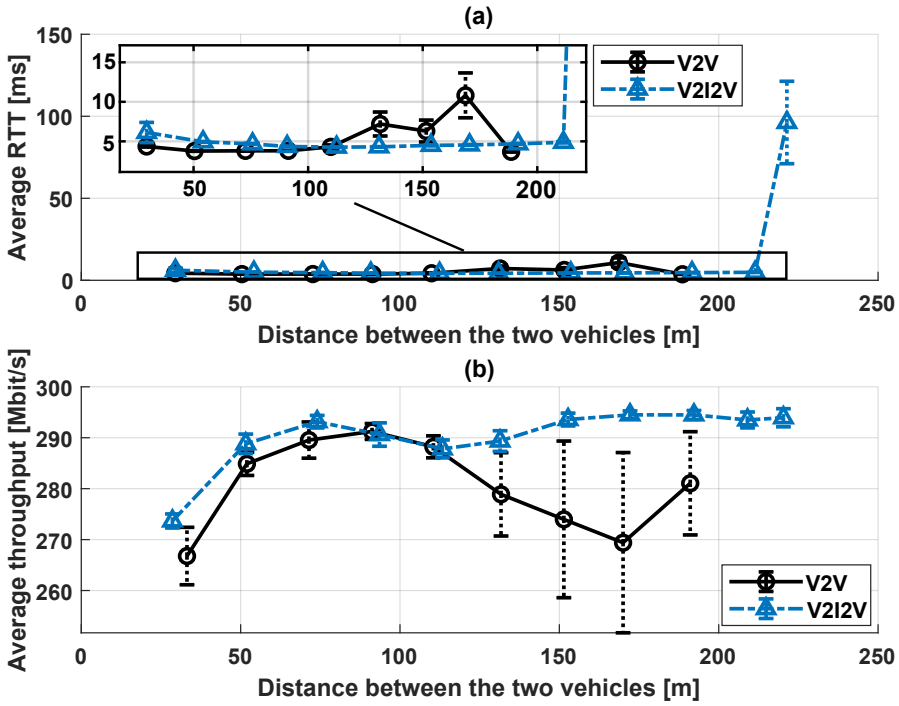


Figure 3.29: RTT and throughput as a function of the distance between two vehicles; each point corresponds to a one-minute-long test, with 95% confidence intervals. The plots show: (a) Average UDP throughput, (b) Average RTT between the on-board devices.

laboratory measurements, considering the addition of two IEEE 802.11ac links between the devices (one inside each vehicle), besides the mmWave communication. The results show how Edge-V is able to provide a very stable RTT, on average lower than 5 ms, up to a distance of 110 m. Then, the V2V communication starts to experience an increased instability when the distance becomes greater than 130 m, even though the RTT mostly remains lower than 10 ms. These values are very good, as they are compatible with the requirements of even the most demanding automotive applications. Indeed, as mentioned in Section 3.3, ETSI defined a maximum end-to-end latency of 50 ms for safety critical applications [7]–[9], which was reduced to 10 ms by the 5G-CARMEN project for highly automated centralized maneuver management.

The relayed V2I2V communication, through the RSU, is instead able to provide stable values up to around 220 m. Then, 230 m represents a limit distance, since the RSU is placed 32 m ahead of *Vehicle 2*. The RTT below 100 m is overall slightly higher, but the results show how a V2I2V communication can effectively provide an extended range mmWave communication, thanks to the deployment of one or more RSUs.

Similar considerations can be held for the measured throughput, which is always above, on average, 270 Mbit/s, thanks to the combination of IEEE 802.11ac and mmWave. Combining these values with the ones measured indoor without the IEEE 802.11ac links (i.e, more than 500 Mbit/s), it is possible to prove how the overlying mmWave network can provide enough bandwidth to accommodate multiple on-board devices on each vehicle.

The results also show how a distance between 130 and 150 m can be considered as a *safe threshold* for a stable and high quality communication. Indeed, the measured values become less stable when the relative distance between the two vehicles, in a direct V2V communication, becomes greater than 130 m. Furthermore, the small throughput reduction for relatively short distances is likely due to the angle between the mmWave devices. Indeed, our devices have a limited angular range (i.e., 60 degrees) and, for shorter distances, the effect of slightly different angles in the device placement becomes more evident.

Finally, it should be highlighted that all the RTT and throughput values have been collected while the vehicles are exchanging enhanced CAMs through the 5.9 GHz DSRC link to populate their LDMs. However, both the internal Wi-Fi and mmWave were unaffected thanks to the usage of different unlicensed spectrum bands.

Task offloading use case

The second use case showcases a DL task offloading application enabled by Edge-V. As in the previous case, it has been tested both in indoor (in-lab) and outdoor (road) scenarios.

It should be recalled how task offloading can be beneficial both to avoid expensive hardware OP, and to make vehicles able to manage a significant number of tasks in parallel, which could exhaust the resources available on-board. The less safety critical tasks (but still with strict latency and throughput requirements) can be thus offloaded to other vehicles (or, if needed, to the infrastructure) that provide free resources.

With the aim of testing DL task offloading, an object detection system based on the Microsoft Common Objects in Context (COCO) dataset [147] has been developed. The system enables a vehicle, each time, to offload either to other vehicles or the cloud. Three OBUs implement the vehicular edge, while a Virtual Machine (a *t2.2xlarge* Amazon AWS instance with 8 virtual CPUs and 32 GB of RAM) has been adopted to implement the cloud. The latter can be reached either through our laboratory network, concerning the indoor tests, or thanks to an LTE link through the T-Mobile network, concerning the outdoor tests. The OBUs rely on a less accurate, but less computationally expensive, object detection model, namely *Faster R-CNN Large*, with MobileNet V3 backbone [148] and run on the Jetson Nano boards. On the other hand, the cloud employs YOLOX-s [149], which

is more computationally expensive but also more accurate.

With the aim of testing task offloading with our prototype, we realized an implementation of the Offloading Manager. In particular, we developed two components:

- An *Object Detection Offloading Manager*, running on one OBU. It includes a full COCO dataset to emulate frames coming from an on-board camera, on which object detection needs to be performed. As the aim is to implement an Offloading Manager based on the VEIP model, the decisions need to be taken to minimize the overall latency. Therefore, this component will select the best destinations for each offloaded task based on the information available inside AIM, including available resources on the target, distance from the target and RSSI of the mmWave channel to target. If no vehicles with enough free resources or near enough are available, offloading is performed to the cloud through the RSU. To this aim, the Object Detection Offloading Manager implements a simplified version of DG-VEIP, in which offloading to another vehicle occurs only if it is within a certain mmWave RSSI and distance (i.e., if the mmWave links can guarantee a stable connectivity and a low $L(i, j)$) and if it has enough resources available (i.e., if it satisfies the VEIP Constraint (3.1f) in the case of one destination node only).

- An *Object Detection Offloading Worker*, running on the other OBUs and on the cloud. This component represents the implementation of a V2X service waiting for frames from other nodes, on which object detection should be performed. When frames are received from the Offloading Manager, it performs the inference, and then return the detection results to the sending node, in a JSON format.

Concerning the indoor tests, they involved all three OBUs and the RSU, connected to the laboratory network, and, in turn, to the Amazon AWS Virtual Machine. The results of a test session on the full COCO dataset (i.e., 5000 images, in the selected version) are shown in Figure 3.30. The plots depict the average end-to-end task offloading and object detection latency, and the mean average precision, as a function of the frame offloading frequency, which has been varied from one image every 1.6 s (i.e., 0.625 Hz) to 10 images per second (i.e, 10 Hz). A cloud-only approach has been considered as a baseline. With the aim of providing comparable results, we measured the time needed by the cloud to perform inference on the Nvidia Jetson Nano model and assigned that computing time, for each image, to each actual inference on the Nvidia boards, instead of considering the embedded board computing times. This is technically sound, as actual vehicles are expected to provide much better computation capabilities.

The results show that offloading to nearby vehicles helps noticeably reducing the overall latency, at the expense of a slight precision reduction. Indeed, offloading to nearby vehicles can help to reduce the end-to-end latency of more than 150 ms, up

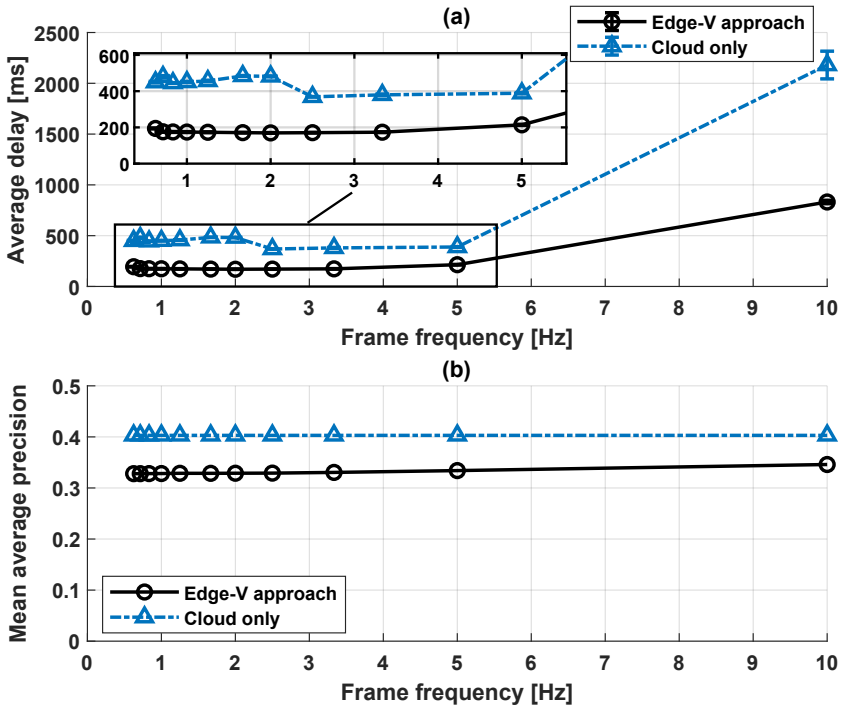


Figure 3.30: (a) End-to-end processing and network latency and (b) Overall mean average precision accuracy of the object detection, as a function of the image generation frequency.

to 5 Hz. The best result is achieved at 1.67 Hz, with up to 286 ms latency saving, corresponding to a 59% improvement with respect to a cloud-only baseline. This comes at a reduction of the precision from 0.403 to around 0.328, which is much less impacting that the actual latency reduction.

Furthermore, frequencies higher than 3 Hz are characterized by a slight increase in average latency in the vehicle edge (*Edge-V approach*), with a corresponding accuracy increase. This is caused by the fact that each OBU, in our testbed, is able to perform inference only on one frame at a time, due to the resources available in each Nvidia Jetson Nano. The offloading manager may find that no vehicle is available for offloading when the frequency is high enough, thus automatically sending that frame to the cloud, which causes a higher latency but provides a slightly better accuracy.

Concerning the road tests, we considered the same setup as depicted in Figure 3.28, with two vehicles (i.e., two OBUs) and one RSU. As shown, the Object Detection Offloading Manager was deployed on *Vehicle 1*, while the Object Detection Offloading Worker has been launched on *Vehicle 2*.

The most significant results are reported in Figure 3.31, showing the average end-to-end latency experienced by the Offloading Manager as a function of the

relative distance between the two vehicles. All the measurements are gathered with *Vehicle 1* offloading frames either to *Vehicle 2* or to the cloud, and compared to a cloud-only offloading baseline. As mentioned earlier, the Object Detection Offloading Manager implements a simplified version of DG-VEIP, with a distance limit of 140 m (to guarantee a stable mmWave connectivity, in accordance with the previous road tests) and an RSSI limit of -65 dBm. The last limit comes from the IEEE 802.11ad field test results initially presented in [12] and detailed in Section 4.3.

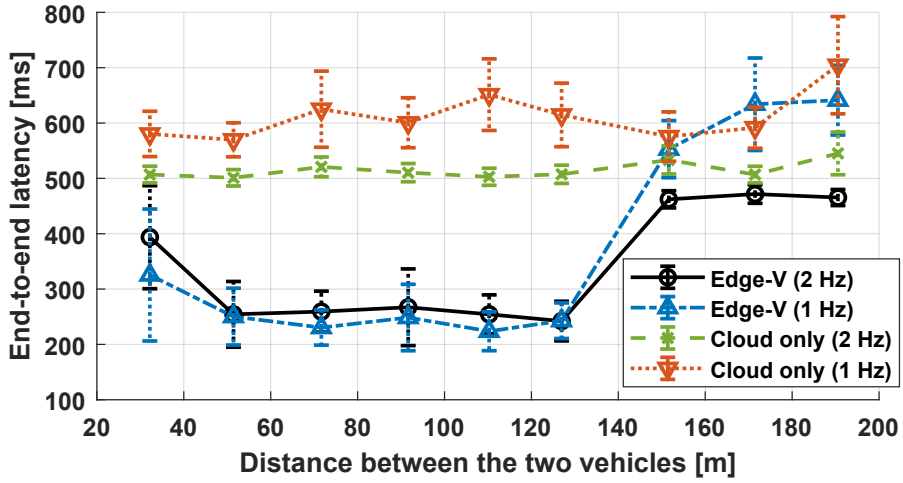


Figure 3.31: Average end-to-end object detection latency as a function of vehicle distance and task generation frequency.

As can be seen, the latency reduction with respect to the exclusive usage of cloud is significant, as long as *Vehicle 2* is available and offloading occurs locally. In particular, the maximum latency improvement was observed to be around 65% for the 1 Hz case (at around 130 m) and 52% for the 2 Hz case (at around 110 m). Moreover, it can be noticed how, when the distance is increased over 140 m, the performance decreases since no vehicular edge offloading is performed and the cloud is used. Despite the latency increase, after some tuning of the distance limit, Edge-V still continues to provide a reliable service, even when no vehicles are available or under no stable V2V mmWave coverage.

The plot in Figure 3.31 also shows an average latency which is slightly lower (up to few tens of milliseconds) when offloading frames at 1 Hz. This can be explained by the fact that the resources on *Vehicle 2* are more often busy when new frames need to be offloaded at 2 Hz, thus causing slightly more frames to be offloaded to the cloud. This effect is expected to be mitigated when more than one vehicle is equipped with the Object Detection Offloading Worker. On the other hand, the difference in the cloud case is solely due to the network architecture between the test location and the Amazon AWS Virtual Machine, which seems to perform better

when data is exchanged more often. Finally, it is worth mentioning how the same considerations reported for Figure 3.27 also apply here to explain the slightly higher latency, on average, when the distance between the vehicles is relatively short.

The results reported in this Section prove how Edge-V is able to practically enable several VEI approaches and demanding use cases. Indeed, it can provide high throughput and very low latency (lower than 5 ms round-trip in most cases), together with a seamless update of the internal Local Dynamic Maps thanks to the combination of different unlicensed spectrum technologies. The assessment of a real-life DL task offloading use case also shows the advantages of local distributed offloading, as enabled by our framework, over cloud-based approaches. Finally, our open prototype has been designed to contain as much as possible open source software, to foster research in the VEI field and let researchers easily reproduce and enhanced it.

3.6.7 Future directions

Future research directions are currently focused on performing additional field tests, with an increasing number of vehicles and in different LOS and NLOS scenarios. Furthermore, additional and more complex heuristics are currently being investigated for the deployment into the Offloading Manager.

3.7 ONIX: Open Radio Network Information Exchange

With the emergence of the MEC paradigm, which moves the computational resources to the edge of the network, latency can be noticeably reduced with respect to cloud-based approaches, together with the utilization of the network backhaul. This enables the real-time data exchange between vehicles and MEC services, to satisfy the tight bitrate and latency requirements of emerging V2X applications.

In this context, ETSI has defined a standard architecture for the provision of MEC services [150], including a key component called Radio Network Information Service (RNIS). This component is in charge of providing MEC applications with up-to-date information about the radio network, which can be of pivotal importance in safety-critical automotive use cases which rely on MEC centralized maneuver management. Moreover, fine-grained information about the channel quality could enable a new generation of AI solutions for predictive quality of service in vehicles.

As very few works in literature systematically discuss the requirements and challenges for the design of a scalable RNIS system, this thesis provides first an analysis on the challenges of RNIS, taking as reference a Collision Avoidance use case, and propose Open radio Network Information eXchange (ONIX), a scalable, fully open, RNIS architecture.

Very few works in the literature discuss the requirements and challenges underpinning the design of a scalable and open RNIS system. For instance, the design proposed in [151] is tightly coupled with a particular Radio Access Network (RAN) and core implementation and provides only little discussion on the associated design choices and requirements.

Similar solutions can also be found in the works by Nasimi *et al.* [152] and Tan *et al.* [153]. Before presenting the design of ONIX, this thesis tackles the RNIS problem in a more systematic manner, discussing first the design challenges and requirements of an RNIS platform, and then introducing our Open radio Network Information eXchange (ONIX).

ONIX is an ETSI-compliant RNIS implementation for 4G and 5G networks that provides mobile application developers with a solution that can bring the edge computing benefits to existing 4G users, while allowing, at the same time, a smooth transition towards a full 5G architecture. These benefits include not only latency reduction, as discussed at the beginning of Chapter 2, but also availability of real-time radio network information at the MEC platform.

The key strength of ONIX is, firstly, its openness and, then, its flexibility to integrate with different RAN deployment models. To achieve this goal, ONIX wraps the RAN behind a technology-agnostic interface, allowing RAN vendors to selectively provide access to UE information. ONIX has been implemented for 4G network, but it can be easily deployed in NSA and Standalone (SA) 5G networks.

ONIX is going to be released under an APACHE 2.0 License for non commercial use as part of the LightEdge platform[14].

3.7.1 The Collision Avoidance use case

It is becoming more and more evident how new markets, supported by mobile networks, are emerging. These markets include, for instance, assisted driving or remote surgery. In this rapidly changing context, platforms such as ONIX will prove to be essential to support a diverse set of users and applications.

A key example in the automotive field is represented by the Collision Avoidance use case. As described in Chapter 2.1.1, collision avoidance is a vehicle safety system designed to reduce the severity of a collision or to prevent it altogether. Basic collision avoidance systems, targeted at lower SAE automation levels, use onboard sensors to monitor the vehicle surroundings and to detect possible threats. More advanced scenarios involve instead an exchange of information thanks to V2X connectivity, such as safety messages between vehicles (e.g., CAMs exchanged via either IEEE 802.11p or LTE-V2X Mode 4) to achieve an increased awareness of the environment. Enhanced collision avoidance systems can use the mobile network to

[14] http://lightedge.github.io/.

extend the vehicle sensing capabilities beyond the range allowed by V2V communication. This is also a key concept for the development of the S-LDM MEC service described in Section 5.2.

A centralized, MEC-based, Collision Avoidance service can greatly benefit from the higher throughput enabled by 5G networks, for instance to share live camera video feeds to facilitate maneuvers around blind spots or in overtaking situations.

To this end, a scalable RNIS system such as ONIX can provide timely information about the RAN conditions to each vehicle collision avoidance system. For example, the low-level measurements provided by ONIX could be combined with AI techniques to derive high-level actionable KPIs, such as the expected bitrate or latency. This can allow each vehicle to determine if the performance of the network meets the requirements of its enhanced collision avoidance system and if this data can be used as complement to V2V and local sensor information to implement highly automated collision avoidance.

3.7.2 Requirements for RNIS

The introduction of the MEC paradigm into the mobile networking arena is blurring the line between public Internet, transport, and radio access, with changes and innovation in one sector making their way into the others. The RNIS component is a good example of this trend. It is indeed a key service specified by the ETSI MEC standard, responsible for interfacing the mobile RAN with the MEC applications. More specifically, the RNIS is responsible for collecting RAN-level information about UEs and making it accessible to MEC services.

The deployment of an RNIS-enabled MEC platform in a production network is, however, a complex process, which faces several challenges.

It is thus possible to identify the following requirements, which have been followed as guidelines for the development of ONIX:

- **5G Integration** [154]. 5G networks supports the exposure of network information and capabilities to external consumers. In this context, the MEC platform, and more specifically the RNIS component, can interact as an Application Function (AF) with the so-called 3GPP Network Exposure Function (NEF). The NEF is in place to enable selective disclosure of information to non-trusted AFs. A viable RNIS solution can thus interface with the 3GPP components to gather the necessary RAN data.

- **Forward compatibility** [155]. Since 4G networks do not specify a NEF, access to RAN data must happen through proprietary interfaces to the network management system. A practical RNIS solution must thus be able to interface with both 4G and 5G, while providing a seamless evolutionary path from the former to the latter.

- **Cloud-native support** [156]. We are currently observing a convergence between mobile technologies such as 5G and cloud technologies such as containers (like Docker containers[15]). It is thus pivotal that, sitting in-between these two worlds, an RNIS solution could satisfy the needs of MEC applications developed with cloud-native principles in mind.
- **Fine-grained RAN information access** [155]. A mobile RAN can generate a significant amount of data. A viable RNIS platform must provide applications with a fine control over the collected information at a UE level.

3.7.3 Mobile Network and MEC Host

This Section provides an overview of the main components of the ETSI MEC reference architecture [150], as depicted in Figure 3.32, and on how they fit into the 3GPP network architecture.

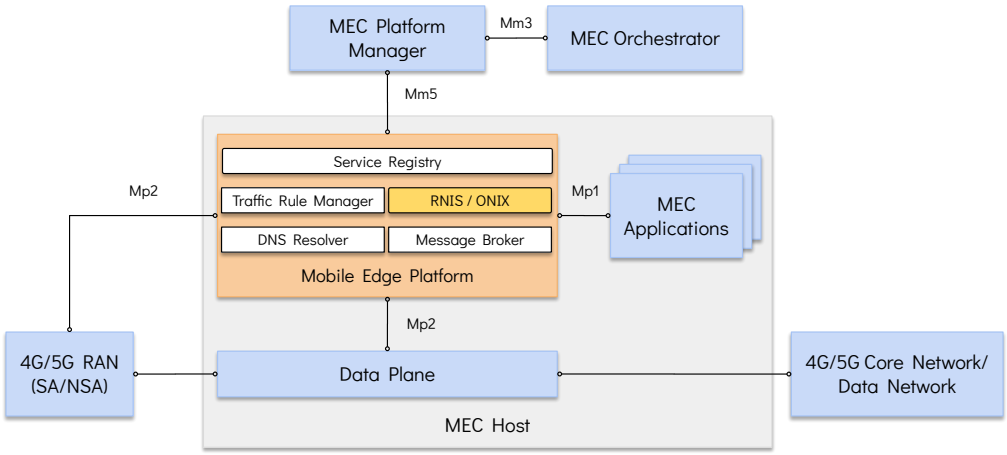


Figure 3.32: The reference ETSI MEC architecture.

ONIX takes into account the standard 3GPP RAN/core network.

It is worth mentioning that the design of ONIX is agnostic with respect to the RAN technology, as the concepts presented in this Section can equally apply to both 4G and 5G deployments. Likewise, no assumptions are made about the RAN deployment options. ONIX also supports O-RAN-like approaches, as described in the specifications by the O-RAN Alliance [157].

Furthermore, as for the RAN, ONIX makes no assumptions on the core network, which could be any of these options:

[15] https://www.docker.com/resources/what-container/

- A 4G Evolved Packet Core serving a 4G RAN;
- A 5G RAN operating in NSA mode;
- A 5G Core serving a 5G RAN.

Another important concept for the design and implementation of ONIX is the *MEC Host*. A MEC Host includes three main components: the MEC Platform, the MEC applications and services, and the programmable data plane.

The *programmable data plane* is responsible for steering the traffic between the eNB and the data network towards the MEC applications according to the rules defined by the *Traffic Rule Manager*. This goal can be achieved in different ways according to the network type. In 5G, the so-called User Plane Function (UPF) naively provides this feature, while in 4G, UE traffic can be extracted either at the Serving Gateway (SGW-LBO), at the P-GW (SGi) or using Bump-In-The-Wire (BITW) approaches (the latter is the approach followed by LightEdge). As the approach for tapping into the UE traffic and delivering it to a MEC platform is not part of the standard, several options are possible, as extensively discussed in [158].

The *MEC platform*, as defined by ETSI [61], offers an environment where *MEC applications* can discover, advertise, consume and offer MEC services, such as centralized maneuver management applications. The MEC platform is also responsible for configuring the MEC Host data plane. The main components of a MEC platform are briefly summarized below:

- **Service Registry**. It hosts the list of services and applications supported by the Platform, such as the RNIS.
- **Radio Network Information Service**. This is the core component behind the design of ONIX, which is better described in the next Section. Note that the RNIS specifications can be found in [155].
- **Message Broker**. It provides a communication channel between the components of the MEC platform. A broker implementing a protocol such as AMQP 1.0 can indeed be used as a MEC platform message collector and dispatcher, where services publish new information to a queue or topic and can, at the same time, subscribe to one or multiple queues or topics.
- **Traffic Rule Manager**. It is responsible for (re)configuring the network fabric. New traffic rules can be issued by the platform manager and then enforced by an L3 switch.
- **Domain Name System (DNS) Resolver**. It maps requests coming from UEs to addresses that are routable within the MEC domain. It is worth mentioning that any standard DNS resolver can fulfill this role.

3.7.4 ONIX System Architecture and Implementation

The ONIX reference system architecture is depicted in Figure 3.33.

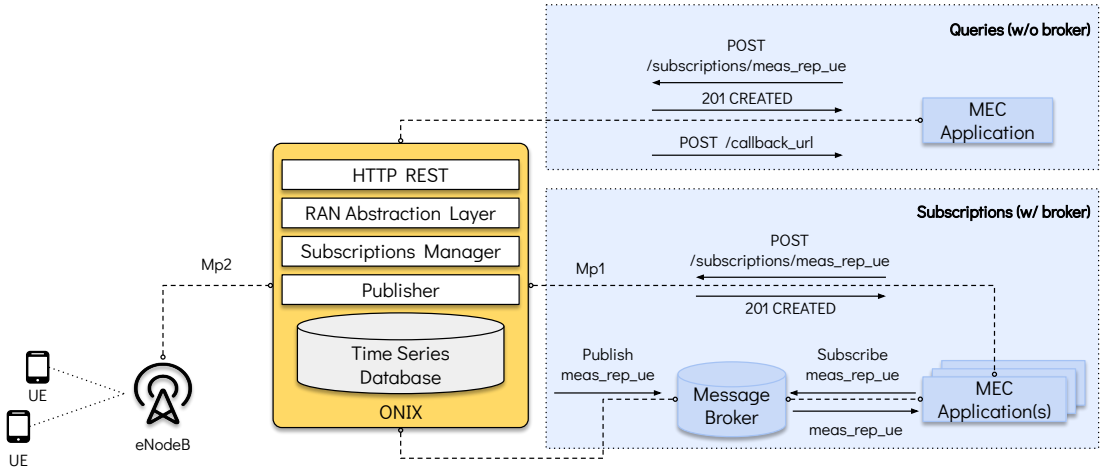


Figure 3.33: The ONIX RNIS system architecture and interfaces

The main aim of ONIX is to allow MEC applications to use the RNIS data at the network edges. Among the most relevant interfaces, ONIX connects to the 4G/5G network through the *Mp2* reference point. In particular, Mp2 is defined between the Mobile Edge Platform and the data plane of the virtualization infrastructure and it is used to instruct the data plane on how to route traffic between applications.

This reference point is not specified by ETSI since each vendor of RAN equipment has its own management interface, which is usually implemented with proprietary protocols [159]. In the design and implementation of ONIX, we are interested in the interface exposed by the control plane or by the network management system in a 4G RAN or in the interface exposed by the NEF in a 5G RAN. To the best of our knowledge, at the time of writing there are neither commercial nor open-source solutions currently supporting this interface. In ONIX, we have developed a *RAN Abstraction Layer*, which wraps the implementation details of the Mp2 interface behind a vendor-agnostic layer, to enable a smooth transition between 4G and a 5G implementation.

The Radio Network Information (RNI) can instead be gathered by authorized MEC applications thanks to the *Mp1* reference point. This reference point is defined between the applications and the Mobile Edge Platform and aims at enabling mobile edge service production and consumption. It thus tackles aspects such as authorization and authentication, service discovery, and application/service lifecycle management. It also provides other functionalities, such as traffic rules and DNS rules activation and time of day. The information exposed by the RNIS (i.e., ONIX) over the Mp1 interface falls into different broad categories, i.e., *(i)* cell changes, *(ii)*

radio access bearers establishment, modification, and release, *(iii)* UE measurement reports, *(iv)* UE timing advance, *(v)* carrier aggregation reconfiguration, and *(vi)* S1 bearers establishment, modification, and release [155].

ONIX provides two ways of accessing the RNI data, through respectively *queries* and *subscription* to a message broker (e.g., an AMQP 1.0 broker). The first has been designed for applications that do not need to access the RNIS very often. These applications can gather the RNI through a simple REST interface exposed by ONIX. The example depicted in Figure 3.33 shows a query with a REST call (`/subscriptions/meas_rep_ue`) allowing MEC applications to access specific UE measurements. The second way is instead targeted at applications which need frequent RNI data updates. Indeed, a REST API cannot efficiently manage frequent periodic updates by means of continuous POST requests. Thus, for higher workloads, ONIX supports a *Message Broker*, to which MEC applications can subscribe and that can efficiently distribute the RNI data to a high number of subscribers thanks to the *Publisher* service. The latter is the service in charge of gathering and sending the RNIS data to the broker for further distribution to all the subscribed MEC services.

The *Subscriptions Manager* is responsible for managing and granting access to MEC applications to the published information, and for terminating the connection after the subscription expires. The subscription for RNI data reception can be managed by means of several tags such as eNB, cell, and UE identifiers. This allows MEC applications to subscribe to a specific topic (on the message broker) and to filter the messages according to different criteria.

In both cases, the RNIS data is stored by ONIX in a *Time Series Database* to make it available to other applications or in general to enable further analysis. A *Time Series Database* has been selected since it represents a database category which supports re-sampling. This means that it is possible to have, for a given UE measurement, multiple time series at different time granularity. In such manner, a MEC application can subscribe to the topic whose sampling period is most suitable for its operation. Note that we expect MEC applications to select sampling periods lower than the native one to avoid being overloaded with information.

Considerations on scalability

ONIX has been designed with scalability in mind. Indeed, the *Publisher* component, as schematized in Figure 3.33 can automatically spawn and manage the lifecycle of several child processes, each of which can handle the RNI data of a certain number of UEs identified via their International Mobile Subscriber Identity (IMSI). Each process is a new instance of the *Publisher* component which, upon the reception of the channel conditions data (i.e., the RNIS data), is in charge of sending it to the message broker, acting as a producer, attaching additional metadata like per-UE sequence numbers and timestamps to be used by the applications.

The number of UEs managed by each process is a key parameter (hereinafter referred to as "UEs per process") that can be tuned according to the available resources and that has been analyzed in the evaluation presented in Section 3.7.5. The "UEs per process" parameter basically specifies how many instances of the *Publisher* component must be spawned at a given time. Notice that in a scenario featuring a substantial number of UEs, it must be also considered that each process creates a new connection with the message broker, which consequently causes an increase in the required resources. Therefore, and especially in cases including a considerable number of UEs, it is important to adequately tune the "UEs per process" parameter to the system resources. This relationship is depicted in Figure 3.34.

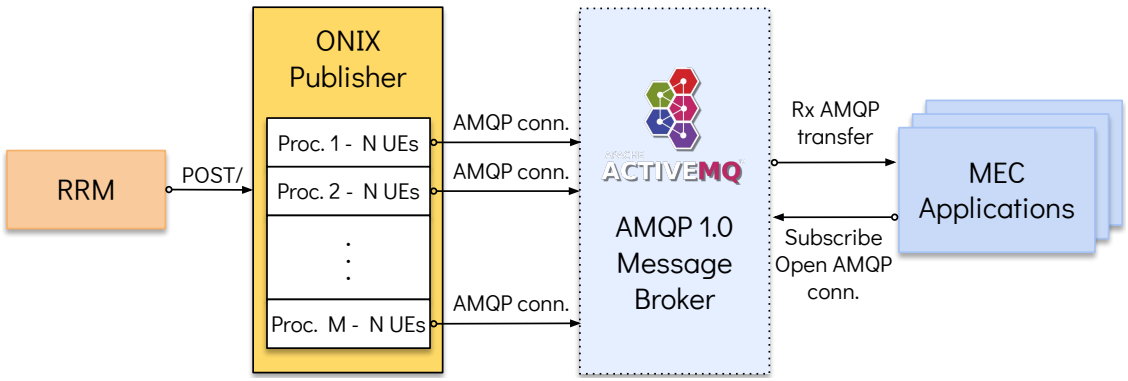


Figure 3.34: Relationship between the "UEs per process" parameter and the number of AMQP connections to the message broker. "RRM" refers to the Radio Resource Management.

ONIX prototype implementation

After the design of ONIX, we have implemented a prototype, deployed on a 4G network based on the following components:

- **RAN:** we selected a 3GPP-compliant LTE stack implemented using srsLTE (now named srsRAN), an open source software radio suite developed by Software Radio Systems (SRS) [160]. The srsRAN stack can run on dedicated SDR hardware, i.e., radios typically supporting a wide frequency range and the re-programmable implementation of the PHY layer features in software. As SDR radios, we selected the Ettus Research USRP B210.
- **EPC:** we leverage Open5GS, an open source implementation of the Release 16 5G Core and EPC [161].
- **Reference MEC platform:** we rely on LightEdge, an ETSI-compliant open source MEC framework. LightEdge is designed to provide mobile operators

with a MEC platform that can bring the advantages of edge computing to 4G users. It follows a cloud-native design since its components can be instantiated as containers and the platform itself is fully compatible with Kubernetes[16]. MEC applications and services can be deployed as containers. More information on LightEdge can be found in [162].

- **RAN control plane:** we selected 5G-EmPOWER, an open source centralized software-defined RAN controller following the control/user plane separation principles defined by 3GPP Release 14. 5G-EmPOWER implements a Radio Resource Manager (RRM) and its northbound interface enables RAN elements configuration and RAN-level statistics collection, such as UE-related Channel Quality Indicator (CQI), Reference Signals Received Power (RSRP) and Reference Signal Received Quality (RSRQ). Essentially, this northbound interface provides an implementation of the ETSI MEC Mp2 reference point. More information on 5G-EmPOWER can be found in [159].

- **Message broker:** we chose the Apache Foundation ActiveMQ platform, an open-source, multi-protocol, message broker that supports AMQP 1.0 and is optimized for scalability and resiliency [163]. As not all the AMQP 1.0 brokers are completely interoperable, the choice of ActiveMQ comes from the increasing popularity of this software, which is becoming a standard choice for AMQP 1.0 communication and is notably supported by Apache. It is also the broker of choice in V2X-related European projects such as 5G-CARMEN.

It is worth mentioning how all these components are based on open source software. Furthermore, despite the choices listed above (i.e., srsRAN and Open5GS), it must be noted that ONIX is vendor-agnostic and can be used with any eNodeB/EPC combination, including commercial ones. Furthermore, ONIX has been designed to be seamlessly integrated with existing ETSI MEC Platforms, such as LightEdge. Finally, websockets are used to exchange information between MEC applications and ONIX.

3.7.5 ONIX Evaluation

To illustrate the potential of ONIX to be used in production environments, we conducted a series of functional tests to assess the overall scalability and performance of the platform. In particular, in addition to a set of operational tests with a couple of srsRAN UEs and one eNB, we have studied the end-to-end latency to provide RNIS data to MEC applications for an increasing number of subscribers.

[16]Kubernetes is a flexible and an open-source container orchestration system, compatible with Docker containers and able to manage their deployment, scaling and whole lifecycle.

Given the difficulty in relying on actual hardware for testing the presence of more than few tens of UEs, we created an additional module called *Trace Player*, which can read real LTE traces and emulate the presence of hundreds of UEs. UEs are emulated in terms of the RNI data they would generate in a real scenario, in which up to hundreds of devices are connected to an eNB, and RNI data for each of them is gathered through the Mp2 interface. Therefore, even if no real LTE network is involved, ONIX is fed with RNI data as if several UEs were connected to an eNB, in turn connected to ONIX through the Mp2 interface.

We selected a set of traces collected by the University of Cork containing radio-level measurements from an operational cellular network [164] to perform the evaluation in the most realistic situation possible. Note that from the standpoint of the MEC platform there is no difference between real and emulated UEs since the RNIS data is generated and fed to ONIX in the same way it would be fed in a real network.

We also developed a *Consumer* module in Python 3, acting as a sample MEC application reading the RNIS data from the AMQP message broker and computing the latency using the timestamps inserted by the *Publisher* as metadata. In our measurement campaign, we studied the performance of the entire end-to-end chain from the *Publisher* module, which receives the RNIS data from ONIX, to the MEC application (where the data is consumed) including all the intermediate MEC Platform components involved in this pipeline. Since MEC applications are normally co-located within the MEC platform, we considered only local measurements, where the application (i.e., the *Consumer*) is deployed in the MEC platform. Thus, there is no need to account for any additional communication delay from the producer to the consumer of the RNIS data.

The number of UEs per process has been varied during the tests to analyze the best trade-off between low resource usage and low communication latency. In general, the obtained results depend on the hardware (and its resources, such as RAM and CPU) where the MEC Platform and the *Publisher* are run. In this evaluation, the testbed comprises mid-performance off-the-shelf laptops, with Intel Core i7 quad-core/eight-thread CPUs (2.80 GHz, turbo: 3.80 GHz) and 16 GB of RAM, running Ubuntu 20.04 LTS. It is expected that, when deploying our platform on a dedicated server, the number of manageable UEs should noticeably increase.

To ensure that the tests run without incurring any performance issue, the analysis reached a total of 250 UEs when using only 1 UE per process due to the limitations imposed by the hardware running the *Publisher*, and 500 UEs in all the other cases. All the tests consider 300 RNIS data chunks for each UE along each run, having on average one new data item available every second. This is compliant with 3GPP specifications, which state that the reporting period of UE measurements can vary from 120 ms up to 60 minutes [165]. Each test has been repeated 15 times.

Finally, the AMQP broker has been run on a customizable Kubernetes container

and deployed on a Kubernetes-managed cluster together with LightEdge. This ActiveMQ container has been made available on the Dockerhub container library and community[17].

The first part of the evaluation focuses on measuring the communication latency from the moment new data is available to the *Publisher* (i.e., the AMQP producer) to the time it is processed by a consumer (i.e., a MEC application).

We computed, for each test, the average latency involving the 300 values obtained from each piece of data coming from the traces. Then, the final measurement is the average of these values over all repetitions, which is reported, along with the 95% confidence intervals in Figure 3.35.

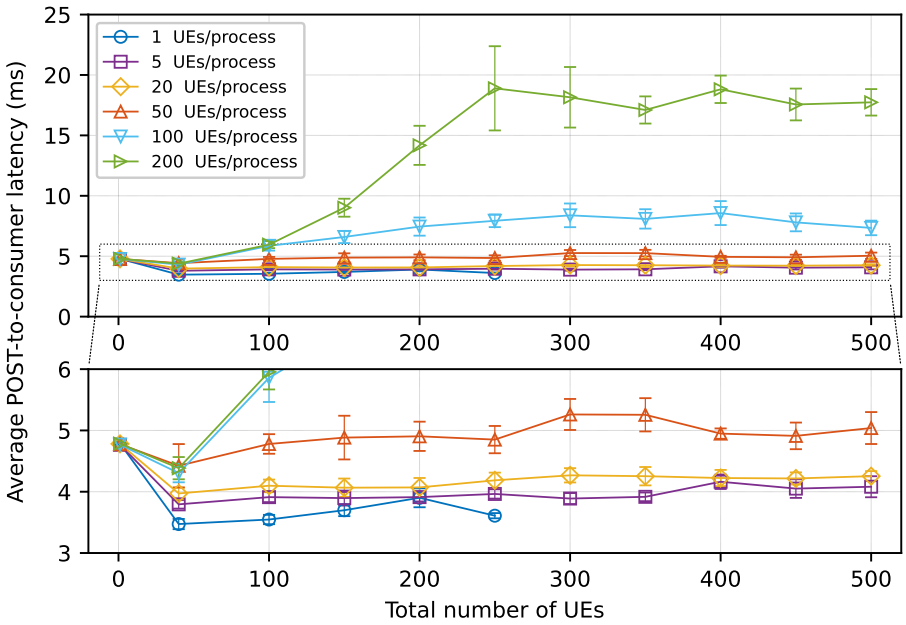


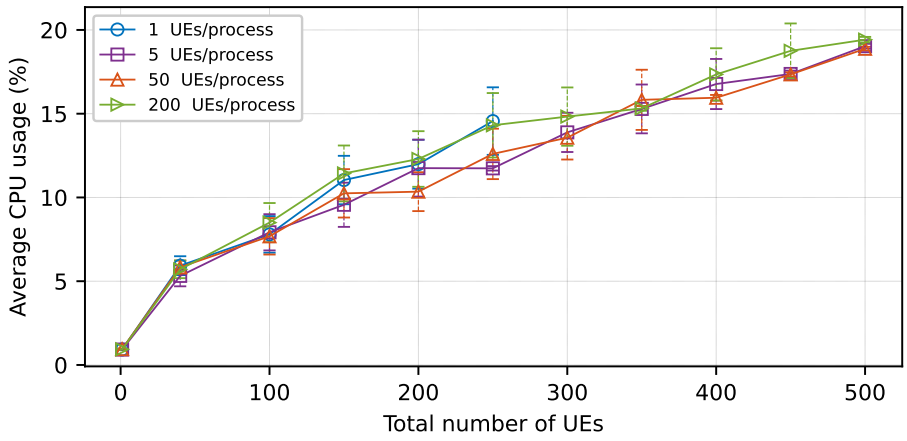
Figure 3.35: Average latency as a function of the total number of subscribed UEs.

As can be seen, increasing the number of "UEs per process" causes a rise in the average latency and in the confidence intervals (especially when assigning more than 100 UEs to each process). This is due to how the AMQP library (i.e., Apache Qpid Proton), used to develop the *Publisher*, interacts with ONIX and to the accumulation of events inside each process when the number of UEs increases. However, when the "UEs per process" is appropriately tuned, our prototype maintains a low latency (less than 10 ms) even when a high number of subscribers are involved. These results also show that the fine-tuning of the "UEs per process" parameter and the scalability features of our platform can help handle a high number of UEs, keeping a low overall latency. The zoomed-in portion at the bottom of the plot

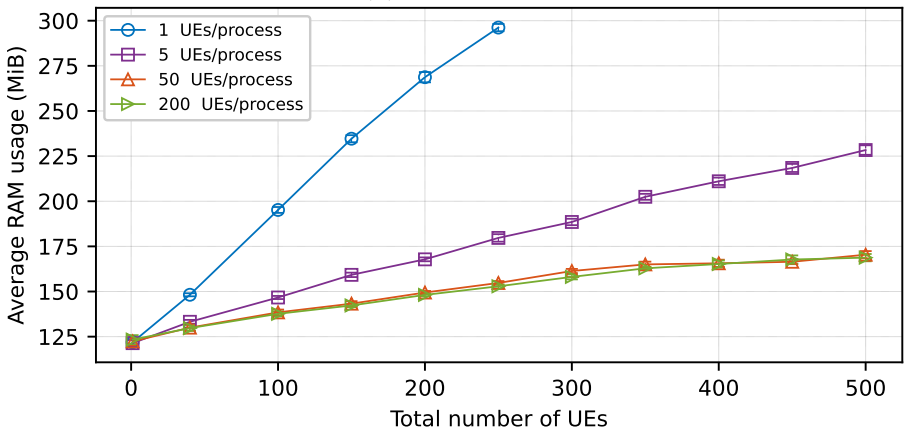
[17] https://hub.docker.com/r/francescoraves483/activemq_5-15-11_alpine

confirms a minimal difference in latency when handling less than 50 "UEs per process". Interestingly, it is also worth noticing that a *plateau* effect can be found in the end-to-end latency starting from approximately 250/300 concurrent UEs. From a system scalability standpoint this is a very important result, as the overall latency is not expected to increase beyond a certain value.

The second part of the evaluation is instead related to the CPU and RAM usage of the AMQP message broker container, obtained via the `docker stats` command, after retrieving the container ID from Kubernetes. The results are shown in Figure 3.36.



(a) CPU usage



(b) RAM usage

Figure 3.36: CPU and RAM usage of the message broker container as a function of the number of UEs.

As can be seen in Figure 3.36a, the CPU usage shows its independence from the "UEs per process" parameter, which indicates how the broker-side CPU usage varies almost solely with the total number of UEs, with an increase up to 20% in the 500 UEs case. It is important to mention that, with the aim of improving the

plot readability, only the most significant lines have been depicted. The observed behavior was the same for all the other values of UEs per process, with the confidence intervals being superimposed for each number of total UEs. Although the obtained values are slightly oscillating and depend on the underlying hardware, the outcome is very good as it is proved that only a fraction of the CPU is used, even when a high number of subscribers are involved.

Concerning instead RAM usage, it can be seen that the container memory requirements increase as the number of served UEs grows. Conversely, assigning less UEs to each process in the *Publisher* causes greater RAM consumption due to a higher number of AMQP v1.0 connections, as depicted in Figure 3.36b.

To conclude the evaluation, we draw some conclusions on the "UEs per process" parameter, which is crucial to improve the performance of the publisher module and of the whole ONIX platform. A tradeoff between the RAM usage, the number of UEs which can be handled and the average measured delay (while the CPU usage is independent from this parameter, as mentioned earlier) is found between 20 and 50 UEs per process. It is important to consider, however, that these values may vary depending on the hardware capabilities, and thus some fine tuning around these values may be needed after the deployment of ONIX in production automotive environments.

Furthermore, we were able to prove how ONIX can effectively scale with the number of UEs, and how it is well suited to serve the needs of the connected vehicles market.

3.7.6 Considerations on future directions

Several research challenges still remain open, and are going to be tackled in the future versions of ONIX. As an example, although ONIX can provide MEC application with real-time RAN information, it contains low-level network parameters such as RSRP and RSRQ. In this domain, AI techniques could be used to generate actionable metrics, e.g., expected bitrate or latency, starting from such low-level measurements, paving the way to a new generation of automatic resource scaling and management operations.

3.8 Publications

This Section reports our publications, related to the topics presented in this Chapter.

3.8.1 Conferences

Open DSRC platform

- F. Raviglione, M. Malinverno and C. Casetti, "Demo: Open Source Platform for IEEE 802.11p NICs Evaluation", IEEE WoWMoM 2019, Washington DC, USA, June 2019, pp. 1-3 [28]
- F. Raviglione, M. Malinverno and C. Casetti, "Demo: Open source testbed for vehicular communication", ACM MobiHoc 2019, Catania, Italy, July 2019, pp. 405-406 [29]
- F. Raviglione, M. Malinverno and C. Casetti, "Characterization and Performance Evaluation of IEEE 802.11p NICs", 1st ACM Workshop on Technologies, mOdels, and Protocols for Cooperative Connected Cars (TOP-Cars), Catania, Italy, July 2019, pp.13-18 [33]

LaMP and LaTe

- F. Raviglione, M. Malinverno and C. Casetti, "A Flexible, Protocol-Agnostic Latency Measurement Platform", IEEE VTC2019-Fall, Honolulu, Hawaii, USA, September 2019, pp. 1-5 [30]

3.8.2 Journals

ONIX

- E. Coronado, F. Raviglione, M. Malinverno, C. Casetti, A. Cantarero, G. Cebrián-Márquez and R. Riggio, "ONIX: Open Radio Network Information eXchange", IEEE Communications Magazine, vol. 59, no. 10, November 2021, pp. 14-20 [31]

Chapter 4

Performance assessment and comparison of V2X technologies and applications

The development and deployment of an open platform for connected vehicles requires, as a critical step, the evaluation of different technologies for V2X, as they represent fundamental enablers for the overlying applications and use cases. Different technologies can indeed provide advantages and disadvantages, depending on the connected vehicle density and on the target application, and their performance assessment can provide very useful insight on which technology (or combination of them) could be best suited for a given use case.

The performance assessment of a V2X technology (e.g., IEEE 802.11p or LTE-V2X Mode 4) and/or application can happen in two main ways: the first is relying on simulation and emulation, in which real vehicles are not involved and a realistic, but simulated, scenario is modelled. The second relies instead on real vehicles and hardware for the execution of field tests, in different LOS and NLOS, urban and rural scenarios. Both ways are fundamental to fully evaluate an access technology, and they require respectively *(i)* the availability of simulation frameworks targeted at evaluating V2X applications, and *(ii)* the availability of dedicated hardware boards, running proper measurement tools and platforms.

Simulations can provide significant advantages when evaluating for the first time a technology in a given scenario, or a novel V2X application. They also prove to be very convenient and effective for large scale scenarios, for which it would be economically unfeasible to equip a large amount of test vehicles. Indeed, because of the economic, logistical, and safety-related limitations which occur whenever

working with real vehicles, the performance evaluation and assessment of V2X technologies is frequently carried out in a simulation environment. This has led several works to focus on the first approach for performance evaluation, with the development of a number of V2X simulation frameworks [166]–[168].

However, the closer the simulated phenomenon gets to the physical medium, the more difficult it is for the model to accurately reflect reality, due to hard-to-predict effects such as Doppler shift, shadowing and multi-path fading. Furthermore, simulation models take into account only the most important aspects of reality, as it would be impossible to exactly shape all the involved variables. Alongside simulation studies, which are nevertheless very useful for large and pre-deployment scenarios, it is thus critical for the assessment of V2X communications to perform field tests, as they reflect the actual capabilities of the devices under test. Moreover, field tests allow researchers to "capture" all the complex interactions of the tested technology with real-world laboratory, urban and rural scenarios.

Therefore, this thesis aims at presenting both ways of assessing the performance of different technologies for V2X, focusing on open hardware and software. First, it presents the ms-van3t simulation and emulation framework, released with an open source license on GitHub [23]. This framework embeds several state-of-the-art models for IEEE 802.11p, standard LTE and LTE-V2X Mode 4, allowing the user to develop and simulate V2X applications on top of these technologies, and possibly comparing them. It also comes with a simplified installation procedure for any Debian-based Linux distribution, together with features not available in other similar solutions, such as the possibility to use pre-recorded GNSS traces instead of a mobility model. Thanks to ms-van3t, the user can simulate small to large scale scenarios, and gather interesting insights on the behavior of different access technologies, such as the ones presented in Section 4.1.6.

Then, this Chapter presents two field test campaigns, aimed at evaluating different IEEE 802.11-based technologies on the field, to assess their performance in a real-world environment, with open source software and low-cost hardware. These tests also allowed us to further validate the DSRC platform presented in Section 3.2, and, at the same time, to employ it to test IEEE 802.11p in the field, in both LOS and NLOS conditions.

4.1 Simulation and emulation: the ms-van3t framework

Because of the complexity and the high deployment costs of vehicular applications, it is often convenient to extensively test them by simulation, before moving to the deployment and field testing phase.

Extensive simulation studies require dedicated tools and framework for the simulation (and, possibly, emulation) of V2X applications. These platforms should

include models for different access technologies, the possibility of easily implementing user-defined V2X service and gathering the metrics of interest. The latter can be network-related, such as latency and throughput, or application-related, such as the number of accidents with and without a collision avoidance system. Furthermore, a fundamental requirement for these frameworks is the ability to simulate realistic mobility patterns of vehicles, starting from either pre-recorded GNSS traces or from accurate mobility models. As a matter of fact, most of the currently available V2X frameworks leverage Simulation of Urban MObility (SUMO)[1], which is interfaced with a discrete-event network simulator. SUMO is a microscopic, open source traffic simulator, supporting large networks and different types of road users (vehicles, trucks, cyclists, pedestrians and so on). SUMO provides a set of tools for the creation of realistic scenarios, and can emulate an arbitrary number of vehicles in urban, sub-urban and highway road networks.

Simulation, at least for large scale analysis and pre-deployment studies, provides several advantages. Indeed, testing an application in its early stage by deploying it to a large number of vehicles may imply non-negligible costs. Furthermore, testing road safety applications with actual vehicles during their early development may pose safety risks for the driver and for other road users, in case the application fails to activate or provides the wrong inputs to the vehicle internal logic.

Simulation and emulation tools thus play a crucial role in this field, as they support the development of applications and the deployment of the underlying physical and access layer technologies.

Therefore, the main aim of this Section is to present a novel open source V2X simulation and emulation framework, with several innovative features which are not available in similar solutions in literature. Our framework, named ms-van3t (Multi-Stack VANET framework for ns-3), is designed to run on Linux with a very simple installation procedure, as opposed to tools like OMNeT++, and can manage both vehicle mobility and connectivity. This is achieved thanks to the combination of SUMO and the ns-3 network simulation framework[2], together with several novel modules and components targeted at V2X applications and advanced mobility simulation. Our framework thus provides a modular and integrated Vehicle-to-Everything (V2X) simulation and emulation tool tailored to evaluate the impact of new communication protocols on ITS applications.

One of the main features of ms-van3t is the inclusion of several state-of-the-art open-source models for V2X communications, enabling the simulated connected vehicles to flexibly exploit them and to easily switch from one to the other depending on the simulated scenario. To the best of our knowledge, no other framework

[1] https://sumo.dlr.de

[2] https://www.nsnam.org/

includes the same wide support for different technologies, all within the same package.

These state-of-the-art models currently include IEEE 802.11p, Release 14 LTE-V2X Mode 4 [169] and standard LTE. NR-V2X Mode 2 [170] is instead in the process of being integrated and tested.

ms-van3t also includes a full ETSI ITS-G5 stack, developed from scratch to best take advantage of the ns-3 capabilities. Our stack currently supports CAMs, DENMs and IVIMs for road signage information. IVIMs are rarely supported by existing simulation frameworks, and they thus represent a distinctive feature of ms-van3t.

ms-van3t does not only provide a simulation framework, but it is also capable of emulating vehicles. This feature allows the user to set up a mobility scenario on SUMO, and then let the simulated vehicles communicate with real entities, thanks to the reception and transmission of CAMs and DENMs from/to the external world through a physical interface of the device on which ms-van3t is run. This effectively enables Hardware-in-the-Loop (HIL) testing.

Besides emulation mode, the main features of our framework can be summarized as follows:

1. it enables the comparison of different technologies (e.g., to study the advantages and disadvantages of IEEE 802.11p versus C-V2X under different scenarios, as presented in Section 4.1.6), involving up to thousands of nodes;

2. it supports very large scale simulations, up to thousands of vehicles in simulation mode, and up to tens of vehicles in real-time emulation mode;

3. it is available on a fully open-source and up-to-date repository, available at the public GitHub repository in [23];

4. it supports the great majority of Debian-based Linux distributions and can be easily installed through an automated `bash` script;

5. it includes an easy-to-use script to switch between different ETSI ITS-G5 message versions (which is not available, to the best of our knowledge, in any other open source simulation for V2X);

6. it allows the user to use real GNSS traces for the mobile nodes, with real GNSS errors, instead of relying solely on synthetic scenarios generated by means of SUMO;

7. it embeds a web-based vehicle visualized to easily show the mobile nodes on real maps, besides the SUMO GUI, which also supports the visualization of real vehicles in emulation mode, thanks to CAMs received by ms-van3t.

4.1.1 Comparison with existing V2X frameworks

Before delving into the main features and architecture of our open source framework, this Section highlights the main advantages of our solution, and compares it with other existing V2X framework, both commercial and open source.

Table 4.1 compares ms-van3t with the most established V2X frameworks available in the literature. Among them, the most well-know is Veins [166]. This open source framework enables the simulation of large scale V2X scenarios and applications thanks to the integration of OMNeT++ with SUMO. OMNeT++ is a discrete-event network simulator, which can be interfaced with SUMO thanks to the so-called TraCI (Traffic Control Interface) interface [171]. TraCI enables a bidirectional communication between SUMO and OMNeT++, letting the latter control the SUMO simulation (e.g., by making a vehicle brake or accelerate) and gather information from the simulated vehicles (such as their position, speed and heading).

Veins provides a native IEEE 802.11p implementation, but its capabilities have been extended thanks to the integration with projects like SimuLTE [167], for the simulation of LTE networks. Moreover, the Artery [168] and Vanetza [172] projects further extended Veins with a flexible platform for V2X applications prototyping and an ETSI ITS-G5 implementation supporting different message types, as reported in Table 4.1.

Name	Multi-stack	Large scale simulations	Native integration with SUMO	Pre-recorded GNSS traces	Emulation mode	Supported ETSI messages	Easy switch between ETSI versions	Web-based visualizer	Open source
Veins	*X* 802.11p	✓	✓	*X*	*X*	No ETSI stack	n.d.	*X*	✓
Artery	*X* 802.11p	✓	✓	*X*	*X*	CAM, DENM, MAPEM, SPATEM, CPM	*X* (version 2)	*X*	✓
SimuLTE	*X* LTE	✓	*X*	*X*	*X*	No ETSI stack	n.d.	*X*	✓
iTETRIS	✓ 802.11p, WiMAX, UMTS, DVB-H	✓	✓	*X*	*X*	CAM, DENM, LDM	n.d.	*X*	✓
VENTOS	*X* 802.11p	✓	✓	*X*	✓	No ETSI stack	n.d.	*X*	✓
RVE	*X* 802.11p	✓	✓	*X*	✓	No ETSI stack	n.d	*X*	*X*
ms-van3t	✓ **802.11p, LTE, Release 14 C-V2X Mode 4**	✓	✓	✓	✓	**CAM, DENM, IVIM**	✓ **(version 1/2)**	✓	✓

Table 4.1: Comparison of ms-van3t features with available vehicular networks simulation and emulation solutions

As opposed to Veins, which leverages OMNeT++, iTetris [173] relies on ns-3 and SUMO, like ms-van3t, providing a full ETSI ITS-G5 stack and models for different access technologies such as IEEE 802.11p, WiMAX, UMTS and DVB-T. However, as opposed to our solution, it does not support emulation mode or the usage of pre-recorded traces instead of SUMO.

In contrast to the aforementioned frameworks, ms-van3t focuses on easily enabling simulation over different access technologies, all included within the same package (without the need of integrating additional packages or platforms), and embeds a full-fledged C-V2X Mode 4 model. Indeed, it comes as a self-contained tool including everything needed to quickly develop and test V2X applications.

Furthermore, as mentioned earlier, ms-van3t provides an emulation mode for V2X applications HIL testing. A number of solutions for vehicular networking emulation can be found in the literature, even though they are often devoted exclusively to HIL emulation. These solutions include RVE [174] and the work by Obermaier *et al.* [175]. Both these works make use of custom components instead of relying on network simulators.

Among the frameworks supporting both large scale simulations and HIL emulation, it is worth mentioning VENTOS [176]. This framework couples OMNeT++ and SUMO with an implementation of the IEEE WAVE protocol stack (as opposed to Vanetza and ms-van3t, which focus on ETSI ITS-G5), providing emulation capabilities. Moreover, its simulation features focus on IEEE 802.11p, and thus do not include several models as in ms-van3t.

In order to support HIL testing, the emulated data must be generated in real-time, which becomes a challenge when considering a large number of vehicles. The authors of [177] propose an extension to Artery to monitor a Device Under Test (DUT), capable of sending and receiving messages from/to vehicles in a simulated scenario. Furthermore, the authors analyze the limitations posed by OMNeT++ when producing real-time traffic. None of these OMNeT++-based solutions, however, analyze and specify the limits in the number of vehicles supported in real-time when emulation mode is enabled. Conversely, we analyze the limits with different CPU models in Section 4.1.5, and provide this information to the ms-van3t users [23].

Finally, as opposed to other open-source solutions, ms-van3t supports the usage of pre-recorded GNSS traces, which can be leveraged as an alternative to SUMO, thus providing paths based on real GNSS data to the simulated vehicles running the applications under test. Furthermore, it is the only framework providing an easy and practical way of testing V2X applications leveraging different versions of the ETSI standards.

4.1.2 Features and architecture

ms-van3t is composed of different software modules, interacting among themselves through dedicated interfaces, providing a powerful platform for V2X applications testing. As mentioned earlier, ms-van3t hinges upon the combination of SUMO with ns-3 (ns-3.33, at the time of writing), a modular discrete-time network simulator, thanks to an ns-3-compatible implementation of the TraCI interface. We implemented an ETSI ITS-G5 stack compatible with ns-3, as well as a set of additional features, and enhanced ns-3 with such novel modules, which can be integrated as libraries thanks to an easy-to-use installation script. The architecture of our framework is schematized in Figure 4.1.

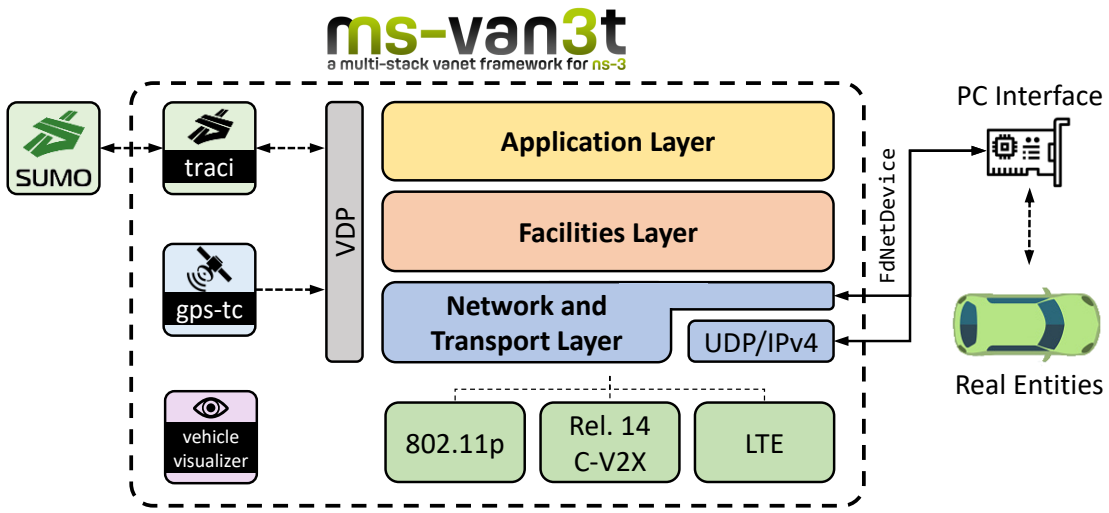


Figure 4.1: High-level scheme of the ms-van3t architecture, including three different state-of-the-art access layer models. and comprising simulation and emulation capabilities.

As can be seen, ms-van3t provides a full vehicular stack, with which simulated vehicles can be easily equipped to perform large scale simulations with different access technologies (highlighted as green boxes). Then, as depicted on the left, the vehicle mobility can be managed either through SUMO, thanks to the TraCI interface [171], or by means of pre-recorded GNSS traces, thanks to the `gps-tc` novel module. Finally, the application layer (depicted as a yellow box) can be fully defined by the user, in order to test many different V2X applications. The interface to the Facilities layer happens through dedicated message transmission and configuration functions and message reception callbacks.

The main features of our framework are described below, with reference to Table 4.1.

Multi-stack

As mentioned, ms-van3t provides three state-of-the-art models, which have been fully integrated into the ns-3 ecosystem, letting the user easily choose one or the other. Additionally, the ms-van3t simulation environment is built to offer a high-level abstraction layer so that vehicular applications can be developed and tested without the need of configuring much about the lower layers. In this way, the underlying communication stack can be used as a sort of black-box, so that the users willing to focus only on application-related aspects can be offloaded from the burden of writing low-level code.

The focus of ms-van3t, as its name suggests, is on supporting multiple technologies, which include:

- **IEEE 802.11p:** this is the ns-3 native DSRC model, integrated with the ETSI ITS-G5 stack. When this access technology is selected, vehicles are equipped with OBUs, enabling both V2V and V2I communications. In the first case, messages are directly exchanged by simulated vehicles, while, in the second case, one or more RSUs are equipped with IEEE 802.11p to receive messages from vehicles and run centralized MEC applications. As required by the standard, messages are broadcasted after being encapsulated inside ETSI BTP and GeoNetworking.

- **Release 14 LTE-V2X Mode 4:** in this model, proposed by Eckermann *et al.* [169], vehicles (and, if required, RSUs for V2I communication) are equipped with C-V2X-compliant devices to directly communicate over the *PC5* interface. Thanks to Mode 4, vehicles can broadcast messages directly with their peers without relying on the eNB arbitration, as it would be required by Mode 3 and standard cellular communication. The exchange of messages happens as in the previous case, with ETSI ITS-G5 messages being broadcasted after the encapsulation into BTP and GeoNetworking.

- **LTE**: thanks to this model, vehicles can be equipped with UEs, connected to a 4G eNB thanks to the *Uu* interface. This access technology is used to model V2N scenarios, since the centralized application is placed in servers that are connected to the EPC, acting as remote hosts in the LTE network. In order to conform with the IP-based addressing scheme of LTE networks, the network entities further encapsulate their messages into UDP/IP headers and send them as unicast packets. It should be mentioned that LTE Multimedia Broadcast Multicast Service (MBMS) is currently not supported by this model, but we are nevertheless investigating its integration to enable broadcast transmissions in LTE V2X scenarios.

Finally, it should be mentioned that the choice of the right technology is up to the user, depending on the application needs and on which access technologies should

be compared for its evaluation. We are currently planning to combine together multiple models, to provide an additional tool to evaluate, for instance, the effects of having IEEE 802.11p and C-V2X interfering with each other.

Large scale simulations

The main ms-van3t feature is represented by its simulation environment, supporting large scale simulation of different scenarios. Indeed, ms-van3t supports the simulation of an arbitrary amount of vehicles, limited only by the hardware capabilities of the device on which our framework is run.

Data collection is one of the critical challenges when simulating a large number of vehicles. Indeed, it may not be trivial to gather the desired metrics for a significant amount of nodes, to be later post-processed (e.g., to collect average values). To this aim, ms-van3t integrates a special measurement module, called `PRRsupervisor`, which works transparently with respect to the underlying access technology. This facilitates the comparison between different protocols and simplifies the collection of metrics for the user.

The `PRRsupervisor` enables the automatic collection of two main metrics: *(i)* the one-way latency between the transmission of a packet and its reception by all nearby vehicles, and *(ii)* the Packet Reception Ratio (PRR). Both metrics are reported as average values at the end of each simulation, and can be saved to CSV files for further processing.

The PRR provides an indication on how many packets are lost due to collisions and harsh propagation conditions, and it is computed according to the 3GPP TR 36.885 technical report [178]. Specifically, the PRR for each message is calculated as the ratio between the number of vehicles, X, that within a baseline distance (e.g., 100 meters) successfully receive the message, and the total number of nearby vehicles, Y, within the same distance. The `PRRsupervisor` then averages the per-message PRR over all the message transmissions (n_{tx}), to compute the average PRR, given a certain value of baseline distance; i.e.,

$$PRR_i = \frac{X_i}{Y_i},\ i = 1, ..., n_{tx} \tag{4.1}$$

As a consequence, a PRR of 1 means that no packets are lost, and represents an ideal condition, while a theoretical PRR of 0 would mean that no communication is possible.

With the aim of evaluating the jitter of periodic messages, such as CAMs, it is also planned to upgrade this module to support the collection of the Packet Inter-Reception (PIR), defined by 3GPP as the time elapsed between two consecutive receptions of two different packets between the same couple of nodes [178].

Native integration with SUMO

ms-van3t is natively integrated with SUMO, thanks to a dedicated implementation of the TraCI interface, adapted from [179]. In ms-van3t, each vehicle travelling in a simulated SUMO scenario is equipped with a full ETSI ITS-G5 stack, thanks to the integration with ns-3, which provides each node with network connectivity, and on which V2X applications can be developed.

TraCI offers a bidirectional coupling between ns-3 and SUMO, letting the network stack retrieve information from SUMO, such as vehicle position, speed, acceleration and heading, and the V2X application control the vehicle dynamics, for instance to make a vehicle brake in case of a detected collision hazard. Complex scenarios can be thus realized, in which connected vehicles can be actively controlled to simulate the presence of partially or fully autonomous vehicles.

Pre-recorded GNSS traces

This is a novel feature, which, to the best of our knowledge, is not found in other similar simulations. Indeed, ms-van3t supports the usage of GNSS traces instead of SUMO, to simulate vehicular mobility in presence of real GNSS errors, without relying on any mathematical traffic model.

This feature is enabled by the `gps-tc` module, which provides functions and classes to simulate mobility based on traces recorded offline and reproduced within ms-van3t. Testing with GNSS errors and real traces can be very useful to evaluate the behaviour of applications in harsh conditions, such as in urban canyons, in which the positioning accuracy is expected to noticeably decrease [180]. It should be mentioned that these errors are often neglected in works leveraging frameworks based on SUMO only.

The user can provide as input any CSV GNSS trace containing at least the following information, for each time instant and for each vehicle:

- Vehicle ID
- Timestamp since any moment in time taken as reference, in seconds (with decimal digits to represent milliseconds)
- Position (latitude and longitude)
- Speed
- Heading with respect to the North
- Acceleration (optional, if not specified CAM messages will contain an “Unavailable” value)

The `gps-tc` module can be configured to support different CSV file formats, recorded by different GNSS devices and with different field names.

It is worth highlighting how both `gps-tc` and TraCI act as Vehicle Data Providers (VDP) for the network stack layers implemented in ms-van3t. According to the ETSI standards, the VDP is the entity providing the Facilities layer with the vehicle dynamic data and status information needed to encode and transmit standard-compliant messages [63].

Emulation mode

As mentioned earlier, ms-van3t provides a full-fledged emulation mode, enabling communication between the framework and external entities, such as real vehicles. This enables HIL testing of V2X applications, together with the creation of hybrid scenarios in which emulated vehicles (through SUMO or through pre-recorded GNSS traces, thanks to the `gps-tc` module) can interact with real vehicles thanks to standard-compliant messages, and vice versa. As an example, it is possible to create a scenario in which a real vehicle communicates with several emulated vehicles in its vicinity.

The emulation mode is based on the so-called *FdNetDevice*, a type of network device available in ns-3, which can be used for the transmission and reception of traffic to/from the external world. In short, this kind of device routes the packets to a physical interface instead of using a simulated access layer model. Thanks to it and to the ETSI ITS-G5 stack, ms-van3t enables the transmission and reception of CAMs, DENMs and IVIMs from real vehicles into the simulated scenario and vice versa.

Two main modes of operation are supported by the emulation mode:

- Standard mode (V2V-like), in which packets generated by the ETSI Application, Facilities and Network and Transport Layers are directly broadcasted through the physical interface;
- UDP mode (V2I-like), in which packets composed by the aforementioned layers are further encapsulated inside UDP and IP, for the transmission to a server on the Internet or residing within an RSU; it should be noted that this is one of the communication profiles foreseen for infrastructure services by the ETSI standards, under the name of Communication Parameter Setting 003 [181].

ms-van3t is able to receive standard-compliant messages in both cases, from a given physical interface. We also developed a dedicated sample application [23] to showcase the emulation mode and provide a baseline to users developing their own emulation scenarios. This application foresees several vehicles travelling on a realistic road layout, and broadcasting CAMs and DENMs on a given interface, or targeting a remote server.

ETSI ITS-G5 stack

ms-van3t integrates an efficient ETSI ITS-G5 stack, implementing, at the time of writing, both the CA Basic Service and the DEN Basic Service. It also includes an initial implementation of the Infrastructure to Vehicle Information (IVI) service for the transmission and reception of IVIMs.

Our stack includes both BTP and GeoNetworking and enables standard-compliant simulations, with vehicles exchanging messages such as CAMs and DENMs. To the best of our knowledge, ms-van3t is the first framework integrating ETSI ITS-G5 into ns-3, thanks to a novel `automotive`[3] module. This component includes the whole ETSI ITS-G5 stack, with the encoding and decoding rules for vehicular messages, by means of ASN.1 UPER, and the networking logic foreseen by ETSI for vehicular scenarios. It also includes the logic of the V2X applications developed on top of ETSI ITS-G5.

The transmission of DENMs can be triggered thanks to dedicated, standard-compliant, functions, while the reception of both CAMs and DENMs is managed thanks to callback functions.

Easy switch between ETSI versions

The standardization activities carried on by ETSI over the years led to the definition of multiple versions of their standards, corresponding to slightly different versions of CAM, DENMs and other kinds of vehicular messages.

A new standard version is normally designed to supersede the older ones, with small updates to some of the fields included in the respective messages. Therefore, it is always recommended to rely on the latest version when developing and testing a new V2X application. However, this is not always true in the automotive field, where the adoption of a new standard may take a significant amount of time, due to possible interoperability issues which may arise and additional testing efforts which are required after each update.

ms-van3t thus provides a way to simulate multiple versions of the ETSI ITS-G5 standards, depending on the scenario and possibly enabling a comparison between different versions.

In particular, concerning CAMs and DENMs, two message versions are currently defined: version 1 and version 2. The latter corresponds to the latest version of the CAM [63] and DENM [64]. These versions can be easily switched thanks to a one-step script included in each ms-van3t installation.

To the best of our knowledge, ms-van3t is the first framework with such a feature, as opposed, for instance, to Artery [168], which supports version 2 messages only.

[3] https://github.com/ms-van3t-devs/ms-van3t/tree/master/src/automotive

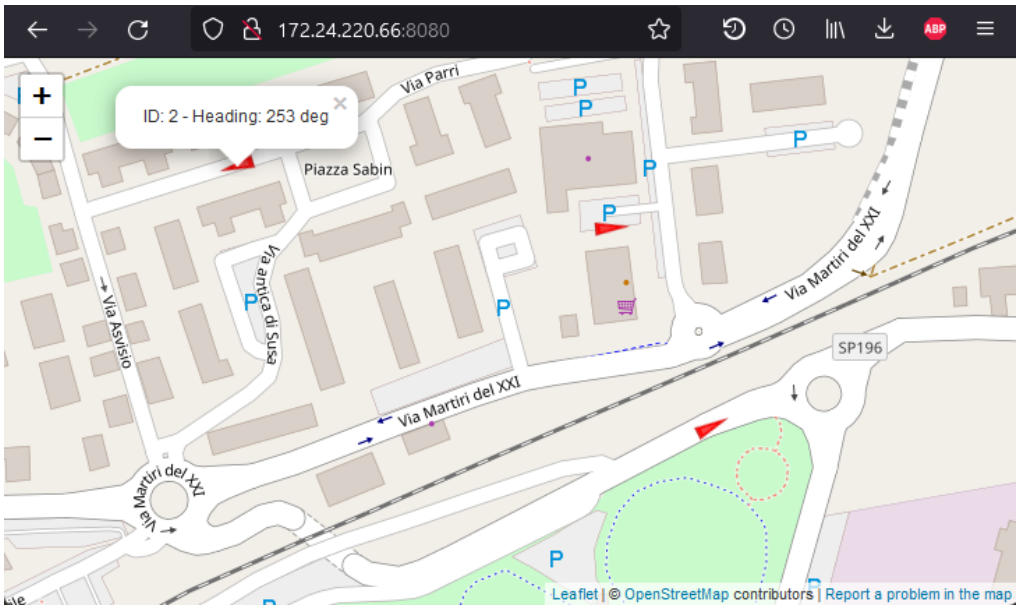


Figure 4.2: The ms-van3t web-based vehicle visualizer, with OpenStreetMap tiles, as rendered by the Firefox browser during a simulation with pre-recorded GNSS traces.

Web-based visualizer

Among its novel features, ms-van3t provides a web-based vehicle visualizer, which is able to display the simulated (and emulated) vehicles on a map, either based on layers by *OpenStreetMap*[4] or by *Mapbox*[5], if a proper token is provided. The map is rendered directly inside the browser and shows in the vehicles' position and heading during the whole simulation, besides the already available SUMO GUI.

This feature is particularly useful for demonstration purposes and when using `gps-tc` to model the vehicle mobility. Indeed, when pre-recorded GNSS traces are used, no SUMO GUI is involved, and the real-time movement of simulated vehicles can be displayed only thanks to the web-based visualizer.

This feature is implemented by the `vehicle-visualizer` module, which launches the visualizer as soon as a simulation is started, making it available on the localhost IP, on a given configurable port.

Furthermore, in emulation mode real vehicles can be easily inserted inside the web-based visualizer, alongside simulated entities, thanks to the received CAMs. This can prove to be very useful even when SUMO is used to emulate the position of vehicles within ms-van3t, since its GUI cannot normally display external entities

[4] https://www.openstreetmap.org

[5] https://www.mapbox.com

alongside the simulated ones.

Figure 4.2 shows a screenshot of the vehicle visualizer during a simulation with pre-recorded GNSS traces.

Open source and Linux support

Finally, ms-van3t is released under a fully open source license. Specifically, it is available on GitHub under the GPLv2 license [23], together with an extensive *README* file. Its openness enables researchers to easily adapt, modify and redistribute its code, fostering research in the V2X field.

ms-van3t has been tested both on Ubuntu 18 LTS and Ubuntu 20 LTS. Furthermore, it can also be executed on Microsoft Windows 11 thanks to the WSL2 Windows Subsystem for Linux.

4.1.3 Developing V2X applications with ms-van3t

ms-van3t has been designed to facilitate the development and testing of vehicular application in a simulated environment. Indeed, arbitrarily complex scenarios can be generated with few steps, thanks to its straightforward implementation.

Five main steps are required to develop and simulate a V2X application based on the European standards for V2X:

1. **Definition of the mobility model.** There are currently two possibilities. If SUMO is used, the user should define a SUMO scenario and upload the corresponding XML files in a dedicated ms-van3t folder. If, instead, pre-recorded GNSS traces are used, the user should provide one or more CSV files and copy them into a dedicated folder inside the `gps-tc` module directory. If needed the user may need to format the CSV file as required by the framework, even though `gps-tc` can be easily configure to support the output format of several GNSS receivers.

2. **Definition of the application logic.** The user should define the application logic and the behaviour of both vehicles and infrastructure nodes when receiving V2X messages. The ms-van3t Facilities layer provides the user, developing the application, with functions to trigger new DENMs and start and stop the dissemination of CAMs, together with dedicated callbacks which are invoked when receiving CAMs and DENMs (and, possibly, IVIMs). The application is defined in dedicated C++ files within the `automotive` module.

3. **Configuration of an access technology.** The user should choose an access technology and configure the nodes to use it at the PHY and MAC layers. The user can keep the default network parameters or configure them depending on the desired configuration. These parameters include the C-V2X RRI, the IEEE 802.11p channel, the transmission power level and many others. A set

of examples for IEEE 802.11p, LTE and C-V2X are available as part of the `automotive` module.

4. **Configuration of additional utilities and modules.** Optionally, the user can configure additional modules, such as the `PRRsupervisor` to transparently compute average latency and PRR. The configuration of additional modules should be performed in the same C++ source file which is also used to set up the access technology parameters.

5. **Execution of one or more simulation campaigns.** After a simulation scenario has been created, it can be easily executed thanks to the ns-3 "`./waf run`" command.

Even though they are fundamental, the aforementioned steps require the user to define non-trivial aspects and, most importantly, to work with different modules of the framework. To facilitate the creation and execution of new scenarios, ms-van3t comes with two main sample applications, intended to provide a baseline for the development of more complex use cases.

These applications are, respectively, the *Area Speed Advisor*, focused on a V2I and V2N scenario with either IEEE 802.11p or LTE, and the *Emergency Vehicle Alert*, focused on a V2V scenario with either IEEE 802.11p or LTE-V2X Mode 4. It should be mentioned that these applications has been presented in a previous conference paper [32].

Area Speed Advisor

The Area Speed Advisor (ASA) is a centralized application based on the exchange of CAMs and DENMs, to efficiently enforce speed limits in specific areas. Specifically, a centralized entity supervises the simulated scenario thanks to the reception of CAMs, and generates DENMs, when needed, to inform the vehicles about more restrictive speed limits. This centralized entity can be deployed to an RSU (in the IEEE 802.11p sample application) or in a remote server, reachable through cellular connectivity (in the LTE sample application).

Two versions of the ASA are thus provided. A V2I version employing IEEE 802.11p, and a V2N version using LTE, to showcase the capability of ms-van3t to simulate centralized applications.

In the first case, DENMs are broadcasted every second by a centralized entity, deployed to an IEEE 802.11p RSU. Each DENM has its destination GeoArea (the destination area of DENMs when leveraging GeoBroadcast routing) to match the intended service area, in which the maximum allowed speed of vehicles should be limited. Only the vehicles located inside the GeoArea (i.e., inside the low-speed zone) will process the DENMs and act accordingly, reducing their speed.

In the second case, the centralized application logic runs on a remote server connected to the EPC. In this case, vehicles rely on IP-based routing due to the

addressing scheme of LTE, and send their CAMs as unicast packets to the server. The server processes the received CAMs and generates unicast DENMs to inform vehicles about new limits when they transition between zones with different speed limitations.

This application is schematized in Figure 4.3, together with the SUMO scenario simulating a low-speed area near to a school.

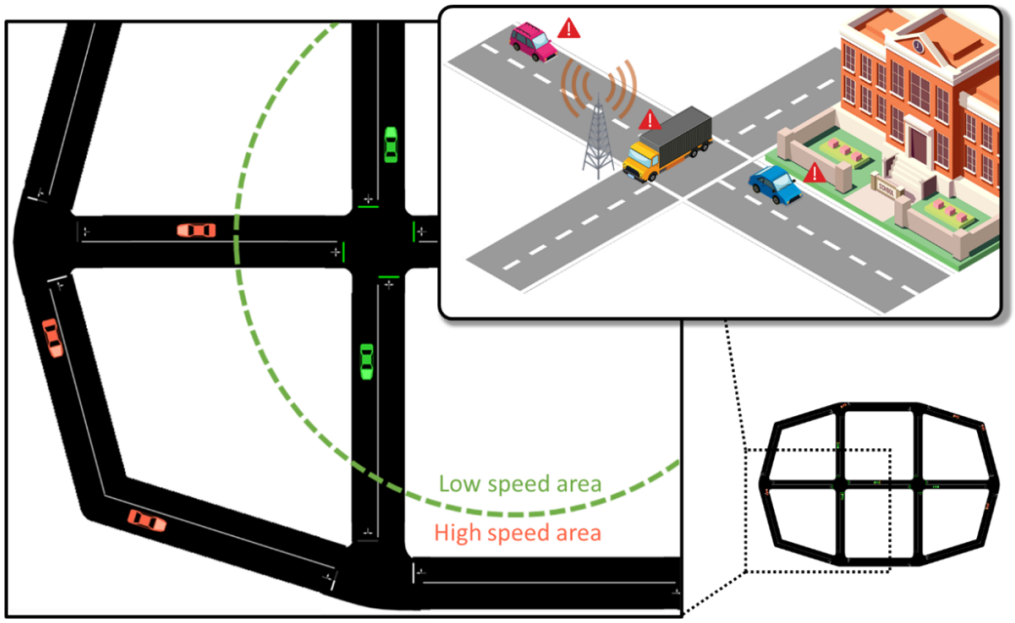


Figure 4.3: Example scenario in which the Area Speed Advisor application is deployed in ms-van3t. The SUMO map is composed of a grid topology with 2 intersections. The area with a low-speed limit is highlighted by the green circle.

Emergency Vehicle Alert

The Emergency Vehicle Alert (EVA) application is targeted at improving the capability of emergency vehicles, such as fire trucks or ambulances, to cross an urban area faster, more efficiently and safely while sharing roads with regular vehicles.

This application models a V2V scenarios, in which both regular vehicles and emergency vehicles exchange CAMs either through IEEE 802.11p or via the *PC5* interface of LTE-V2X Mode 4. Each time a CAM is received by a regular vehicle, it is decoded and the sender type field is checked. If the sender is a nearby emergency vehicle (i.e., its station type, in CAMs, is set to `specialVehicles`, corresponding to `0x0A`), the regular vehicle will perform a set of actions to let the ambulance or fire truck pass without reducing its speed. In particular, if the recipient finds itself to be on the same road segment as the sender (this information acquired from the received CAM messages), it will facilitate the passing maneuver either by reducing

its speed, or, if the regular vehicle is passing other vehicles, by increasing the speed to clear the passing lane as promptly as possible.

This application is schematized in Figure 4.4, and it has been designed to exemplify how to develop a V2V application and configure the underlying access technologies, i.e., IEEE 802.11p or LTE-V2X Mode 4. Furthermore, EVA also represents a relevant example of how distributed strategies aiming at increasing road safety can be efficiently enabled by V2X technologies.

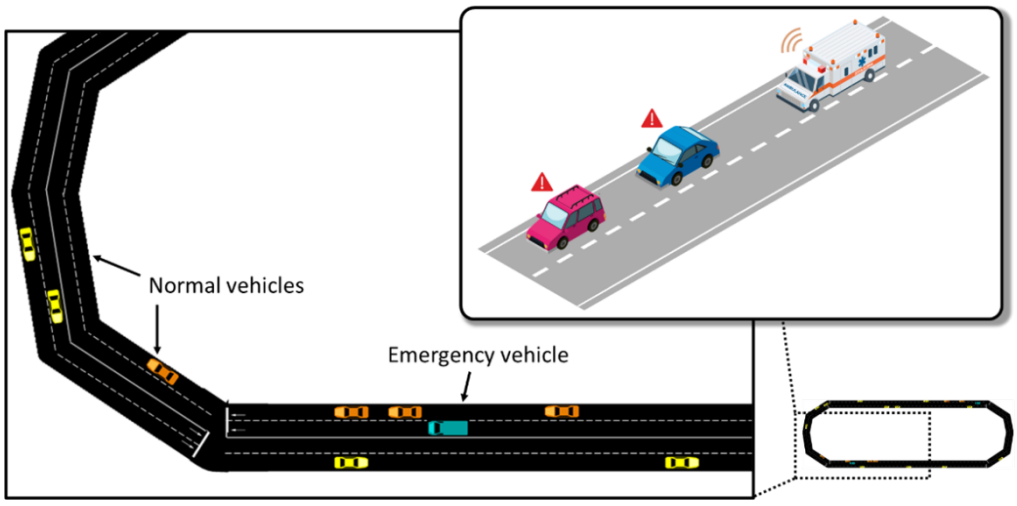


Figure 4.4: Scenario for the sample EVA application in ms-van3t. The SUMO map is composed of a ring topology with two lanes for each direction of travel. The emergency vehicle is the green vehicle, while all others are regular vehicles. A possible implementation of EVA is shown in the overlying image, where it is used by an ambulance to inform other vehicles about its presence.

4.1.4 Evaluation of V2V and V2I applications

The performance of both the EVA and ASA sample applications have been evaluated through several simulations, focused on Application-layer KPIs. The main aim is to showcase how ms-van3t can be used to efficiently and easily gather application KPIs, showing how beneficial results can be obtained from simple applications.

Furthermore, the simulation results also demonstrate the benefits brought by V2X connectivity to road safety and traffic efficiency applications, based on the exchange of standard-compliant messages.

Concerning the EVA application, it has been evaluated considering different vehicle densities, from around 3 veh/km up to around 13 veh/km. Each measured metric comes from 100 simulations (each with a different mobility trace), including two emergency vehicles, one per direction of travel. We gathered then the average speed of the two emergency vehicles, over all the simulations, in three cases: *(i)*

when using IEEE 802.11p, *(ii)* when using LTE-V2X Mode 4 and *(iii)* when no V2V communication is in place (i.e., no EVA is deployed), to provide a comparison baseline.

It should remarked that the main aim of this Section is to exemplify and showcase the capabilities of ms-van3t when simulating V2V and V2I applications, and not to analyze in detail a novel V2X use case. We thus decided to limit the analysis to up to 13 veh/km, as opposed to the more extensive comparison between LTE-V2X and IEEE 802.11p, presented in Section 4.1.6.

Figure 4.5 depicts the average emergency vehicles speed as a function of the vehicle density, with 95% confidence interval over the 100 simulations.

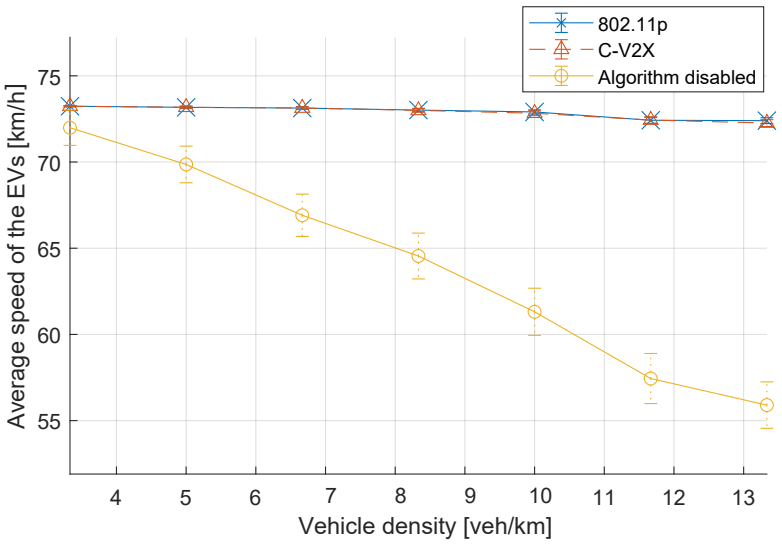


Figure 4.5: Results of the evaluation of the EVA application: the adoption of the algorithm determines a 30% increase of the average speed.

As can be seen, EVA brings significant advantages when deployed. Indeed, when the algorithm is activated, the emergency vehicles can almost always keep their maximum speed, no matter the vehicle density or traffic pattern, up to at least 13 veh/km. Instead, without EVA the emergency vehicle speed can be heavily affected by the traffic caused by regular vehicles, with an average speed reduction of up to more than 10 km/h.

It should also be noted how, for the considered vehicle densities, the selected access technology does not make a significant difference for the algorithm effectiveness, at least from an Application KPI point of view. More evident differences are expected to be observed when, as described in Section 4.1.6, more congested scenarios and network KPIs are taken into consideration.

Concerning instead the ASA application, we focused on a safety scenario. We introduced, in the road topology depicted in Figure 4.3, two modified traffic lights always showing a green light to incoming vehicles, to simulate either a distracted

or reckless driver, or an intersection regulation system failure. In this way, the vehicles will reach the intersections without slowing down due to the traffic lights, and, in some cases, they will be unable to brake in time and avoid a collision. Two dangerous intersections are thus introduced to define a low speed area, inside which additional care must be taken. We then observed the average number of collisions with and without ASA, considering both LTE and IEEE 802.11p as access technologies.

Since our goal is to provide a sample application to show the potentiality of our framework, a collision avoidance scheme, in this context, has not been implemented; this solution also allows us to define a metric (i.e., the number of collisions when no traffic lights are regulating the traffic) to evaluate our sample application, both when a speed reduction is mandated through DENMs and when, instead, the algorithm is disabled and all the nodes are able to keep their maximum speed in correspondence to the dangerous intersections.

The speed advised by ASA for the dangerous low speed area has been set to 25 km/h, while the vehicles can keep a maximum speed of 100 km/h when outside this area or when ASA is disabled.

As in the previous case, the results have been gathered by launching 100 simulation with different traffic patterns for each vehicle density. The numbers of collisions have then been averaged over all the simulations, with 95% confidence intervals.

The most significant results are reported in Figure 4.6.

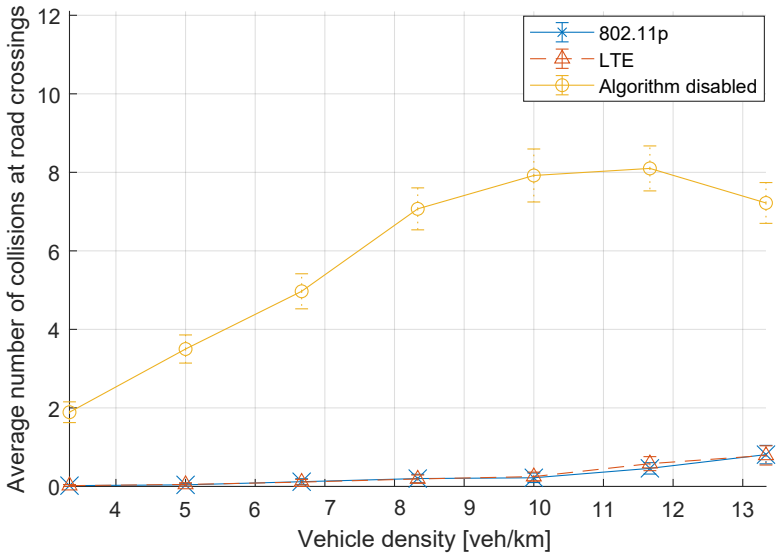


Figure 4.6: Results of the evaluation of the ASA application.

As can be seen, ASA can bring noticeable advantages when a dangerous low speed zone is defined and special speed limits should be enforced. Thanks to the

timely reception of DENMs, vehicles entering the dangerous zone can limit their speed in the proximity of the faulty intersections and avoid the great majority of collisions. Indeed, on average less than 1 collision per simulation occurs when ASA is activated, for all the selected vehicle densities. This is not the case when no ASA is deployed, leading up to, on average, 8 collisions per simulation, when the density is between 11 and 12 veh/km. It is also possible to notice how the average number of collisions, without any speed limitation, decreases when the density grows over 12 veh/km. This is due to a more congested traffic condition which forces the vehicles to partially slow down, reducing the overall number of collisions at the dangerous intersections.

4.1.5 Validation of the emulation capabilities

Several interoperability tests have been performed with the aim of validating the emulation feature of ms-van3t. Most of them involved the usage of a commercial ETSI ITS-G5 stack, specifically, of the latest releases of the V2X SDK by Cohda Wireless, typically used in conjunction with their MK5 boards.

The Cohda SDK comes with a Linux virtual machine containing a vehicle emulator, which can be used to run Cohda V2X applications on a PC before deploying them on a real vehicle. Thanks to this virtual machine, it was possible to run the Cohda Wireless stack on standard laptops, without the need of deploying all the software on a physical MK5 board.

Consequently, we set up two laptops in our laboratory to validate the emulation capabilities of our framework: *(i)* one laptop, running Ubuntu 18.04 LTS, was set to run a Cohda Wireless virtual environment to emulate one vehicle and send/receive V2X packets over a physical interface, while *(ii)* the second device, running Ubuntu 20.04 LTS, was used to run ms-van3t. Both devices were connected via Gigabit Ethernet to our laboratory network. Figure 4.7 shows the experimental setup for the interoperability tests between ms-van3t and the commercial Cohda Wireless stack.

Even though no wireless channel was established between the two laptops, these tests allowed us to fully validate the capability of ms-van3t to communicate with real nodes and commercial V2X stacks thanks to standard-compliant messages. It should be highlighted how replacing the Ethernet connection with a real hardware represents a seamless operation. For instance, it is possible to easily replace the laboratory LAN with a couple of devices capable of providing IEEE 802.11p connectivity, such as the open DSRC platform presented in Section 3.2.

The configuration of a simple scenario with few vehicles in SUMO allowed us to perform a first set of interoperability tests. We observed that the reception from, as well as the transmission to, the commercial stack, were performed in accordance to ETSI standards. In particular, ms-van3t was able to receive all messages from the virtual vehicle running the Cohda Wireless SDK, and the Cohda Wireless stack

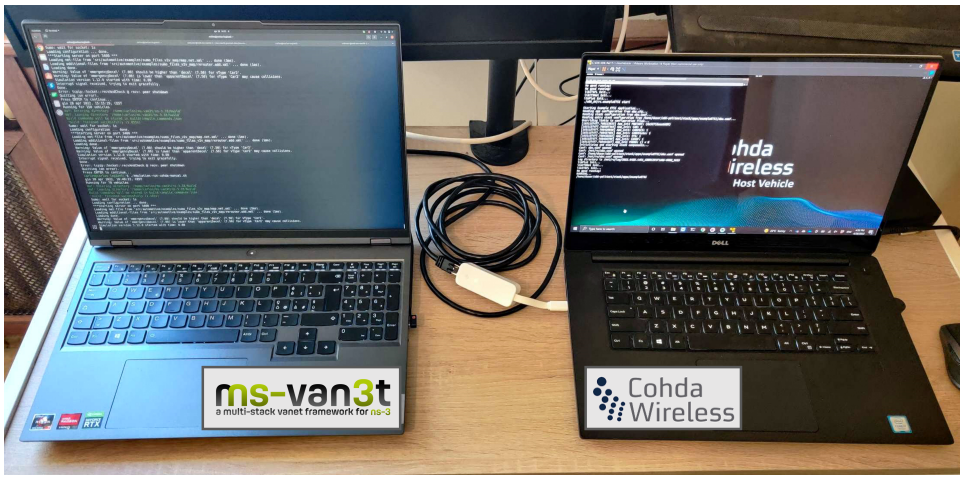


Figure 4.7: Laboratory setup for the interoperability tests between ms-van3t and a commercial, automotive-grade, ETSI ITS-G5 stack. These tests allowed us to validate the emulation feature available in ms-van3t.

was able to properly decode all messages coming from vehicles emulated by ms-van3t. One of the tests, involving the exchange of CAM messages between two vehicles (the Cohda virtual vehicle and a passenger car emulated by ms-van3t), is schematized in Figure 4.8.

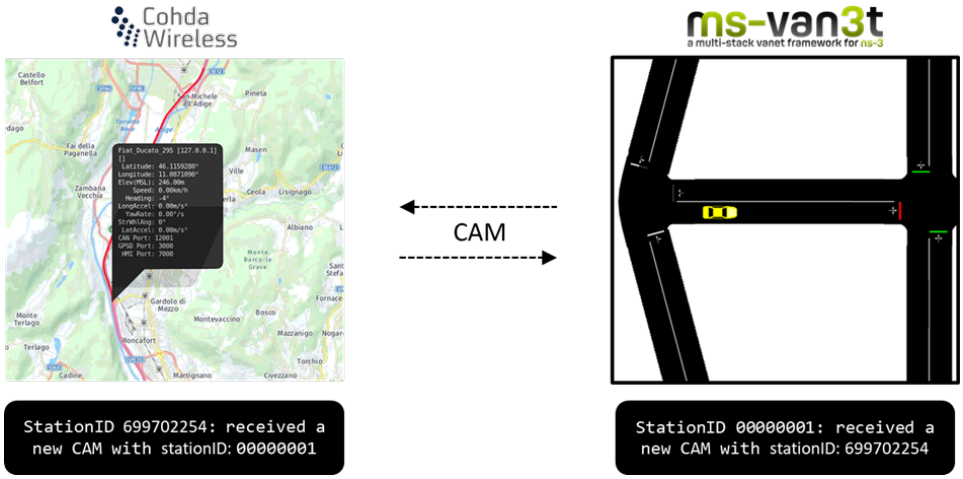


Figure 4.8: Scheme showing the communication between ms-van3t (configured to emulate one vehicle with stationID = 1) and a commercial ETSI ITS-G5 stack (configured to emulate a vehicle with stationID = 699702254).

The picture shows, on the left, the Cohda Wireless emulator GUI together with the commercial stack output when receiving CAMs from ms-van3t. On the right, instead, the Figure depicts the ms-van3t SUMO GUI, with an emulated vehicle

sending CAMs to the Cohda Wireless stack through the physical Ethernet interface of the second laptop. The same vehicle is also receiving CAMs from the commercial stack, and properly decoding them, displaying the output shown in the bottom-right part of the picture. It is worth highlighting that the Cohda stack was set up to disable the security protocols, since ms-van3t, which can currently only transmit non-secured GeoNetworking messages, does not yet have those protocols fully implemented.

A second set of tests was instead aimed at evaluating the capability of ms-van3t to emulate a large number of vehicles in real-time.

To this aim, a more complex scenario has been set up on the laptop running ms-van3t in emulation mode, starting from the scenario presented earlier for the EVA application. Here, one of the simulated vehicles is considered as a Vehicle Under Test (VUT). Every time the VUT receives a message from another vehicle inside the simulation, after going through the simulated wireless link and being correctly received by the simulated MAC layer, it relays the message to a physical network interface. The aim of these tests was to analyze the performance of the framework to emulate a high number of vehicles sending messages to a VUT for HIL testing. With this approach even if there is no wireless channel between the two physical network devices, the propagation loss and channel congestion are being handled by ns-3 accurate models. We used the IEEE 802.11p model, with a transmission power of 23 dBm for all emulated OBUs.

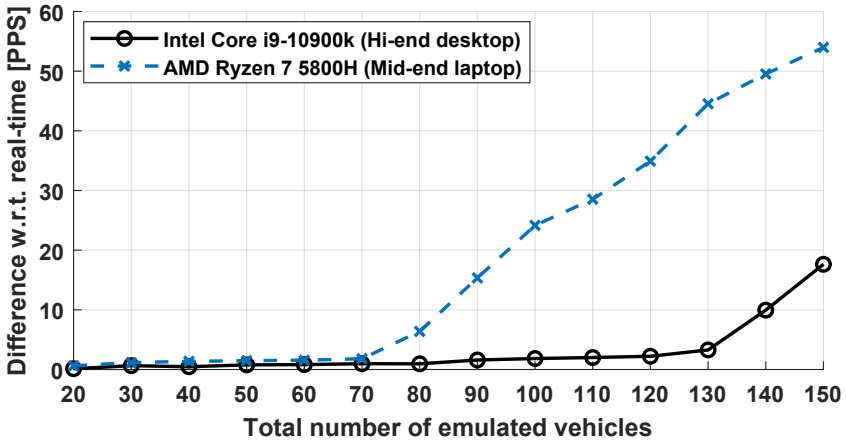
Additionally, an ad-hoc application has been developed for the Cohda SDK device, sending CAMs to ms-van3t and receiving messages from all emulated vehicles. This application has been programmed to return the average number of received Packets-Per-Second (PPS) during the whole test time, and each emulation session has been set to last for 1000 s (i.e., around 16 minutes), for each ms-van3t vehicle density.

The obtained PPS values, for different numbers of vehicles, has been then compared to a set of *expected values*. These expected values have been obtained by running pure simulations of the ms-van3t scenario presented above, which do not require any real-time constraint. These values thus represent the ideal PPS values which should be obtained if everything was running in real-time when in emulation mode. When the number of vehicles emulated by ms-van3t increases too much, a noticeable difference is expected between the obtained and expected values, showing the limit after which ms-van3t starts to slow down and cannot support real-time emulation anymore.

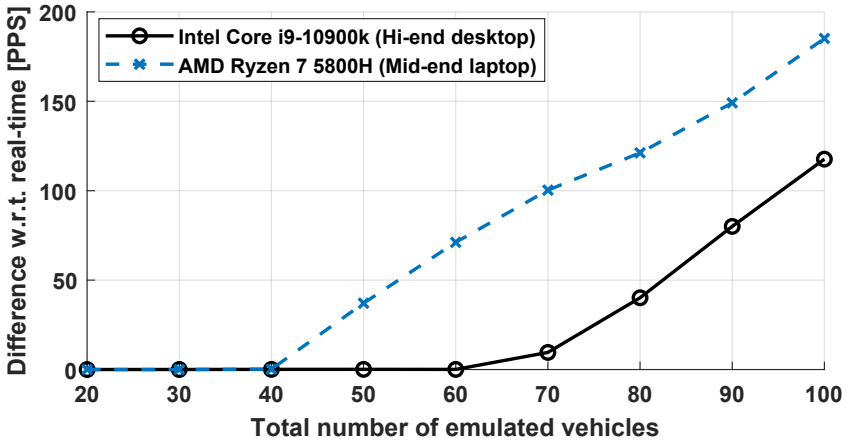
All tests have been performed for two different devices running ms-van3t, i.e., both the laptop mentioned earlier, with an AMD Ryzen 7 5800H CPU and a mid-performance hardware setup, and a high-performance desktop PC with an Intel Core i9-10900K, to assess the limits when providing a substantial amount of computing resources. We performed the same measurements both while keeping

standard-compliant dynamic CAM frequency management [63], and when such frequency is set to 10 Hz (the highest possible frequency according to ETSI [63]), to investigate a worst-case scenario.

The results are depicted in Figure 4.9, which shows the average difference between the achieved PPS on the Cohda stack and the expected PPS, as a function of the total number of vehicles emulated by ms-van3t. A difference of zero means that ms-van3t is emulating the corresponding number of vehicles in real-time.



(a) CAM dynamic frequency management [63]



(b) CAM fixed 10 Hz frequency (worst-case)

Figure 4.9: Average difference between the achieved PPS on the Cohda stack and the expected PPS, for different CPUs and as a function of the total number of vehicles emulated by ms-van3t.

As expected, the capability of emulating vehicles in real-time depends on the available hardware resources (mainly, on the CPU). Indeed, a high-power Intel Core i9 CPU is able to emulate, in real-time, up to around 120 vehicles when keeping the ETSI dynamic frequency management, and up to 60 vehicles with a

worst-case frequency of 10 Hz. These values are almost halved for a standard AMD laptop CPU, which is able to emulate, in real-time, respectively up to 70 vehicles in the first case, and up to 40 vehicles in the second case. Even though the laptop CPU can emulate fewer vehicles in real-time, if compared to an Intel Core i9 CPU, the outcome is nevertheless quite good. Indeed, the results show that, with ms-van3t, even a typical laptop CPU can emulate a sizable number of vehicles without any slow down, as if the emulated nodes were actual vehicles sending messages in real-time.

During our experimental sessions, we also observed that the main impacting factor is the single-core performance of the CPU on which ms-van3t is run. Therefore, the interoperability and real-time emulation tests, other than validating our framework, proved that:

- Given that new generations of more powerful CPUs are becoming available, ms-van3t may be able to manage even a larger number of emulated vehicles in real-time with the proper amount of resources.
- Future work may consider investigating a possible parallelization of the ms-van3t code, to better exploit the multi-core capabilities of modern CPUs.

4.1.6 Comparison between C-V2X and IEEE 802.11p

After the development of ms-van3t, we exploited its features to perform a series of simulations aimed at providing a comparison between the two main technologies for direct V2V/V2I communication, i.e., IEEE 802.11p and C-V2X.

The debate on which of the two protocol stacks is best suited to meet market needs is still open in the scientific community, where some argue that the solution proposed by 3GPP provides higher performance in terms of range and reliability [182], [183]. Moreover, the presence of two interfaces, namely *PC5* and *Uu*, suggests that C-V2X as a whole could be more suitable to enable reliable V2N applications.

On the other hand, supporters of IEEE 802.11p claim that C-V2X is still in its early development stage, with fewer validation tests in the field with respect to IEEE 802.11p. Indeed, the IEEE solution was first approved in 2009 and later amended in 2016 and 2020 and, in the meantime, there have been several performance assessment campaigns. Furthermore, as mentioned earlier, carmakers are starting to embed V2X communications based on DSRC on their vehicles, such as the Volkswagen Golf 8.

We propose in this thesis a comparison between the two technologies in a realistic scenario, thanks to ms-van3t, aimed at providing interesting insights on the advantages and disadvantages of the two solutions.

Concerning C-V2X, it has been evaluated thanks to the model embedded in our framework. We thus focused on LTE-V2X Mode 4 [169], while simulations

campaigns for NR-V2X Mode 2 are planned to start as soon as the respective model [170] will be fully integrated.

As network KPIs, we focused both on one-way latency and PRR, as described earlier. Each metric has been collected and averaged over 10 simulations lasting 200 seconds, each characterized by a different traffic pattern. Furthermore, a relatively wide range of vehicle densities has been considered, from around 3 veh/km to around 34 veh/km. 3 veh/km represents a very low density scenario, corresponding to a total of 9 vehicles travelling in the selected map. 34 veh/km corresponds instead to a relatively congested traffic condition, with a total of 102 vehicles all sharing the same wireless channel.

The EVA base map, with two lanes for each direction of travel, has been selected as a SUMO scenario for this evaluation campaign. Furthermore, IEEE 802.11p has been configured with a physical data rate of 3 Mbit/s, while C-V2X has been configured with an MCS of 20, an RRI equal to 20 ms and a probability P (as defined in Section 2.3.1) equal to 0. These settings were among the ones making both technologies perform the best in the selected scenario. Indeed, IEEE 802.11p usually suffers from a lower PRR, if compared to C-V2X with an equal transmission power level, and a data rate of 3 Mbit/s (corresponding to the simplest modulation) can help improving the number of packets which are successfully received. On the other hand, C-V2X Mode 4 is usually characterized by a higher latency, as highlighted for instance in [183], and a higher MCS (i.e, 20) should be able to guarantee a lower latency. Both IEEE 802.11p and C-V2X are configured to use the DSRC frequencies at 5.9 GHz.

Both technologies have been evaluated while considering three different transmission power levels, i.e., 20 dBm, 26 dBm and 30 dBm.

The most significant results related to the average one-way latency at 26 dBm are reported in Figure 4.10, as a function of the spatial vehicle density. It should be mentioned that very similar results were observed for the other tested power levels (i.e., 20 dBm and 30 dBm), which are thus not reported here.

As can be seen, both IEEE 802.11p and C-V2X Mode 4 show a stable and low latency, over all the tested vehicle densities. In particular, the technology achieving the best performance is IEEE 802.11p, with latency values around 0.5 ms, while C-V2X Mode 4 is characterized by values slightly above 16 ms, thanks to the selection of the minimum allowed RRI (i.e., 20 ms), together with the SPS mechanism to select the available resources for transmitting the V2X messages (mainly, CAMs). Except in the case of 3 veh/km for C-V2X Mode 4, in which, however, very few vehicles populate the scenario, the confidence intervals are fairly small too. This further suggests how both technologies can handle well and with a stable connection the exchange of CAMs following the ETSI dynamic frequency management [63], at least up to 33 veh/km.

The results related to the PRR are instead depicted in Figures 4.11 and 4.12, as a function of the baseline distance. Specifically, we selected three baseline values, i.e.,

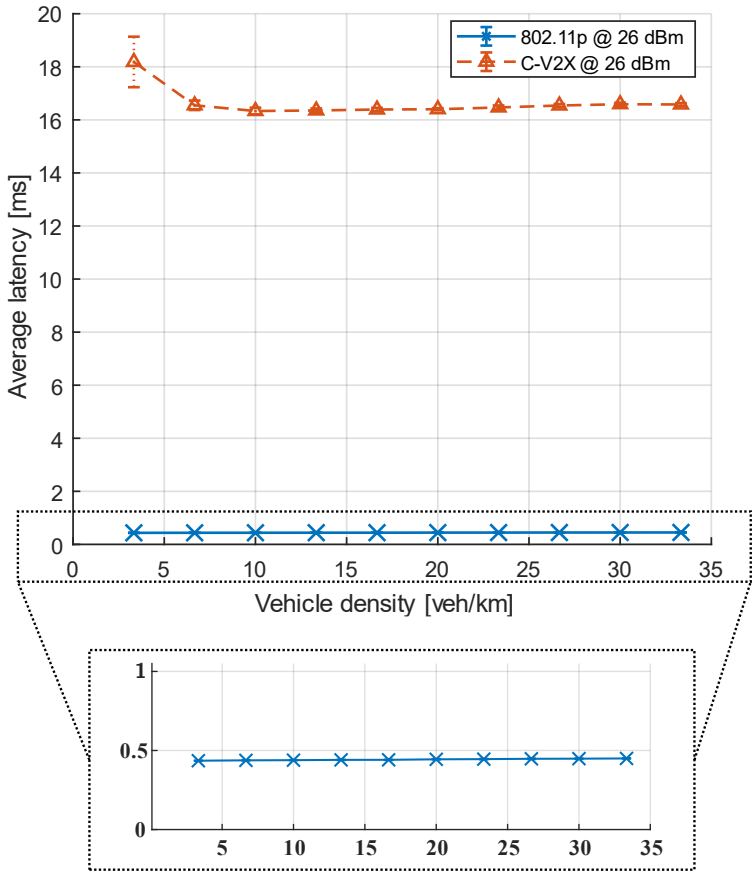


Figure 4.10: Average one-way latency as a function of the vehicle spatial density. The error bars represent 95% confidence intervals over 10 simulations with different traffic patterns.

100 m, 150 m and 200 m. With the aim of providing a compact representation, the two plots focus only on a single vehicle density, i.e., a medium density of 20 veh/km (Figure 4.11) and a higher density corresponding to 30 veh/km (Figure 4.12).

As can be seen, the difference between 20 veh/km and 30 veh/km is related to slightly lower PRR values for the higher density, for all the tested configurations. This is due to a higher channel congestion which affects both technologies, as more vehicles try to transmit their messages on the same shared medium.

The average PRR shows instead that C-V2X Mode 4 can correctly deliver a very high number of messages, in excess of 98%, at a maximum distance of 200 m. IEEE 802.11p exhibits instead worse results, especially when the OBUs are configured to transmit at 20 dBm, with the PRR dropping to less than 0.3 for a baseline of 200 m. The obtained values also show that IEEE 802.11p is characterized by a worst PRR for a high baseline value (i.e., 200 m). This is due to DSRC requiring a higher

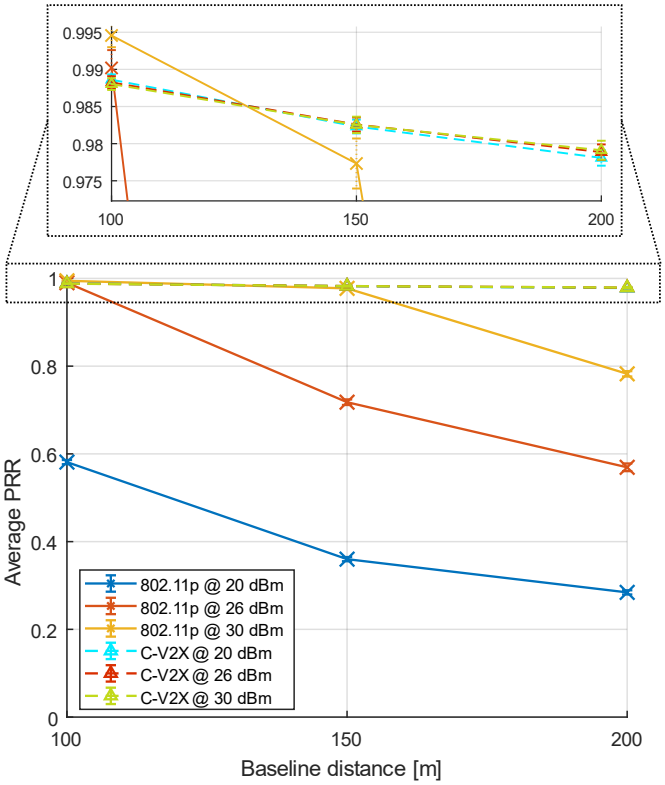


Figure 4.11: PRR as a function of the baseline distance from the sender. At the top the zoom in of the region around 0.96 and 1, where the advantages in using cellular-based technologies are more evident. Vehicle density: 20 veh/km.

power than C-V2X to reach all vehicles within the baseline. Indeed, the worst PRR values are obtained for a fairly low transmission power, i.e., 20 dBm. On the contrary, if a sufficient transmission power is leveraged and the baseline is properly set, IEEE 802.11p can deliver more than 99% of packets, as shown in the plots for a baseline of 100 m and a transmission power higher than 26 dBm.

It should be mentioned how the obtained results are strongly dependent on the propagation loss model adopted by the various simulation models: in our case, we kept the default configurations, with IEEE 802.11p adopting the Log-distance path loss model and C-V2X adopting WINNER B1 (urban microcell, as described in [178]). Nevertheless, the obtained trend retains generality and shows how IEEE 802.11p can provide the smallest latency, but it is outperformed by C-V2X when measuring the PRR. Furthermore, we observed how C-V2X requires a lower transmission power than IEEE 802.11p to reach all vehicles within a given baseline.

Finally, it is worth highlighting how the obtained results show that ms-van3t can be an easy-to-use, handy framework for simulating and emulating V2X scenarios, and gathering useful metrics such as latency and PRR.

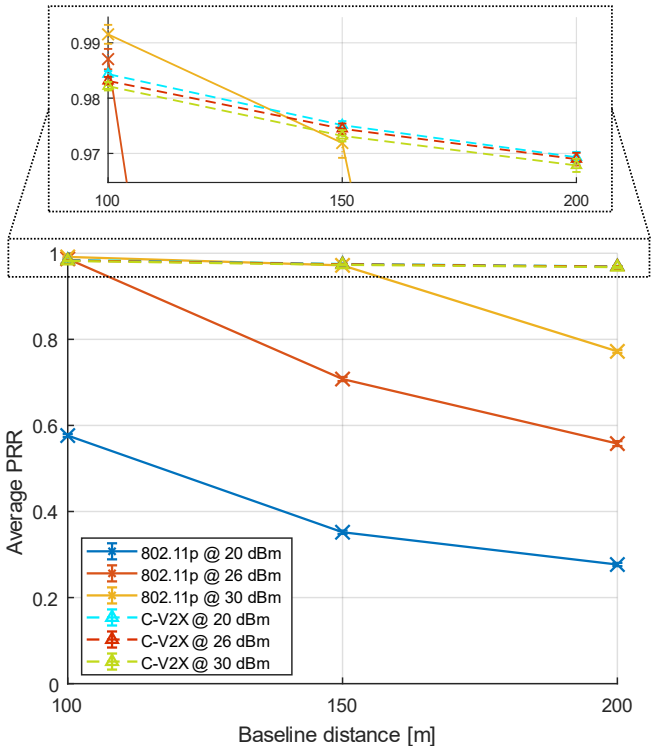


Figure 4.12: PRR as a function of the baseline distance from the sender. At the top the zoom in of the region around 0.96 and 1, where the advantages in using cellular-based technologies are more evident. Vehicle density: 30 veh/km.

4.2 Field testing: baseline characterization and performance evaluation of IEEE 802.11p

After the presentation of the ms-van3t framework for the evaluation of small to large scale scenarios through simulation and emulation, this Section focuses on several measurements in the field. These measurements have been performed in a static laboratory environment with the aim of providing a baseline evaluation of IEEE 802.11p in a real-world scenario, with real hardware and IEEE 802.11p-enabled WNICs.

In particular, we used the fully open platform described in Section 3.2, based on two PC Engines APU1 boards, mounting UNEX DHXA-222 mPCIe wireless modules.

The aim of these field tests is twofold: first, they provide several measurements and interesting data on a low-latency IEEE 802.11p communication in a real-world scenario, focusing on throughput, packet loss and receive power sensitivity. Then, they further validate our open DSRC platform and characterize the performance of

the selected UNEX DHXA-222 cards in a controlled environment.

The DSRC platform was complemented, for this set of measurements, with the installation of two 5 dBi dual-band PC Engines antennas for each embedded board.

The static tests presented in this Section are then completed by the tests performed in the field with two real vehicles, described in Section 4.3.

4.2.1 Throughput and packet loss measurements

This Section describes the main results related to UDP throughput and packet loss. UDP was selected as a reference protocol for the throughput tests as it is usually the IP-based protocol of choice for vehicular communication. Furthermore, it represents a connection-less protocol, like ETSI BTP in the ETSI Transport and Networking layers.

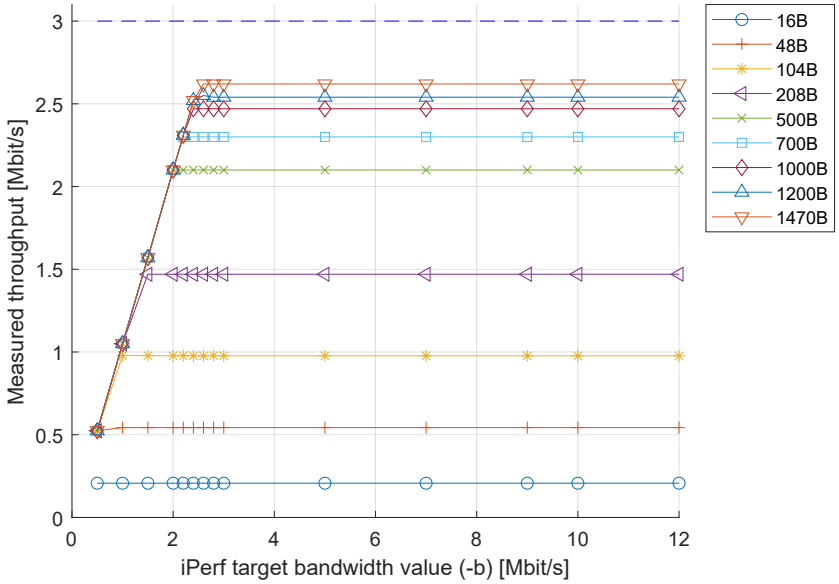
These measurements have been performed on the two APU boards by placing them near to each other (with a distance of ~16 cm between enclosures), such that the received power level was enough to provide an analysis in nearly "ideal" conditions. As open source tool, we made use of the patched version of iPerf 2 described in Section 3.2, in which we left the default AC for the test traffic (i.e., AC_BE).

The tests were carried out by selecting different physical data rates (among the mandatory ones, i.e., 3, 6 or 12 Mbit/s) and by varying the quantity of offered traffic and the payload size, from 16 B to 1470 B, which is the iPerf default value for large UDP test packets. Each test, providing a single data point, lasted for 60 seconds.

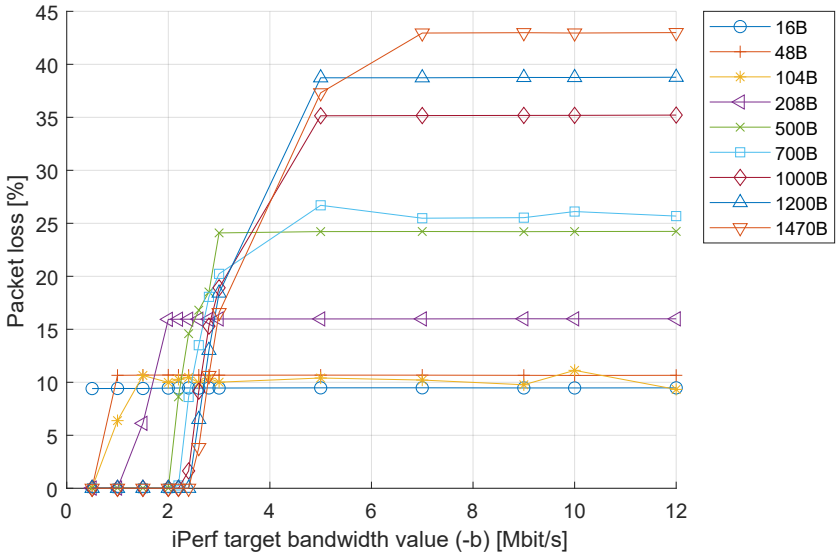
The results for the physical data rate of 3 Mbit/s are depicted in Figure 4.13, where the top plot shows the measured throughput as a function of the offered traffic, while the bottom plot shows the packet loss percentage.

As expected, the maximum reachable throughput depends on the UDP payload size. Indeed, higher payload sizes correspond to a higher measured throughput. In particular, with a 1470 B payload size, we were able to reach a maximum throughput of 2.62 Mbit/s. This trend also shows how, throughput-wise, the best performance is reached with large packet sizes.

Looking instead at the packet loss trend, it tends to increase when offering too much traffic with respect to the network capacity, while it is almost always 0% when the offered traffic is lower than the maximum reachable throughput. For instance, if 4 Mbit/s are offered when the maximum reachable throughput for that packet size is 2.62 Mbit/s, some packet loss will be observed. The only exception is represented by the 16 B case, which is, in any case, a very low, non-optimized value. These values are indeed due to packets dropped in the buffers inside the kernel, which are often full, and not due to losses over the air. This is a peculiar behaviour of the "unconnected" OCB mode implemented as part of the Linux kernel and of the *ath9k* driver. When transmitting packets in OCB mode, if the application pushes more



(a) Reachable throughput for a 3 Mbit/s physical data rate



(b) Packet loss for a 3 Mbit/s physical data rate

Figure 4.13: Throughput and Packet loss measurements for 3 Mbit/s of physical layer data rate, with different UDP payload sizes and offered traffic values. The dotted blue line represents the physical data rate (i.e., the limit which cannot be overcome by the reachable throughput values).

packets than what can be accommodated by the driver/*mac80211* buffers, some internal drops may be observed. Indeed, by analysing the kernel behaviour in OCB mode, it is possible to observe how `ieee80211_txq_enqueue()`, i.e., the function

to queue new packets for transmission inside *mac80211*, is called for every packet generated by iPerf, even when offering more traffic than the maximum reachable throughput. The function used by the *ath9k* driver to dequeue packets from the *mac80211* queues (i.e., `ath_tid_pull()`) appears to be called, instead, only when the transmission on the physical channel can happen, and it is thus called only for the packets which are actually transmitted (and received by the other device). When offering more traffic than what can be achieved by the network, few packets appears thus to be internally dropped between *mac80211* and the driver.

The specific values reported in Figure 4.13b are determined by the interaction between the application (i.e., in this case, iPerf) and the lower layers, which happens through buffers at the socket level. When these buffers get full due to the offered traffic being too high, the kernel will start dropping packets, until the buffers start to have some free space again.

It is also worth noticing how the packet loss trend, as a function of the offered traffic, is less linear than the throughput one. This is due to two effects combining together, i.e., the WNIC transmitting packets over the wireless medium at a given modulation (and physical data rate), and the effect of the UDP buffers. As these packet loss values are due to buffer losses, and not due to losses in the wireless medium, they are overall less interesting for our analysis, and the results for the other two data rates are thus omitted from this thesis. The obtained curves are in any case very similar to the 3 Mbit/s one, with the losses increasing as the offered traffic increases over the maximum reachable throughput point.

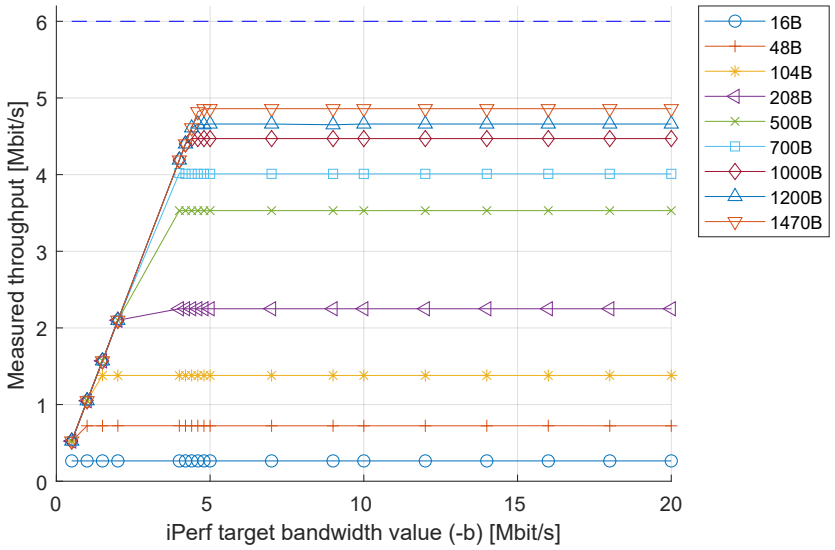
The throughput results for the two other mandatory IEEE 802.11p data rates are instead depicted in Figure 4.14.

As can be seen, the throughput, as a function of the offered traffic, follows a similar trend as in the 3 Mbit/s case. In particular, for a packet size of 1470 B, we were able to reach 4.86 Mbit/s for 6 Mbit/s and 8.42 Mbit/s for 12 Mbit/s.

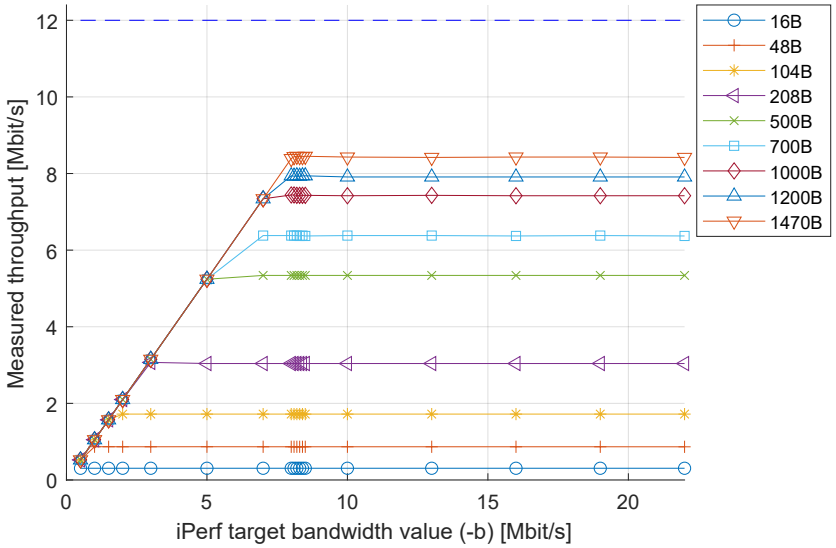
These values can be compared to the theoretical achievable Maximum Throughput (MT), as defined in [184], taking as reference the AC_BE traffic class.

Specifically, we can define the following parameters and values [184]:

- T_P: Transmission time of the physical layer preamble.
- T_{PHY}: Transmission time of the SIGNAL field in the Physical Layer Convergence Protocol (PLCP) header, which is always transmitted at the basic BPSK rate.
- T_{SYM}: Transmission time for an OFDM symbol.
- $L_{SERVICE}$: Length, in bits, of the SERVICE field in the IEEE 802.11 packet [185].
- L_{TAIL}: Length, in bits, of the Tail field in the IEEE 802.11 packet [185].
- L_{H_DATA}: Length, in bytes, of the MAC header.



(a) Reachable throughput for a 6 Mbit/s physical data rate



(b) Reachable throughput for a 12 Mbit/s physical data rate

Figure 4.14: Throughput measurements for physical data rates of 6 Mbit/s and 12 Mbit/s, with different UDP payload sizes and offered traffic values. The dotted blue lines represent the physical data rate (i.e., the limit which cannot be overcome by the reachable throughput values).

- L_{DATA}: Length, in bytes, of the payload (excluding MAC and PHY headers, but including the other layers such as UDP, IPv4 and LLC).
- $L_{DATA_{Appl}}$: Length, in bytes, of the application layer payload (i.e., the actual

useful data).

- L_{ACK}: Length, in bytes, of IEEE 802.11 Acknowledgements (ACKs).
- $N_{DBPS_{DATA}}$: Number of data bytes per OFDM symbol (for data packets).
- $N_{DBPS_{ACK}}$: Number of data bytes per OFDM symbol (for ACKs).
- T_{D_DATA}: Transmission time for a data packet, considering the transmission cycle of EDCA.
- T_{D_ACK}: Transmission time for an ACK, considering the transmission cycle of EDCA.

The aforementioned parameters have been set to the values reported in Table 4.2, for IEEE 802.11p [46].

Parameter	Value
T_P	$32\mu s$
T_{PHY}	$8\mu s$
T_{SYM}	$8\mu s$ (as reported in Table 17-16 on [46])
$L_{SERVICE}$	16 bits [185]
L_{TAIL}	6 bits [185]
L_{H_DATA}	28 bytes
L_{DATA}	Depends on the actual packet
L_{ACK}	14 bytes [184]
$N_{DBPS_{DATA}}$	Depends on the physical data rate for data packets [46]
$N_{DBPS_{ACK}}$	Depends on the physical data rate for ACKs [46]

Table 4.2: Parameters for the computation of the theoretical achievable Maximum Throughput (MT) of IEEE 802.11p.

As defined in Table 17-4 in the IEEE 802.11-2020 standard [46], N_{DBPS} should be set to 24 for a physical data rate of 3 Mbit/s, 48 for 6 Mbit/s, and 96 for 12 Mbit/s. We also set the L_{DATA} value to match the size of the packets used by iPerf when achieving the best throughput, i.e., 1470 B, plus 8 B of LLC header, 8 bytes of UDP header and 20 bytes of IPv4 header, for a total of 1506 B. The application layer payload $L_{DATA_{Appl}}$ has been set, instead, to 1470 B, as this is the actual value used by iPerf to the report the achievable UDP throughput.

With the aim of computing the theoretical throughput in ideal channel conditions (i.e., without errors), we used the following formulas [184]:

$$T_{D\_DATA} = T_P + T_{PHY} + T_{SYM} \cdot \lceil \frac{L_{SERVICE} + L_{TAIL} + 8 \cdot L_{H\_DATA} + 8 \cdot L_{DATA}}{N_{DBPS_{DATA}}} \rceil \tag{4.2}$$

$$T_{D\_ACK} = T_P + T_{PHY} + T_{SYM} \cdot \lceil \frac{L_{SERVICE} + L_{TAIL} + 8 \cdot L_{ACK}}{N_{DBPS_{ACK}}} \rceil \tag{4.3}$$

$$MT = \frac{8 \cdot L_{DATA_{Appl}}}{AIFS(AC) + \frac{CW_{min} \cdot aSlotTime}{2} + T_{D\_DATA} + aSIFSTime + T_{D\_ACK}} \tag{4.4}$$

As AC_BE is considered, CW_{min} has been set to 15 and $AIFS(AC)$ to 110μs. Furthermore, it should be recalled how, in IEEE 802.11p, $aSlotTime$ is equal to 13μs and $aSIFSTime$ to 32μs. The ACK physical data rate has been to be equal to the data physical data rate, as this is the current default setting when changing the used modulation in our DSRC platform.

The computation led to a MT equal to 2.68 Mbit/s for a physical data rate of 3 Mbit/s, 5.07 Mbit/s for 6 Mbit/s and 9.15 Mbit/s for 12 Mbit/s. It it thus interesting to observe how the achievable throughput measured on the field is slightly lower than the theoretical MT, in spite of the ideal conditions in which tests have been performed, with no external interference. This difference could be due to a sum of small inefficiencies in the system and inside iPerf, when each packet is passed from the application layer down to the MAC layer, and then transmitted over the shared wireless channel.

4.2.2 Measurements over different Access Categories (multiple flows)

After the baseline evaluation of throughput and packet loss, we performed several measurements aimed at assessing the performance of IEEE 802.11p, through our DSRC platform (based on the UNEX DHXA-222 cards), when selecting different Access Categories.

The tests were performed by launching an iPerf client and an iPerf server on each of the APU boards, with the client on one board sending packets to the server on the other board, and vice versa. The two flows of traffic were configured to use different Access Categories, with one flow being the data under measurement (i.e., the actual traffic of interest), and the other acting as interfering traffic. We then collected the reachable throughput returned by iPerf, and estimated the connection stability, computed as the maximum percentage variation in the throughput values reported by iPerf every 2 seconds, according to the following formula:

$$\%_{max} = 100 \cdot \frac{throughput_{max} - throughput_{min}}{throughput_{max}} \tag{4.5}$$

A low $\%_{max}$ value represents a quite stable connection, while a high value means that the connection cannot guarantee a stable throughput over time. A value of 100% corresponds to being unable to communicate for a full 2 seconds interval (i.e., $throughput_{min} = 0$).

The throughput results are depicted in Figure 4.15 for all the three mandatory data rates, together with the connection stability for a data rate of 3 Mbit/s. The stability values for the other data rates proved to be very similar to the ones at 3 Mbit/s, and they are thus not explicitly shown here.

As can be seen, when a board is transmitting data with a higher priority AC, such as AC_VI or AC_VO, it can always reach a higher throughput, especially when the interfering traffic is using a low priority AC (e.g., AC_BK or AC_BE). Furthermore, as expected, when a lower priority AC is used, and the interfering traffic is characterized by a high priority traffic class, such as AC_VO, a drastic throughput reduction is observed. Selecting a high priority AC also helps to improve the connection stability, no matter the traffic class of the interfering traffic. This can be seen by looking at the AC_VI and AC_VO lines in Figure 4.15d.

Another important consideration is related to the fact that, when the tested and the interfering flows are using the same traffic class, the channel usage is fair. This can be seen in Figures 4.15a, 4.15b and 4.15c by drawing a horizontal line that could be ideally traced around 1.2 Mbit/s for 3 Mbit/s, 2.3 Mbit/s for 6 Mbit/s and 4 Mbit/s for 12 Mbit/s.

4.2.3 Measurements over different Access Categories (single flow)

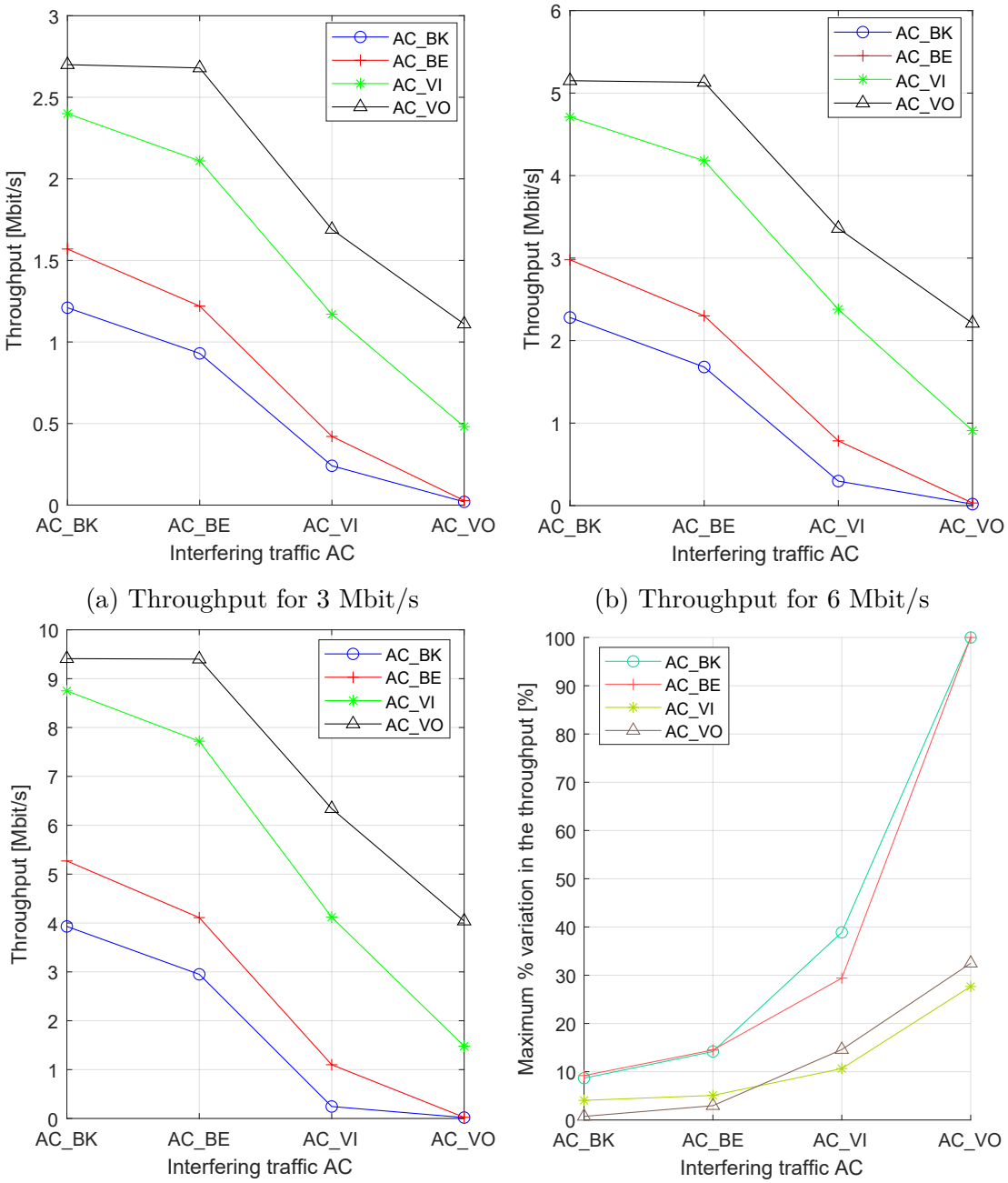
We also performed a set of throughput measurements with a single traffic flow between two APU boards equipped with our open DSRC platform. In our setup, one board was acting as iPerf client and sending test traffic to an iPerf server running on the second board. iPerf was configured to try to offer a higher amount of traffic with respect to each of the physical data rates, with the aim of testing the maximum reachable throughput for each AC.

We varied the AC used by the client and the physical data rate, considering again the physical data rates mentioned earlier (i.e., 3 Mbit/s, 6 Mbit/s and 12 Mbit/s).

The results are depicted in Figure 4.16 and reported in Table 4.3.

As can be inferred from the plot and the Table, the reachable throughput values increase as the traffic priority is increased, from AC_BK to AC_VO. This is due to smaller contention window sizes and shorter AIFS times for the higher priority traffic. Indeed, as reported in Table 2.2, the AIFS values range from 149 μs for AC_BK to 58 μs for AC_VO.

Other than providing a baseline characterization of prioritized IEEE 802.11p traffic, the obtained results allowed us to validate the EDCA functionalities of our open DSRC platform.



(a) Throughput for 3 Mbit/s

(b) Throughput for 6 Mbit/s

(c) Throughput for 12 Mbit/s

(d) Connection stability for 3 Mbit/s

Figure 4.15: Reachable throughput (Mbit/s) and connection stability ($\%_{max}$) when using different ACs, for different physical data rates.

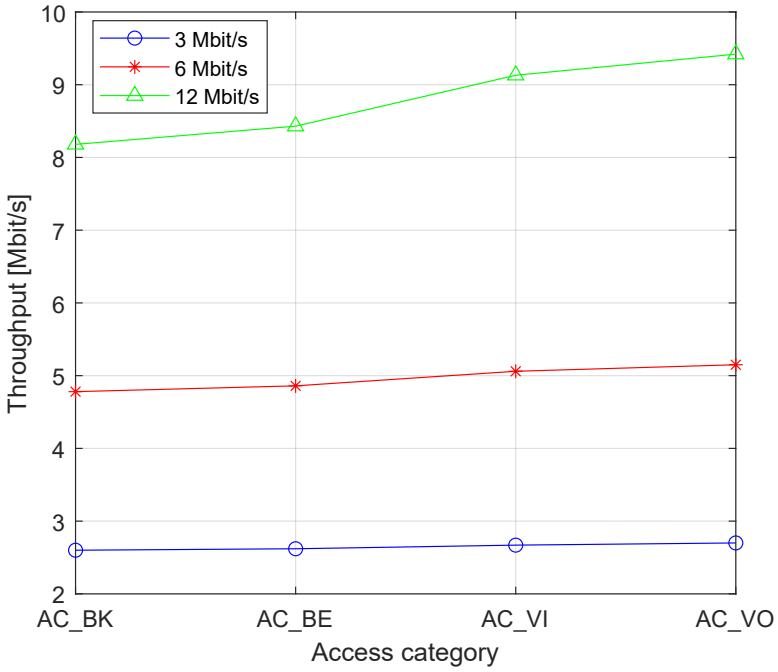


Figure 4.16: Measured throughput with a single traffic flow, at different ACs and physical data rates; payload length: 1470 B, offered traffic: 10 Mbit/s.

4.2.4 Received power and connectivity measurements

The final measurement campaign was aimed at evaluating the received power levels and reachable throughput, with a single stream over a given traffic class. As a reference AC, we selected the default one in the Linux kernel, i.e., AC_BE. This allowed us to evaluate the behaviour of IEEE 802.11p in presence of obstacles and signal reflections, and to assess the received power sensitivity of the UNEX DHXA-222 WNICs.

As in the previous case, an iPerf client was launched on one board, sending UDP traffic to the second board running an iPerf server. The boards were placed in different points of our laboratory, at increasing distances, letting us correlate the achievable throughput with the average received power level in dBm, as reported by the `iw` tool.

Indeed, we considered an average signal level value returned by the driver. In this regard, a few remarks are in order:

- Oscillations in the reported values of the order of 1 dBm were always present, so these values have to be always taken with at least a ±1 dBm precision.
- There was a particular situation in which the same average received signal level (-88 dBm) was detected in two slightly different positions, with almost

Data rate	Modulation	AC	Measured throughput [Mbit/s]	Maximum variation [%]
3 Mbit/s	BPSK	AC_BK	2.60	0.3846%
3 Mbit/s	BPSK	AC_BE	2.62	0.3817%
3 Mbit/s	BPSK	AC_VI	2.67	0.3731%
3 Mbit/s	BPSK	AC_VO	2.70	0.0000%
6 Mbit/s	QPSK	AC_BK	4.78	0.4175%
6 Mbit/s	QPSK	AC_BE	4.86	0.4107%
6 Mbit/s	QPSK	AC_VI	5.06	0.3945%
6 Mbit/s	QPSK	AC_VO	5.15	0.3876%
12 Mbit/s	16-QAM	AC_BK	8.18	0.6090%
12 Mbit/s	16-QAM	AC_BE	8.43	0.7092%
12 Mbit/s	16-QAM	AC_VI	9.13	0.4372%
12 Mbit/s	16-QAM	AC_VO	9.42	0.6363%

Table 4.3: Reachable throughput for each IEEE 802.11p Access Category and modulation.

the same ±1 dBm oscillations. It is likely that the true received power was a little higher in one point with respect to the other, looking at how better results, in terms of throughput, could be achieved. In order to discriminate the two points and considering that probably the true power was a little higher in one of the two, we assigned it a value of *-87.5 dBm*.

As a reference, we set a 10 dBm transmit power, in OpenWrt-V2X. This power level allowed us to easily move the boards inside the laboratory, measuring all the range of received power levels (up to -91 dBm) without the necessity of maintaining a very short or long distance between the boards.

Figure 4.17 shows the layout of our laboratory. The client board was kept in the position marked with "C", while the server board was moved in the points marked with the red circles and with the numbers from 1 to 5 and the "P" letter.

Each position marked by a red circle and a number corresponds to a different received power level with the client transmitting at 10 dBm, i.e., **(1)** -83 dBm, **(2)** -87 dBm, **(3)** -88 dBm, **(4)** -89 dBm and **(5)** $<$ -90 dBm. Notice that one closed door was located between the client and the server board, in order to analyze the effect of indoor NLOS, with the signal being affected by several obstacles. The layout also shows a farther away point, labeled "P". This point is located where the boards could still communicate with each other at a data rate of 6 Mbit/s, by

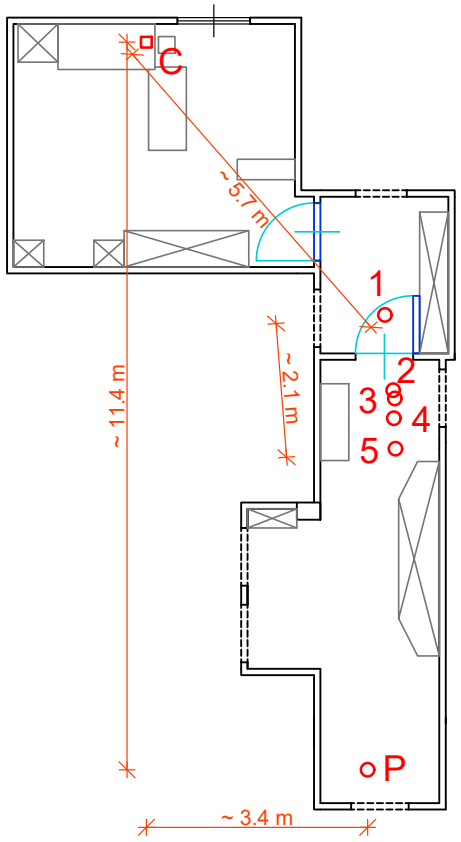


Figure 4.17: Layout of the laboratory for the receiver power and throughput measurements. The main obstacles are shown as grey boxes, with the crossed ones being the bigger pieces of furniture.

setting, inside OpenWrt-V2X, a transmission power of 18 dBm (i.e., the maximum supported by the UNEX DHXA-222 modules). The communication could happen with a closed door in between, with a minor throughput drop; however, when closing also the second door, which is shown as open in the map, the connection became quite unstable.

Figure 4.18 shows instead the achievable throughput as a function of decreasing received power values, taking as a reference a physical data rate of 6 Mbit/s.

It is possible to observe how, in static conditions, the connection remains stable, with a quite high reachable throughput, until the received power drops below -87 dBm. Below this value, the throughput experiences an abrupt drop until the connection can no longer be established. Furthermore, when the signal is weaker, small distance variations are sufficient to cause oscillations in the received power and throughput average values. Finally, as depicted in the plot, we were never able

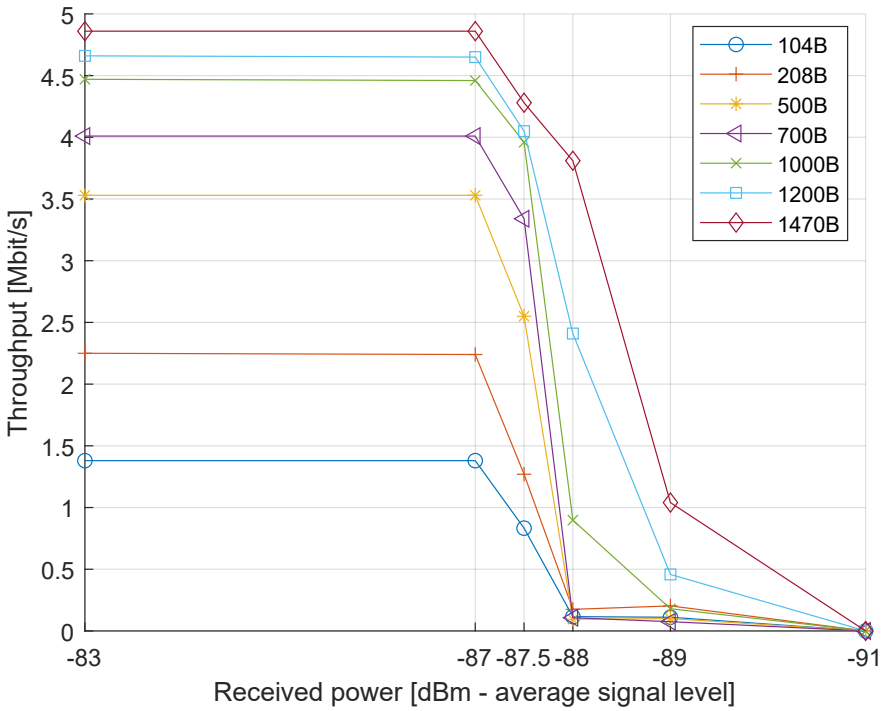


Figure 4.18: Throughput measurements for different values of received power; physical data rate: 6 Mbit/s, payload length: 1470 B, txpower: 10 dBm, offered traffic: 10 Mbit/s.

to establish a connection with an average signal power, reported by the system, of -91 dBm or less, thus assigning to these points a throughput of 0 Mbit/s.

4.3 Field testing: experimental assessment of IEEE 802.11-based V2I technologies

After perming several measurements in our laboratory, this Section presents the results of a campaign of dynamic field tests involving a number of devices implementing different amendments of the IEEE 802.11 family of standards. These tests have been performed by equipping two real vehicles with open platforms for V2X connectivity, relying both on technologies specifically developed to support vehicular communications, i.e., IEEE 802.11p, and on promising technologies, such as mmWave (IEEE 802.11ad) and high-power standard Wi-Fi (IEEE 802.11ac with a maximum output power of 30 dBm, plus the antenna gain). As mentioned in Section 2.4 mmWave represents indeed a very promising technology for the future generations of connected and autonomous vehicles.

The main goals of this set of field tests, which complement the ones presented

in Section 4.2, can be summarized as follows:

- Test, in the field and with real vehicles, the capability of IEEE 802.11p, 802.11ac and 802.11ad to provide vehicles with V2I connectivity, and the quality thereof.
- Compare the three aforementioned IEEE 802.11-based technologies.
- Perform measurements in the field by relying on the open DSRC platform described in Section 3.2, together with other off-the-shelf, low cost hardware running open source software.
- Analyze the impact of different configurations, and the advantages or disadvantages in choosing one technology over the others. Indeed, the tested technologies differ in some fundamental aspects, such as the maximum transmission power, the operating frequency, and the antenna directivity.

The main performance metrics, on which we focused, are service availability (i.e., the maximum achievable communication distance), communication latency, RSSI (Received Signal Strength Indicator, linked to the static received power measurements), and UDP throughput. As mentioned earlier, UDP was chosen as a reference transport protocol as it appears to be better suited to IP-based vehicular communications, as opposed to TCP, which requires an initial handshake before the data transfer can start.

All the results have been collected through properly equipped vehicles and increasing the distance between the communicating devices, so as to assess the real-world performance of the various access technologies. The measurements have been performed in scenarios both with Line-Of-Sight (LOS) and Non- Line-Of-Sight (NLOS) conditions.

To the best of our knowledge, this is also the first time mmWave at 60 GHz is compared, in the field, with other V2X technologies, i.e., IEEE 802.11p, by exploiting low cost and open platforms and focusing on V2I connectivity.

It is worth mentioning that, even if our measurements focus on V2I connectivity, the results for IEEE 802.11p also yield for V2V, thanks to OCB mode. Indeed, this mode makes the V2V and V2I communication equivalent from the wireless medium point of view, since there is no properly defined Access Point, as opposed to standard Wi-Fi, and the communication towards a RSU happens in the same way as towards other vehicles (with the only difference that an RSU is static and other vehicle may be moving too).

4.3.1 Review on field tests for V2X technologies

Even though they usually represent a smaller portion if compared to simulation-based studies, there are a number of research works in literature focusing on field

tests for V2X technologies. The importance of field tests has been indeed highlighted by several works in the literature, including [186]–[188]. Different technologies are usually evaluated with respect to some metrics of choice, such as in [187] in which IEEE 802.11p UDP throughput is tested in a V2I urban scenario, by relying on an open custom implementation, based on PC Engines ALIX.2 boards. ALIX.2 boards represent the previous generation of the PC Engines APU boards we selected for the development of our open DSRC platform[6]. Most of the studies available in the literature focus on a single technology, as opposed to the comparison presented in this Section.

In [189], Klapez *et al.* present field test results and investigate the effects of interference due to hidden terminals, presenting also an exhaustive list of other related works. Our analysis differs from theirs, as we evaluate the maximum range in LOS and NLOS conditions, which was deemed as future work in [189], and we assess the performance of mmWave solutions for V2X communications. Indeed, while past works compare IEEE 802.11p with other V2X technologies, none of them considers IEEE 802.11ad.

In [186], Shi *et al.* present an experimental evaluation of IEEE 802.11p and LTE C-V2X, focusing not only on communication parameters but also on application-oriented metrics and considering different scenarios and driving patterns. They mainly use commercial devices. Some of our findings, described in the next Sections, confirm the results obtained by Shi *et al.*, for instance concerning the latency that always remains low as long as the connection is stable, or concerning the NLOS conditions noticeably reducing the reachable range.

In [188], Chen *et al.* perform a comparative analysis of IEEE 802.11p and 802.11n, considering four scenarios. For what 802.11p is concerned, 20 dBm devices with 5 dBi antenna are used, yielding a similar Equivalent Isotropic Radiated Power (EIRP) as the one tested in our work. However, a 2.4 GHz device is used for IEEE 802.11n, which is an older amendment than 5 GHz IEEE 802.11ac.

Furthermore, the work by Zishan *et al.* [190] studies a heterogeneous V2X communication system, comparing the performance of IEEE 802.11p against that of LTE and general-purpose Wi-Fi (with an IEEE 802.11a/b/g/n wireless card).

The assessment of IEEE 802.11ac in vehicular networks, as a possible promising technology, has been instead scarcely addressed. One of the few works that have studied this aspect is [191], where the use of IEEE 802.11ac in automotive scenarios is investigated by simulation only, with the integration of SUMO traffic patterns into ns-3.

Concerning IEEE 802.11ad, it is worth mentioning an interesting work by Nitsche *et al.* [192]. In their paper, they describe the standard and its design assumptions,

[6] https://pcengines.ch/alix2.htm

besides the study of the transition from omnidirectional to highly directional communication.

Specifically-targeted works focusing on field tests in the automotive domain are instead more rarely found, since most studies are simulation-based. In [42], the authors compare the performance of IEEE 802.11p and the mmWave technology by simulation, when supporting V2V communications. LTE and mmWave for V2I communications are instead evaluated in [193]. Both works found that IEEE 802.11p and LTE outperform mmWave in terms of connection robustness and reliability.

However, only a mmWave technology can address the demand for extreme high throughput by several emerging automotive applications, such as "See Through", as described in Section 2.1.1, and "Birds Eye View" [130], which are pivotal to enable such relevant use cases as adaptive platooning.

MmWave-based V2V communication is also addressed in [194], which introduces a framework to efficiently pair vehicles and optimize both transmission and reception beamwidths by jointly using matching theory and swarm intelligence. In particular, it considers both Channel State Information (CSI) and Queue State Information (QSI) when establishing the link between vehicles; the results, however, are obtained via simulation campaigns only.

The scarcity of studies comparing different technologies for V2X on the field, considering both mmWave and non-mmWave devices, and with open and low-cost solutions led us to the work described in this thesis.

As described in the next Sections, we performed a thorough investigation, varying different parameters for the same technologies. For instance, IEEE 802.11p is evaluated by fixing different physical data rates (like in Section 4.2) and IEEE 802.11ac is evaluated considering different transmission power values.

4.3.2 Experimental setup

All the experiments have been conducted in a rural area near Turin, Italy, where a few long straight roads allowed us to test the devices in both LOS and NLOS conditions, and in absence of interference from other devices.

The measurements focused on V2I scenarios, using two cars equipped with the radio interfaces to be tested and considering a direct communication between the two. Each test has been carried out by fixing the position of one vehicle and letting the other move progressively away from the former, at an average speed of 15 km/h, with the aim of providing repeatable results while limiting the impact of the Doppler effect. Since our focus is on a V2I scenario, the stationary vehicle acts as a Road Side Unit (RSU) and the moving vehicle as an On-Board Unit (OBU).

We assessed the performance as the distance between the two communicating devices varied, focusing on the metrics mentioned earlier. To this end, we equipped the moving vehicle with a GNSS receiver, so as to track its position and thus the

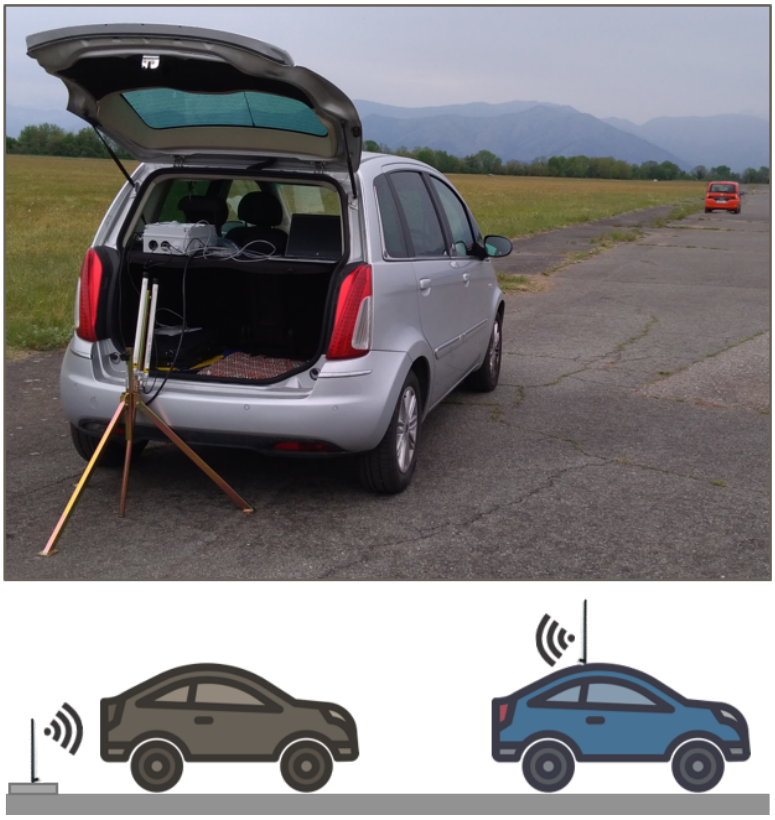


Figure 4.19: Test setup for measurements under NLOS conditions.

distance from the fixed device. The chosen receiver is a Navilock NL-8012U magnetic mount Multi GNSS receiver, based on the u-blox M8 chipset. This chipset provides a configurable update rate, which was set to 5 Hz for our measurements, together with support for multiple satellite constellations (including GPS, Galileo and GLONASS). It can be interfaced in OpenWrt (and OpenWrt-V2X) either through a serial interface, or thanks to the Linux `libgps` library.

Furthermore, in the LOS tests, the antennas have been placed on the roof of both cars. Instead, to create NLOS conditions, the fixed device was placed behind the trunk of the stationary vehicle, as shown in Figure 4.19, thus making such vehicle act as an obstacle between the two communicating devices. This allowed us to test a common NLOS condition in urban scenarios, i.e., a vehicle blocking the LOS between two communicating devices.

The setup for the three access technologies, including the selected antenna configuration, is resumed below:

- **IEEE 802.11p:**
  - *Txpower:* 18 dBm;

  - *Rate adaptation:* off (3, 6, 12 Mbit/s);
  - *RSU antennas*: 2x 6 dBi MobileMark ECOM6-5500 - omnidirectional;
  - *OBU antennas*: 2x 6 dBi MobileMark ECOM6-5500 - omnidirectional;
  - *Channel:* 178 (5.890 GHz @ 10 MHz).

- **IEEE 802.11ac:**
  - *Txpower:* 18, 30 dBm;
  - *Rate adaptation:* on;
  - *RSU antennas*: 3x 12 dBi Interline Horizon Maxi - omnidirectional;
  - *OBU antennas*: 3x 6 dBi MobileMark ECOM6-5500 - omnidirectional;
  - *Channel:* 149 (5.745 GHz @ 20 MHz).

- **IEEE 802.11ad:**
  - *Txpower:* auto;
  - *Rate adaptation:* on;
  - *RSU antennas*: 6x6 embedded antenna array;
  - *OBU antennas*: 6x6 embedded antenna array;
  - *Channel:* 1 (58.320 GHz @ 2160 MHz).

The next Sections better detail the software and hardware we used to carry out the measurements, together with the setup and configuration for the various communication scenarios.

IEEE 802.11p

Concerning IEEE 802.11p, we used the open DSRC platform described in Section 3.2.

As mentioned earlier, during the measurement sessions, the devices could directly communicate thanks to the IEEE 802.11p OCB mode. Thus, the obtained results, even though directly comparable with the other technologies in the V2I scenarios, can be considered valid also when a V2V communication is in place.

IEEE 802.11ac

The platform chosen for IEEE 802.11ac is a custom-built open platform based on the same PC Engines boards as our IEEE 802.11p solution, i.e., PC Engines APU1D.

The two devices run a clean version of OpenWrt 19.07.1 and the wireless cards used to enable the communication are two Compex WLE900V5, allowing the communication via IEEE 802.11ac in the 5 GHz band with a maximum transmit power

of 30 dBm, also thanks to their 3X3 MIMO (Multiple-Input and Multiple-Output) configuration and to an auxiliary 5 V power supply. These wireless card are supported by the *ath10k* driver, which can be included in any recent OpenWrt installation.

In the measurements involving IEEE 802.11ac, the static device is equipped with three outdoor, high gain, omnidirectional antennas mounted on an industrial tripod. Moreover, the static device is configured to act as an Access Point, while the moving device acts as a wireless station. So doing, these settings represent a V2I scenario, with the static node in RSU-like configuration, and the moving node acting as a vehicular OBU connected to the network infrastructure.

As mentioned earlier, although IEEE 802.11ac was not originally designed to enable connectivity in a vehicular environment, it is interesting to explore this opportunity, given the maturity of the protocol and the great availability of devices that implement this technology, including standard smartphones.

Furthermore, it is worth mentioning how, during our tests and as opposed to IEEE 802.11p, the IEEE 802.11ac rate adaptation mechanism has been kept on, leading to a variable Modulation and Coding Scheme, automatically selected based on the channel conditions.

Figure 4.20 depicts one of the APU1D boards we used for the IEEE 802.11ac platform. Notice the Compex WLE900VX mPCIe module, and the SD card as main storage. The latter was necessary as the Compex module has a very large form factor which impedes the installation of the mSATA SSD on the nearby slot.

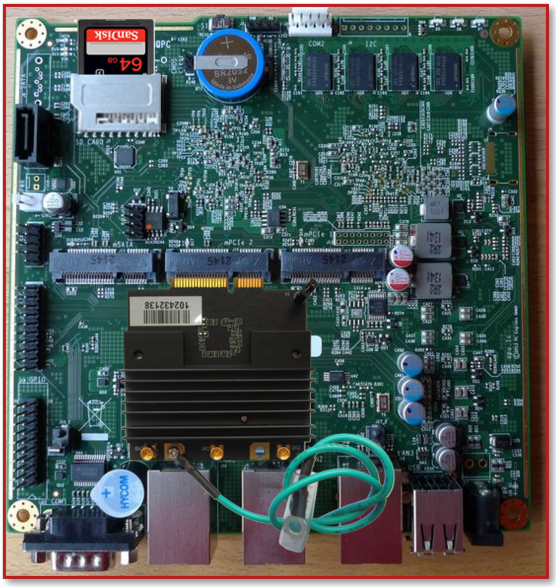


Figure 4.20: Top view of a PC Engines APU1D board with the main components enabling communication over 5 GHz IEEE 802.11ac (i.e., the high-power Compex WLE900VX mPCIe module and a high-speed SD card as main storage).

Finally, it should be mentioned that we performed all tests by keeping the default Number of Spacial Streams (NSS) (i.e., 3), Aggregated MAC Protocol Data Unit (A-MPDU) size, and Aggregated MAC Service Data Unit (A-MSDU) size [195] (i.e., no specific values or limits have been enforced, leaving the maximum values, respectively, to 64 and 3).

IEEE 802.11ad

The mmWave scenario is composed of two IEEE 802.11ad-compliant MikroTik wAP 60G routers [146]. These devices are equipped with a Qualcomm Atheros QCA6335 60 GHz chipset, with an internal planar phased antenna array of 6x6 elements able to cover an angular range of 60 degrees. Five different non-overlapping channels with a 2.16 GHz bandwidth are supported, from 58.32 GHz to 66.00 GHz. Furthermore, the maximum link distance declared by the manufacturer is up to 200 m.

The installed OS is a proprietary Linux-based distribution called RouterOS (version 6.48 has been used in this experimental evaluation). This operating system provides a reduced set of available tools with respect to OpenWrt; however, it proved to be very effective in managing the devices and obtaining the desired metrics, thanks to its full integration with the MikroTik firmware. This is the reason why we have chosen to keep RouterOS, even through efforts have been made to port OpenWrt to the MikroTik wAP 60G devices [196], [197].

The high frequencies at which mmWave technology operates are subject to severe path loss and harsh propagation over the air. Despite the high potential mobility of nodes, mmWave has been studied for vehicular applications since it is an IEEE 802.11-based technology achieving data rates of the order of multiple Gbit/s and a very small RTT. As noted via simulation in several works and highlighted by our measurements, mmWave could be coupled with more reliable technologies (e.g., IEEE 802.11p) to achieve at the same time robustness and reliability, as well as low latency and high throughput.

Similarly to the IEEE 802.11ac case, the static device acts as an AP and the moving one as a station. To assess the performance of IEEE 802.11ad in controlled conditions, the moving vehicle always follows a straight line path, keeping the angle between the two devices close to 0 degrees.

Finally, it is worth mentioning that the transmission power is always automatically set by the device, as opposed to the configuration of the other non-mmWave devices.

4.3.3 Open source measurement tools

As part of our testbed setup, we relied upon different open source software tools to successfully collect the metrics of interest, namely, RSSI, RTT and UDP

throughput, all gathered with respect to the distance between the two communicating devices.

None of the available tools, however, natively support reporting the measured values, at each instant of time, together with the localization and distance data gathered from a GNSS receiver. To overcome this issue, we developed three utility programs, one for each metric, able to output synchronized network KPIs and distance information. These utility programs interface with both the main measurement tools and the GNSS receiver, and rely on the Haversine formula [126] to compute a good estimate of the distance between the devices, considering the mean Earth radius. Furthermore, they can gather the localisation data thanks to the `libgps` Linux library. Their output is a set of CSV files reporting, for each distance value, the RSSI, RTT or UDP throughput measured in that position (together with other useful information).

iw and monitor for RSSI measurements

As far as the RSSI measurements are concerned, each device, acting either as OBU or RSU, is connected to a laptop via Gigabit Ethernet. Both laptops are equipped with high-quality Intel I219-V NICs. As the devices can update the value of the measured RSSI only when data is actively received, a ping session is started from the fixed device to the device on the moving vehicle, with an inter-packet frequency of 100 ms. Then, the laptop on the moving vehicle is used to gather the RSSI metrics every 200 ms (i.e., at the same frequency as our GNSS receiver updates), by relying upon:

- The Linux `iw` tool for the APU boards (concerning IEEE 802.11p and IEEE 802.11ac); notice that this is the same tool we used to gather the received signal power level for the static laboratory tests reported in Section 4.2.4.
- The `/monitor` command for the wAP 60G devices, providing RSSI values in a similar way as `iw`.

Both commands are launched from the laptop to which the GNSS receiver is connected, via the `ssh` protocol.

iPerf for throughput measurements

iPerf 2, as a reliable and state-of-the-art software [198], is the open source tool of choice for the throughput measurements.

The physical devices (either the APU boards or the MikroTik routers) are set in bridge mode, in order to act as bridges between the laptops running the measurement tool. The fixed laptop, hosting an iPerf client, is set to push as much UDP traffic as possible toward the moving vehicle (trying to reach a 1 Gbit/s traffic load with packets of 1470 bytes), while the moving vehicle's laptop is running an

iPerf server that measures the maximum achievable throughput. To produce the final logs, we used the aforementioned utility programs, interfacing to the output of both iPerf and the GNSS receiver at the same time.

Finally, it should be mentioned that, when configured with the special bridge mode [199], our open DSRC platform does not select any specific AC, as of now. This results in all traffic using the default AC_BE traffic class, and allows us to produce comparable results with respect to the ones gathered with the other technologies, which do not support traffic prioritization through Access Categories as in IEEE 802.11p.

LaTe for latency and service availability measurements

The last set of tests involved the measurement of the RTT between the two devices. As software tool, we rely on LaTe v0.1.6-beta, measuring latency with LaMP over UDP and IPv4, as described in Section 3.3.3. As in the previous case, the physical devices are set in bridge mode and a GNSS data synchronization utility is used. The latter, in particular, exploits the capability of LaTe of sending the measured latency, for each packet, to external applications for further processing. This external application is represented by the synchronization utility, which also receives data from the GNSS device. A sample Python version of this utility is available as part of the LaTe GitHub repository[7].

Three main reasons drove our choice towards LaTe, instead of simpler tools like `ping`:

- LaTe provides more advanced features for reliably measuring latency in mobility scenarios, including the possibility of automatically logging data in CSV files for post-processing the raw data.
- As mentioned in the previous Chapter, LaTe relies upon packets transmitted over UDP, instead of ICMP, which is typically the transport protocol of choice when vehicular applications are transmitting data over the IPv4 stack.
- LaTe, by leveraging LaMP, is able to provide a clear estimate on the latency experienced at the application layer (i.e., the actual latency that would be experienced by V2I applications).

A LaTe client is launched on the laptop inside the moving vehicle, while a LaTe server is launched on the stationary device, i.e., the one acting as RSU. The packet periodicity is set to 50 ms and the UDP packet size to 24 bytes (corresponding to LaMP packets with no payload, i.e., the smallest packets that can be sent with

[7] https://github.com/francescoraves483/LaMP_LaTe/tree/development/examples/w%20option%20sockets%20example

LaTe). Furthermore, all measurements have been performed by selecting the "Uset-to-user" latency type.

Finally, LaTe can reliably measure RTT, and react to a network disconnection longer than 4 s (i.e., when no packets are received at either side for more than 4 s) by terminating the current test and automatically relaunching the server. This allowed us to evaluate the maximum achievable communication distance for the tested technologies.

4.3.4 Field test results

To assess the performance of the different 802.11-based solutions, we have performed several field tests, in both LOS and NLOS conditions. Although we collected and analyzed the data for all the mandatory data rates of IEEE 802.11p, for the sake of clarity, only the one yielding the best results for each metric are shown here. The same applies to the levels of tested *txpower* (i.e., the transmission power level set in OpenWrt) for IEEE 802.11ac: only the 18 dBm case is reported, as it can be compared to the maximum *txpower* value that can be set for IEEE 802.11p. Furthermore, as reported earlier, we selected an IEEE 802.11ac channel bandwidth of 20 MHz, to test the most similar value to the 10 MHz bandwidth of IEEE 802.11p.

With the aim of extracting a meaningful trend and taking into account the error of the GNSS device (in the order of maximum 3 m in open air, with an accuracy of 2.5 m Circular Error Probable, according to the data sheet of the Navilock device), the collected data points have been grouped into bins of 5 meters each, over which the metrics of interest have been averaged. Each bin is then represented by its central point, e.g., if a bin goes from 0 to 5 meters, a new point at 2.5 m is inserted in the plot. The size of the bins has been, however, increased from 5 to 15 meters when showing the RSSI results, in order to improve the readability of the figures and better show the overall evolution with respect to the distance. This holds for all the plots but the one in Figure 4.21, which reports exact distance values.

The first field test campaign was aimed at determining the maximum achievable distance by the three technologies under study, before a disconnection is observed between the communicating devices. As software tool, we relied on LaTe, by which we recall that a disconnection occurs when no packets are received for more than 4 s. The results are shown in Figure 4.21.

The results show that IEEE 802.11ac reaches the greatest distance, while providing an acceptable stability of the connection. This can also be explained thanks to *(i)* the usage of the lowest frequency among all the technology considered, and *(ii)* the high-gain antennas and maximum configurable transmission power of the IEEE 802.11ac platform, enabling higher values of EIRP than the ones of the other communication systems.

On the other hand, IEEE 802.11ad, which operates at 60 GHz, can reach substantially lower distances, although, as detailed later, it provides the highest throughput

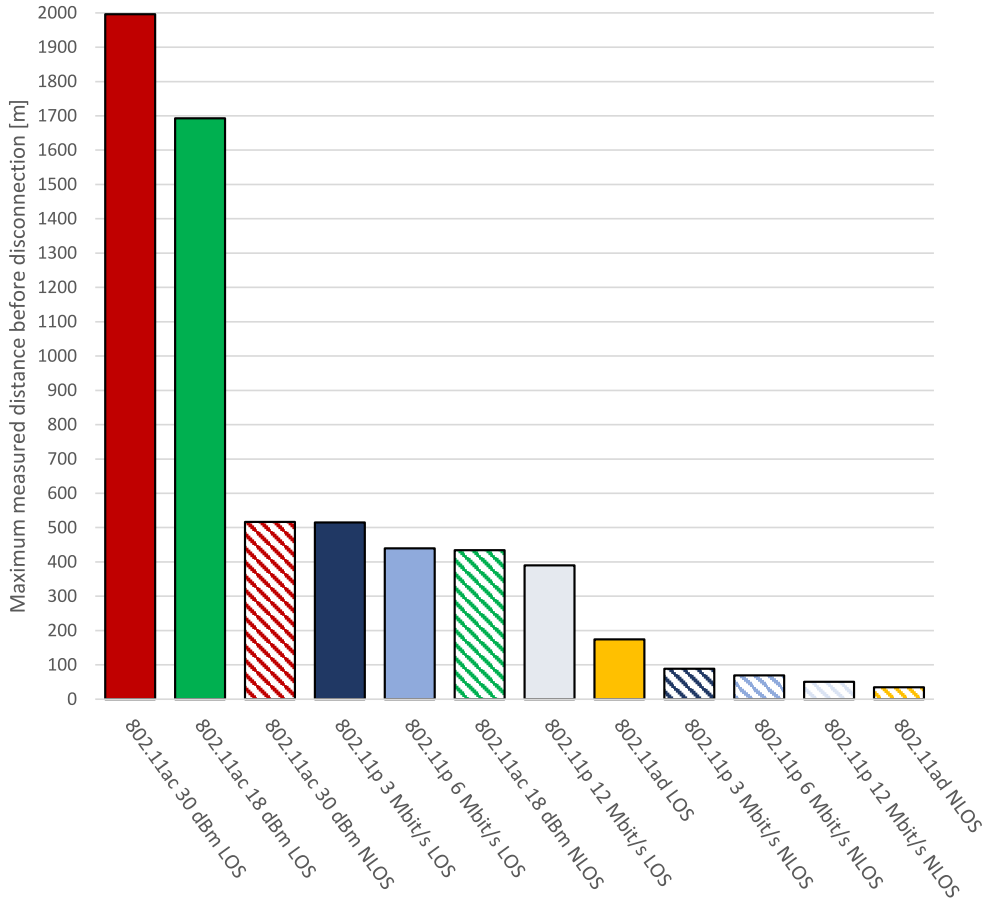


Figure 4.21: Comparison among IEEE 802.11p, 802.11ac, and 802.11ad: maximum achievable distance before a disconnection between two communicating devices is detected, under LOS and NLOS conditions.

and the lowest RTT. Nevertheless, it is worth mentioning how even this technology can achieve few tens of meters in NLOS conditions (we measured an achievable distance of around 34 m), which could be enough to support high throughput data exchange in dense inner-city scenarios. Another important observation is about the impact of NLOS conditions: for the considered technologies, NLOS implies a reduction in the radio coverage of more than 50% with respect to LOS. This confirms the importance of accounting for NLOS conditions when designing V2X communication systems.

Next, Figure 4.22 depicts the evolution of the RSSI as a function of the distance between two communicating devices, in LOS (top plot) and NLOS (bottom plot) conditions.

Comparing the two plots (note the different scale of the x-axis in the two plots), one can see that, for IEEE 802.11p and 802.11ac, the highest values of RSSI in

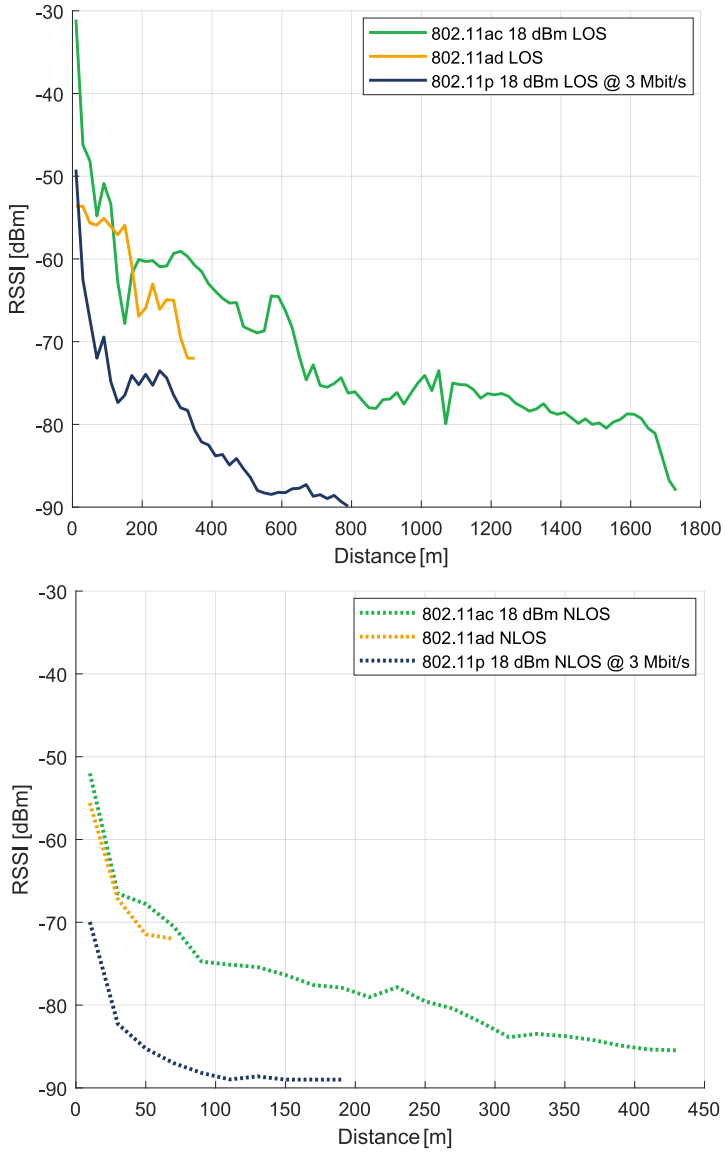


Figure 4.22: RSSI measurements with IEEE 802.11p (18 dBm and 3 Mb/s), 802.11ac (18 dBm), and 802.11ad. LOS (top) and NLOS (bottom) conditions.

NLOS are 20-25 dBm lower than the respective values in LOS conditions. The behavior of IEEE 802.11ad is instead different, due to beamforming and the significantly different frequencies at which the technologies operate. In this case, the RSSI is just slightly higher in LOS condition for sufficiently short distances, while it drops very rapidly in NLOS conditions, relatively to the LOS scenario. In general, looking also at the other measured KPIs, it is clear that IEEE 802.11ad cannot support a stable communication when the RSSI is below -71 dBm, which is consistent with the receiver sensitivity values reported in the standard [46]. On the contrary,

with IEEE 802.11ac, one starts experiencing communication disruptions only when the RSSI is lower than -82 dBm, while with IEEE 802.11p a quite stable connection can be maintained, on average, till -87 dBm is reached, which is consistent with the static measurements reported in Section 4.2.4. Thus, IEEE 802.11p (with its mandatory modulation schemes) appears to be the best technology in terms of connection stability, which is consistent with the fact that it is indeed a technology specifically designed for vehicular communications.

The UDP throughput results are depicted in Figure 4.23.

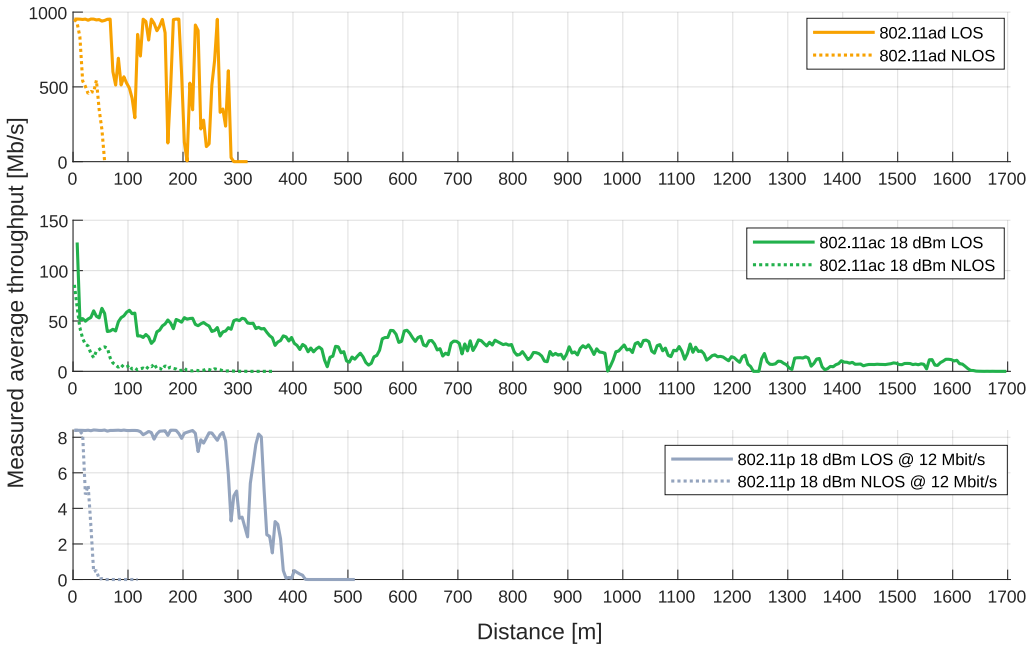


Figure 4.23: UDP Throughput as a function of the distance (LOS and NLOS conditions).

Looking at all the technologies, when the distance is very short, the same values of throughput can be achieved in LOS and NLOS conditions, even though the drop is much faster in NLOS than in the case of LOS when the distance increases. As mentioned earlier, IEEE 802.11ad can provide very high values of throughput, exceeding 1 Gbit/s (here capped at around 953 Mb/s due to the Gigabit Ethernet connection to the laptops), even though the maximum reachable distance is quite low, especially in NLOS conditions. IEEE 802.11ac can instead provide up to 127 Mb/s in LOS conditions and below few meters, which is then quite smoothly reduced as the distance increases, thanks to the rate adaptation mechanism. Even though a comparison with precise theoretical values is not possible as few details are, unfortunately, missing (e.g., the MCS selected for each tested distance value), the measured values appear to be quite good, considering the mobility of the nodes, the real outdoor environment, with a minimum tested distance around 6 meters,

and the configuration with 20 MHz channels. Furthermore, it was possible to verify, when iPerf was transmitting UDP test packets, that A-MSDU was used by aggregating two MSDUs at a time, and that the actual A-MPDU size was instead varying during the test duration.

On the contrary, the absence of an active rate adaptation in the IEEE 802.11p system causes the throughput to oscillate much more when the connection becomes less stable. It is important to take into account that IEEE 802.11p, even though providing a lower throughput (around 8.43 Mbit/s when the physical data rate is set to 12 Mb/s and the traffic class is left to AC_BE), works on a dedicated spectrum and 10 MHz-wide channels, providing a higher degree of resiliency with respect to interference from non-V2X communications and high speed. As these tests focus on a baseline characterization involving two vehicles only, these advantages are not clearly visible here, but they have nevertheless to be taken into account. It is also worth mentioning that no association procedure is foreseen in IEEE 802.11p, thus enabling direct communication between the devices, as opposed to IEEE 802.11ad and IEEE 802.11ac.

Figure 4.24 presents the experimental data on the RTT, gathered thanks to LaTe.

It is evident from the plot that the latency remains quite stable for all the technologies over all the measured distances, until the RSSI can no longer guarantee a stable connection. When this happens, the latency rapidly increases until a disconnection occurs. The exception is the IEEE 802.11ac NLOS case. This technology can reach a longer range than IEEE 802.11p, as highlighted before, but it provides a less stable RTT when non-ideal conditions are in place (i.e., in NLOS conditions). By comparing the different values when a stable connection is established, IEEE 802.11ad provides the lowest RTT values (around 1.5 ms), then IEEE 802.11ac provides on average around 3 ms in LOS and NLOS condition, which is very similar to what can be achieved with IEEE 802.11p (i.e., around 3.4 ms). These values can also be explained by looking at the maximum reachable throughput: the higher the throughput, the lower the overall transmission time. All these values are, in any case, low enough to fully support safety applications that exploit V2X communications, when taking as reference the latency-critical use cases reported in [200].

Finally, Figure 4.25 presents a comparison between the IEEE 802.11ac RSSI measured when transmitting at 18 dBm and 30 dBm, both in LOS and NLOS conditions.

As can be seen, NLOS causes a drop in the measured RSSI in the order of 20 to 26 dBm, depending on the selected value of *txpower*[8]. Increasing the transmission power can lead to visible advantages until relatively high distances are reached

[8]It is important to remember that this value is not always equivalent to the overall EIRP, which also includes the contribution of the specific antennas.

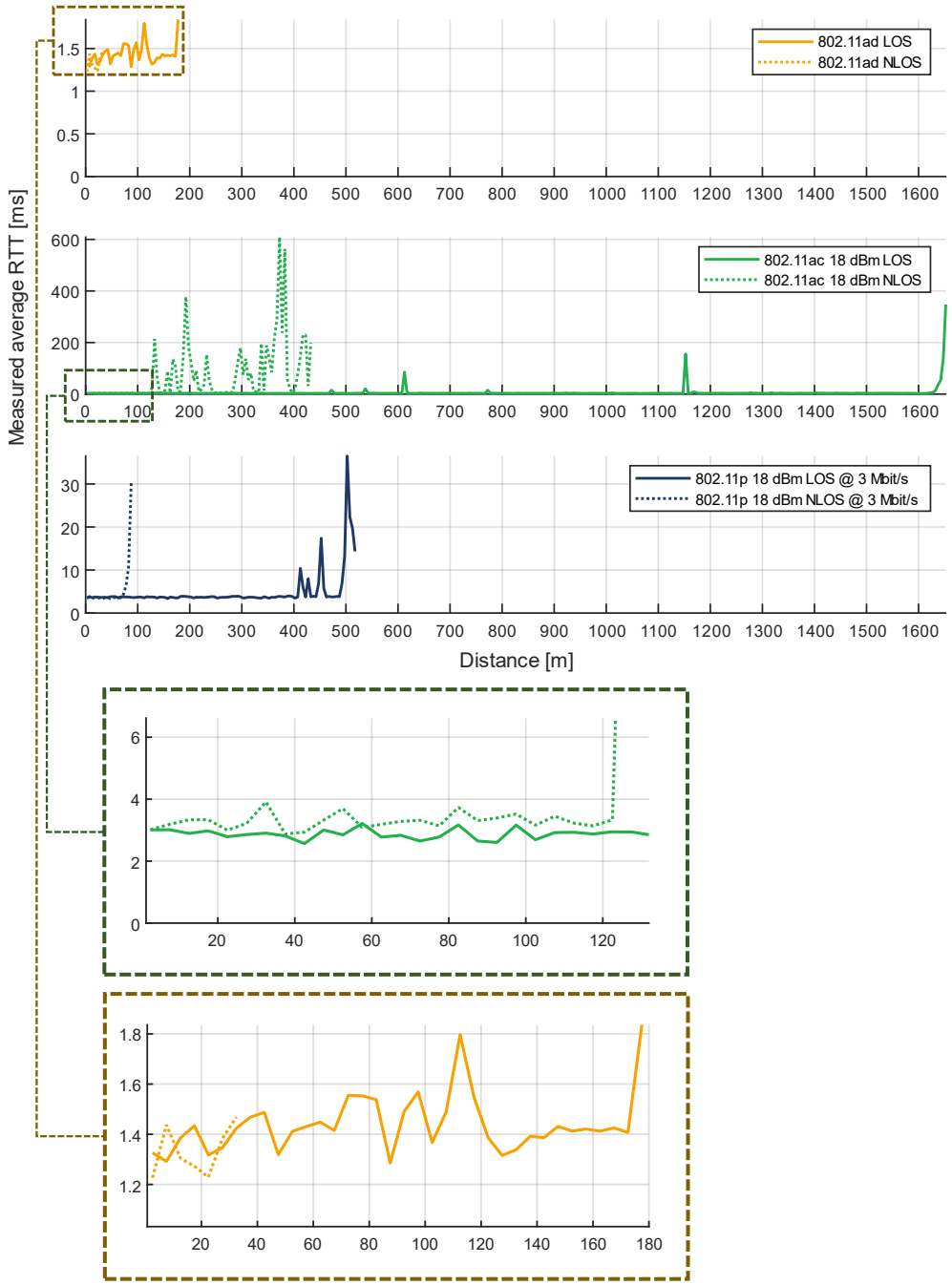


Figure 4.24: Round Trip Time as a function of the distance (LOS and NLOS conditions). The zoomed portions of the plots are intended to facilitate the interpretation of the results.

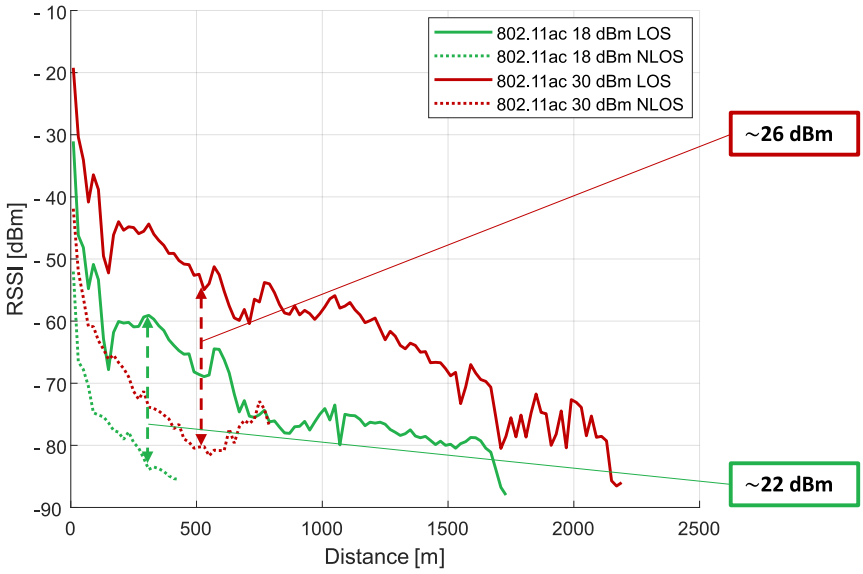


Figure 4.25: RSSI measurements with IEEE 802.11ac, for two values of *txpower* and in LOS and NLOS conditions.

(1.5 km). Then, both the 30 dBm and the 18 dBm case lead to an unstable connection (causing, for instance, the throughput to drop to 0) between 1.6 and 1.7 km, with a gain of less than 100 m when transmitting at 30 dBm (also due to RSSI values reaching around -82 dBm). Thus, for very large distances, the reported results show that there is no evident advantage in increasing the IEEE 802.11ac transmission power to its maximum, as it brings little benefit in terms of reachable range and RSSI.

Final discussion and future directions

Comparing the different technologies, IEEE 802.11ac proved to be the one maximizing the radio range, both in LOS and NLOS, providing acceptable throughput up to 1.5 km-distance. However, in non-ideal NLOS conditions the RTT becomes less stable after around 120 m, as opposed to IEEE 802.11p, which provides an almost constant latency until the RSSI drops below -87 dBm (considering a physical data rate of 3 Mb/s), which is the lowest value among all the tested technologies.

On the other hand, IEEE 802.11ad can provide the highest throughput, even exceeding 1 Gbit/s, and the lowest latency, also thanks to a very small transmission time, at the price of a noticeably reduced radio range and of the need for an association procedure, which is absent in IEEE 802.11p.

Thus, IEEE 802.11ad appears to be a promising technology for the short-range transmission of large quantities of data, like in the case of upload and download

of sensor data at intersections. Even though the radio range is relatively short, it is worth highlighting that it is possible to reach few tens of meters even when there are obstacles between the communicating devices. This could be sufficient to enable automotive use cases that require very high throughput and low latency in dense urban scenarios, despite the usage of the 60 GHz frequency spectrum which is commonly thought to suffer from complete communication disruption under NLOS conditions. The obtained results also suggest that IEEE 802.11ad could be used in combination with longer-range access technologies, to provide very high throughput in proximity of mmWave dedicated RSUs, while providing at the same time an extended range thanks to the coverage capability of the other technologies.

Given the importance of field tests, we are currently planning to investigate the effect of medium to high longitudinal speed on the studied access technologies, as well as the impact of the antenna height and configuration. Concerning instead IEEE 802.11ad, further field tests are planned to evaluate the effect of the angle between communicating devices as a function of their relative distance, as the antenna array is highly directional and the angular range is limited to 60 degrees. We are also planning to investigate in the field the problem of beam training, which may represent an impacting factor when vehicles are moving at a sufficiently high speed. Finally, we are also going to address V2V scenarios and compare on the field IEEE 802.11p and LTE-V2X Mode 4, thanks to our open DSRC platform, combined with dedicated embedded C-V2X boards running a Linux-based OS.

4.4 Publications

This Section reports our publications, related to the topics presented in this Chapter.

It should be mentioned that a journal publication on ms-van3t has been already submitted to Elsevier Computer Communications, and, at the time of writing, it is currently under review.

4.4.1 Conferences

ms-van3t

- 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”, ACM DIVANet 2020, Alicante, Spain, November 2020, pp. 17-24 [32]

Baseline characterization and performance evaluation of IEEE 802.11p

- F. Raviglione, M. Malinverno and C. Casetti, "Characterization and Performance Evaluation of IEEE 802.11p NICs", 1st ACM Workshop on Technologies, mOdels, and Protocols for Cooperative Connected Cars (TOP-Cars), Catania, Italy, July 2019, pp.13-18 [33]

Experimental assessment of IEEE 802.11-based V2I technologies

- F. Raviglione, M. Malinverno, S. Feraco, G. Avino, C. Casetti, C. F. Chiasserini, N. Amati and J. Widmer, "Experimental assessment of IEEE 802.11-based V2I communications", ACM PE-WASUN 2021, Alicante, Spain, November 2021, pp. 33-40 [12]

Chapter 5

Innovative services and protocols for connected and autonomous vehicles

When deploying and testing a complete open V2X environment, three main wide areas can be defined: *(i)* development of open source platforms able to provide (and test) vehicular connectivity and enable V2X applications, based on standardized (or promising) access technologies and networking stacks, *(ii)* performance assessment of different access technologies and protocols, both in large scale simulations and in real-world outdoor scenarios, *(iii)* development of actual V2X services and novel open protocols for connected and autonomous vehicles.

While Chapter 3 tackles the first point, and Chapter 4 is related to the second wide area, the aim of this Chapter is to present our work on the third field.

This Chapter is indeed devoted to the presentation of a novel, ETSI-compatible, protocol for raw GNSS data exchange between vehicles, enabling the so-called *Cooperative Positioning (CP)* approaches thanks to V2X connectivity and cooperative exchange of data. These novel approaches, as mentioned in the introductory Chapter, enable several GNSS-related applications, including: *(i)* enhancing the localization accuracy, especially in harsh environments, *(ii)* providing positioning integrity and *(iii)* enabling distributed time synchronization through cooperative exchange of raw positioning information. All these approaches require the exchange of raw GNSS data between road entities, and, especially, between vehicles thanks to V2V connectivity. However, there is currently no open protocol, designed to be encapsulated into the ETSI BTP and GeoNetworking layers, for the exchange of this kind of data between vehicles. We thus design, implement and evaluate a novel protocol called *Cooperative Enhancement Message (CEM) protocol*.

After the presentation of the CEM protocol, this Chapter presents an innovative V2X service developed in the context of the 5G-CARMEN project, tackling the

need for a centralized Local Dynamic Map (LDM) [26], able to collect messages from vehicles thanks to 5G and storing the most up-to-date and historical data of all vehicles (and other non-connected objects detected thanks to sensors) travelling in a given area. This service acts as "middleware" and can very efficiently provide a filtered and processed version of this data to other highly automated maneuver management MEC services, when it detects certain "triggering" conditions on the road (e.g., a vehicle trying to perform a centralized automated lane merge). These maneuver management MEC services are indeed safety-critical and with very tight latency requirements, and the presence of a centralized LDM can offload them from the burden of receiving and processing a huge amount of raw data from vehicles.

5.1 The CEM protocol: from Collaborative Awareness to ETSI-compliant Information Enhancement

Thanks to the remarkable localization capabilities of modern navigation systems, several new successful paradigms in urban mobility have emerged. A growing number of integrated systems can ensure relative positioning and navigation capabilities with respect to nearby objects by gathering heterogeneous information from different on-board sensors. Despite the growing availability of such sensors, GNSS receivers are still the solution of choice for the estimation of absolute timing and position, thus supporting a plethora of services based on localization at the application layer. Position, Velocity and Time (PVT) estimates have become so relevant in applications that they are often overrated in terms of accuracy and assumed for granted on a continuous base. As a matter of fact, most V2X literature typically assumes positioning as very accurate, without considering the estimation uncertainties, which is often not the case especially in urban scenarios. Urban environment can indeed present harsh conditions due to multipath, fading and occlusions. These effects impair the quality of the positioning solution provided by GNSS devices and hybrid, integrated positioning and navigation units.

On the other hand, many GNSS-related algorithms proposed by recent and past research give the transmission of data for granted, without considering the issues related to the communication network.

Furthermore, in line with the growing interest towards Cooperative Intelligent Transport Systems [201], vehicular communications are expected to support novel paradigms in positioning and navigation technologies that are gathered in the literature under the name of Cooperative Positioning (CP) [202]. To meet this trend, the data transmitted via CAMs (i.e., position, speed, heading, acceleration, and so on) may not be sufficient, and additional GNSS measurements need to be exchanged among cooperating and connected network nodes. Indeed, the exchange of the pure PVT data obtained by means of GNSS receivers and encoded in CAMs

cannot supply the novel approaches addressed by CP.

With the aim of filling this gap and tackling the issues mentioned above, we:

- Design a novel vehicular message type, named Cooperative Enhancement Message (CEM), that extends the capability of the CAMs and CPMs specified by ETSI [203]–[205], so as to enable more advanced cooperative applications for connected vehicles; this message essentially stores in an efficient way raw GNSS and sensor data from vehicles;
- Define a novel dedicated transmission protocol, named CEM protocol, that exploits CEMs to let connected vehicles share GNSS raw data; it is worth mentioning that such data is essential to the evolution of both mobile devices and intelligent transportation systems;
- Analyze the impact of the additional CEM traffic on the network performance through a newly developed open-source simulation environment, i.e., a dedicated version of ms-van3t named *ms-van3t-CAM2CEM* [25].

To tackle the challenge of simulating and emulating vehicular scenarios with real GNSS data, affected by estimation errors, we have also integrated in ms-van3t a set of real-world traces that were recorded through a highly accurate GNSS Inertial Navigation System (INS) positioning and navigation unit. These traces include both real-world GNSS PVT and acceleration/heading data to be used in conjunction with the ms-van3t `gps-tc` module, and a set of sample traces with raw GNSS data. This enables testing the feasibility and performance of CP solutions, which is of utmost importance to gain an understanding of the behavior of the network and advanced ITS applications. Furthermore, this data complements a simulation framework (i.e., ms-van3t-CAM2CEM) that allows for the evaluation of both CP approaches using the proposed CEM protocol, and of V2X applications at a larger extent. The present work is conceived as a toolbox for an actual implementation of CP algorithms. However, case studies that assess performance gaps between conventional and collaborative approaches are already present in the literature and are thus out of the scope of this thesis.

5.1.1 An introduction to GNSS raw data and observables

Since the early steps of GNSS integration in vehicular navigation subsystems, the main output of a GNSS receiver has been considered to be the positioning solution (i.e., post-processed latitude, longitude, velocity and time data). However, the PVT inference is performed thanks to the preceding estimation of specific measurements, namely, *raw GNSS measurements* or *observables*, along with their uncertainties. Such quantities typically consist of *pseudorange* measurements, which are estimates of the distance between the receiving antenna and the visible satellites. Receivers are also able to extract other quantities from the received signals, such as the

variation rate of such pseudoranges (which is related to the Doppler shift), and the Carrier-to-Noise ratio of the received navigation signal (C/N_0). The current trend in modern GNSS receivers is to provide the uncorrected pseudoranges as outputs together with the set of observables.

Broad use of raw GNSS measurements in high-end receivers has also recently pushed manufacturers to their disclosure in mass-market devices [206]–[209]. Furthermore, the incremental use of such measurements paved the way for several applications in both mobile devices and transportation systems [210].

Modern multi-constellation and multi-frequency receivers can exploit up to hundreds of channels to track as many GNSS signals, transmitted by all the available satellite systems over different bandwidths. The availability of a large amount of raw measurements, obtained from multiple constellations and over multiple bandwidths, paves the way to new positioning solutions, for instance in terms of tight integration with other sensor measurements in vehicles. This GNSS raw data also enables several novel Cooperative Positioning solutions, as described in the next Section.

5.1.2 An overview on Cooperative Positioning solutions

A remarkable number of cooperative applications, going under the name of *Cooperative Positioning*, aim at improving the accuracy, integrity, and reliability of positioning information by exploiting raw GNSS data concurrently available at different locations [211]. At the same time, they also enable an augmentation framework for those network paradigms and services that take advantage of positioning data.

With the intent of reaching these goals, the ability of vehicles to cooperate with each other and exchange data through V2X was indeed recognized as a promising trend in [212], and later in [213].

In line with the literature on CP, any connected receiver (e.g., vehicle) that is capable of independently estimating its own absolute position will be referred to in this work as either *agent* or *node*.

Among the many valuable examples of CP, recent research works have identified five main cooperative applications enabled by connected vehicles:

1. **GNSS-based ranging.** These applications combine GNSS observables from two connected nodes to estimate their inter-node distance, even in harsh NLOS conditions. As an example, differential techniques, such as Differential GNSS (DGNSS), are able to provide relative ranging with a low computational effort and minimum delay [214].

2. **Multi-agent cooperative positioning and navigation.** Few contributions proposed tight integration schemes to merge asynchronous, non-independent,

non-stationary inter-agent distances [215]. Bayesian estimators such as Extended or Unscented Kalman Filter and Sequential Monte Carlo methods are exploited to fuse ranging measurements and GNSS legacy observables to improve estimation accuracy in harsh environments [216], [217]. Likewise, AI/ML solutions have been recently explored to integrate such quantities, such as in [218]. Recent approaches that use sensitive information related to geographical trajectories and geospatial data for improving performance [219], are prone to benefit from a dedicated protocol, as well.

3. **Cooperative integrity.** The integrity of the positioning data aims at guarantying the correctness of information supplied by the on-board navigation systems. To complement or extend the monitoring (for integrity) of standalone, local data, approaches like Receiver Autonomous Integrity Monitoring (RAIM), and more recent variants (e.g., ARAIM, RRAIM, ERAIM), have been proposed, together with new collaborative solutions. Among these, Cooperative Enhanced Receiver Integrity Monitoring (CERIM) has been proposed to exploit GNSS raw data shared among networked receivers, in order to perform Fault Detection and Exclusion (FED) [220].

4. **Time synchronization.** In this CP application, GNSS receivers can provide *distributed* timing reference, by leveraging an accurate estimation of their onboard clock bias. The availability of in-orbit accurate clocks comes as a distributed timing source that can be exploited by many applications in ITS. Among these, road safety, communication channel scheduling in IEEE WAVE, network interoperability and coordination, time-to-collision monitoring are posing remarkable synchronization constraints that may be not reached through conventional approaches, such as the usage of NTP or PTP. GNSS-based timing is thus gaining a large interest, given also the high accuracy and the lower complexity of its timing estimation. Indeed, modern receivers can reach up to $30\,ns$ accuracy time synchronization between two receivers, as demonstrated in urban environments [221].

5. **Authentication.** The current usage of message authentication in vehicular networks [222], [223] can be further enhanced through the introduction in modern GNSS receivers of the Galileo Navigation Message Authentication (NMA) and GPS CHIp-MEssage Robust Authentication (CHIMERA). Indeed, GNSS message and signals authentications enable discriminating legitimate transmissions from illegitimate ones, thus introducing an independent and time-related source of authentication for network messages and communications at a large extent.

Alongside these applications, the possibility to exchange lower-level information about localization sensors also enables modern approaches such as soft information paradigms [224].

All the aforementioned CP applications require the peer-to-peer exchange of GNSS raw data through general-purpose (e.g., 5G) or dedicated connectivity, such as DSRC or LTE-V2X/NR-V2X. An efficient and open protocol for the exchange of such data in a vehicular network is thus of pivotal importance for the practical deployment of CP solutions.

Finally, it is worth mentioning that, given the multiplicity of applications addressed by the CP algorithms available in the literature, an application-layer performance analysis falls outside the scope of the current study, which focuses instead on the CEM protocol and on its characteristics.

5.1.3 Existing protocols for navigation data exchange

The concept of transmitting or broadcasting GNSS raw data between receivers and reference stations is not new. Currently, there are three main formats that support the transmission of position-related data among connected nodes, each with different kinds of limitations that do not enable the full applicability of the aforementioned CP paradigms.

In particular, it is possible to mention the following protocols:

- **National Marine Electronics Association (NMEA) 0183:** it is an ASCII-based serial communication protocol used in GNSS receivers. It supports the so-called *GGA* strings that provide Time, Position and fix related data for a GPS receiver. The NMEA standard is proprietary and does not support raw GNSS data.
- **Radio Technical Commission for Maritime Services (RTCM) 10403:** it is a *proprietary* standard series that describes messages and techniques for supporting GPS and GLONASS operation with one reference station, or a network of reference stations. The Network Transport of RTCM via Internet Protocol (NTRIP) can be used to transmit RTCM messages for differential corrections in RTK schemes, to noticeably improve localization accuracy. RTCM natively supports real-time-oriented exchange of raw GNSS measurements.
- **Receiver Independent Exchange Format (RINEX)**: it is a data interchange file format for raw GNSS data popular in geodesy. RINEX files can include observation data (i.e., raw measurements), navigation message data and atmospheric condition models. In spite of the usage of some compression schemes (i.e., the so-called *Hatanaka* file compression), RINEX files are not suitable for near real-time applications, and they are used mostly for post-processing and offline investigation.

Considering the various limitations of these formats and protocols, an open protocol for the exchange of raw navigation data between vehicles is still missing, as well as a simulation and emulation framework for the experimental assessment

of the impact on network performance of the transmission of GNSS data needed to enable cooperative positioning algorithms. The main features and advantages of the proposed CEM protocol are compared to the ones of the other available solutions in Figure 5.1.

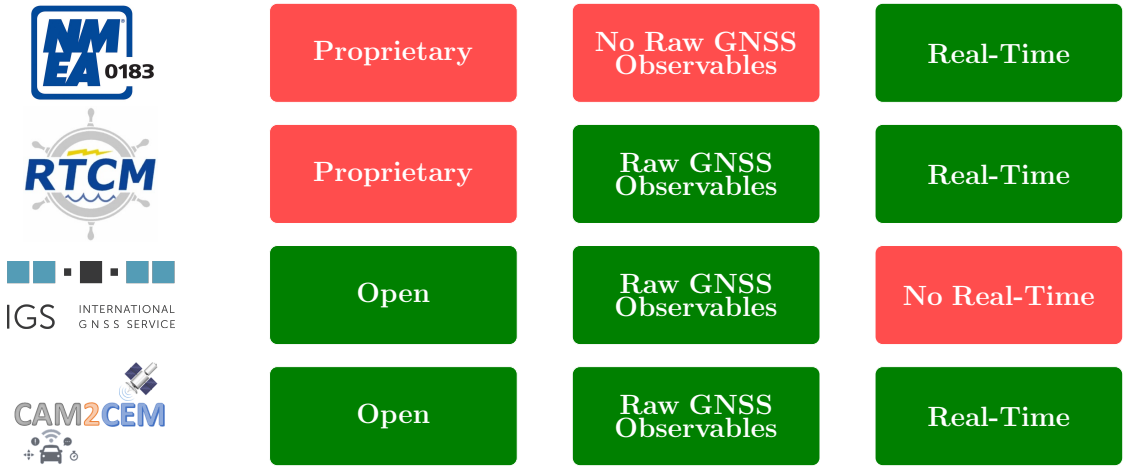


Figure 5.1: Comparison of the main features of existing protocols for the exchange of GNSS data. The last line is related to our CEM protocol.

It is also worth mentioning that ETSI has specified the RTCMEM messages, as described in Section 2.2.4. Nonetheless, the proposed CEM have some significant differences from such messages, in particular *(i)* the main purpose of RTCMEMs is to enable a V2I differential positioning scheme, rather than to directly enable peer-to-peer CP approaches, *(ii)* they encapsulate RTCM data that, as just mentioned, is encoded using a proprietary and closed-source protocol, *(iii)* no optimization approach is employed to reduce usage of network resources (i.e., there is no exploitation of differential data, differently to CEMs, as detailed in Section 5.1.4). Furthermore, CPMs [69] can provide an enhanced awareness of the surroundings, thanks to sharing of objects detected by the on-board sensors, but they still do not allow any improvement to absolute and relative localization, based on exchanged GNSS information.

Finally, it should be noted how our CEM protocol has been designed to be easily implemented and tested on smart city and connected highway architectures, such as the one proposed by C-Roads, after equipping the vehicles (and possibly also the RSUs) with proper GNSS receivers. The plug-and-play solution described in Section 3.5 could be leveraged for instance as a CEM-enabled OBU, thanks to the ArduSimple RTK receivers, supporting the access to raw observables.

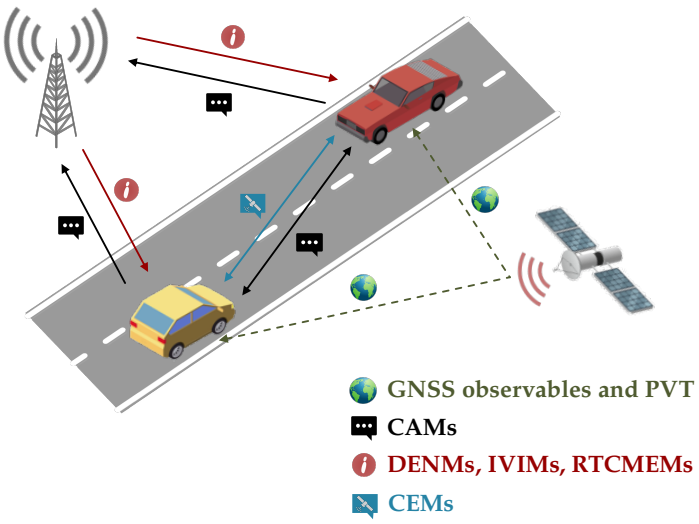


Figure 5.2: Scheme of a road segment showing two vehicles and one infrastructure node transmitting and receiving different ETSI standard-compliant messages, alongside the proposed CEMs for the exchange of raw GNSS data. In this scenario, vehicles are equipped with both V2X technologies for wireless communication and accurate GNSS receivers.

5.1.4 The CEM message and protocol

The CEM protocol hinges upon the key idea of Cooperative Enhancement (CE), i.e., enhancing the cooperative awareness, perception and localization capabilities in vehicles by adopting the incorporation of raw data to support the GNSS-based cooperative paradigms described in the previous Section.

Our protocol is a dedicated solution for the exchange of raw GNSS data in vehicular networks, released with open specifications. This novel protocol has been designed to leverage the CEM messages, which are used to encoded and transmit raw GNSS data among the network nodes. Although CEMs are meant to complement the information provided by CAMs, the CEM protocol can also be easily adapted to be employed as standalone. It is indeed worth highlighting how CEMs can be used both in standalone fashion, as well as in parallel with other kinds of messages such as RTCMEMs, if required by the overlying V2X applications. Furthermore, even if the main purpose of CEMs is to be exchanged between vehicles through V2V communications (as depicted in Figure 5.2), they can also be exchanged between moving nodes and the network infrastructure (e.g., by V2I or V2N communication).

The general format of a CEM message is depicted in Figure 5.3.

CEM messages have been designed to be fully ETSI-compatible. Indeed, as can be seen, each CEM includes an ITS Packet Data Unit (PDU) Header, as in CAMs

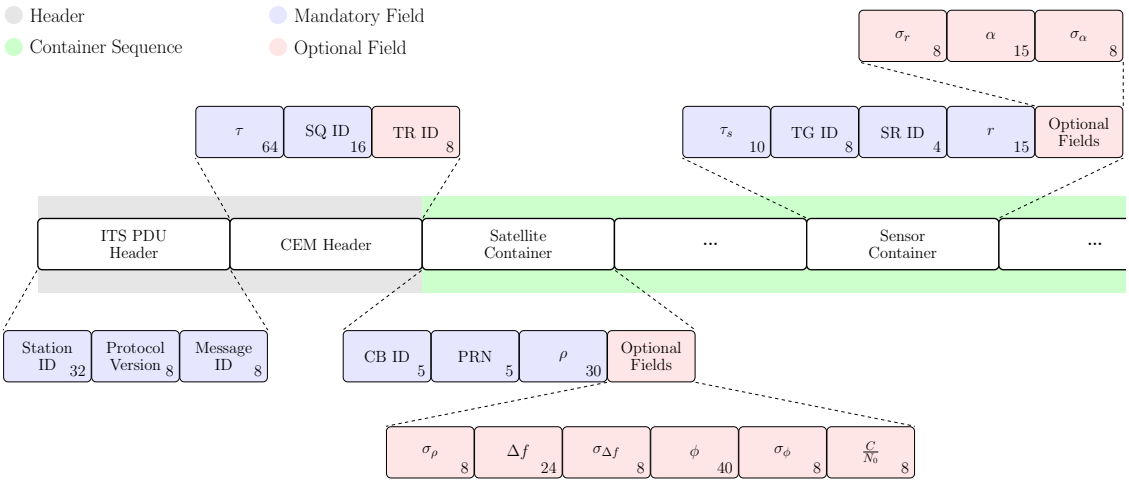


Figure 5.3: Structure of a CEM *I* frame. Field size (bit) is indicated by a value in the bottom corner of each field. *D* frames are similar to *I* frames, but contain an additional mandatory sequence identifier (DSQ ID) in their CEM header. The difference between *I* and *D* frames is explained in the "CEM Encoding" subsection below.

[63] and in DENMs [64]. We also defined a dedicated message ID, following the specifications from ETSI which reserve a set of IDs from 1 to 14, but allow this field to assume values up to 255. In the case of CEMs, this ID corresponds to `200`.

Furthermore, CEMs can be potentially encapsulated into any transport and network layer, such as UDP over IP, or the ETSI ITS-G5 Transport and Networking layers [21], [59], depending on the user needs. As mentioned in Section 2.2.4, the latter normally represents the stack of choice for the dissemination of vehicular messages in Europe, and CEMs are thus designed to fully support their encapsulation inside BTP and GeoNetworking. In particular, CEMs should use the BTP-B header when being encapsulated inside BTP, as this kind of header is normally used for broadcast messages. Concerning the BTP port number, we defined a new value for the CEM protocol, which is set to `2200`, since it is currently unused by ETSI. As in the case of CAMs, CEMs are designed to be specified by means of the ASN.1 description language, encoded with UPER and broadcasted with GeoNetworking SHB routing (i.e, only to the nearest first hop) [21].

After the ITS PDU header, CEM messages include the following network and GNSS data and observables:

1. τ (in the *CEM header*): accurate timestamp of when the observables were measured;
2. SQ ID (in the *CEM header*): sequence number of the current message;
3. TR ID (in the *CEM header*): identifier of the transmitting node;

4. CB ID (for each *satellite data container*): identifier of the constellation and signal frequency;
5. PRN (for each *satellite data container*): identifier of the satellite from which observables are measured; the receiver can infer the satellite identifier since each one transmits a unique pseudo-random code;
6. Pseudorange ρ (for each *satellite data container*): a value in meters of the distance between satellite and receiver (it also accounts for the time difference between transmitter and receiver clocks);
7. Carrier-phase ϕ (for each *satellite data container*): a value of distance expressed between satellite and receiver, but expressed instead in terms of number of phase cycles of the carrier frequency;
8. Doppler Δf (for each *satellite data container*): shift in received frequency caused by the relative velocity between satellite and receiver;
9. Variance σ (for each *satellite data container*): a value of the measurements uncertainty that can be estimated by receivers (one value is associated with each observable ρ, ϕ, and Δf);
10. $\frac{C}{N_0}$ (for each *satellite data container*): Carrier-to-noise ratio.

As can be seen in Figure 5.3, each CEM frame consists of a single CEM header, followed by a sequence of up to 10 *satellite containers*, one for each signal received from GNSS satellites. After the satellite containers, a sequence of *sensor containers* can be optionally included, as better detailed in Section 5.1.5.

The header of CEMs contains a single timestamp, which can be used to synchronize measurements from different nodes, all of which in principle work independently from one another and take measurements at different time instants (and possibly different rates as well). It is worth mentioning how synchronization can be fundamental to enable most cooperative applications.

This timestamp is defined as the number of nanoseconds from January 1st 2004, at midnight Coordinated Universal Time (i.e., 2004-01-01T00:00:00.000Z), represented over 64 bits. The format is compliant with the ETSI standards, since it represents the same format of timestamps stored inside CAM messages [65].

A transmitting node ID (TR ID) can optionally be included in the header, with a value ranging from 0 to 255 (i.e., over 8 bits). This ID can be used to define any ad-hoc strategy for the identification of the nodes involved in a CP approach. If not specified, the CEM protocol will use the standard ETSI *Station ID*. The usage of identifiers (either ad-hoc or based on the station ID) is needed when including *sensor containers* from other ranging sensors, as described in Section 5.1.5.

We also plan to include in future versions of the protocol the possibility of encoding other general purpose information in the header, together with application-specific flags. The inclusion of these fields in the header, however, should not contribute to a significant increase in the packet size, and thus should not noticeably influence the overall network performance, as assessed in this thesis.

All the other information is encoded instead in the *satellite containers*. The (optional) measurement uncertainty σ can be included in every container for each of the three types of GNSS observables encoded in the same container (i.e., up to three σ values are included, one for the pseudorange ρ, one for the carrier-phase ϕ, one for the doppler Δf).

Is it worth mentioning how the current version of the protocol foresees up to 10 containers. If more than 10 satellites are visible, the node transmitting CEMs has the freedom to select the best signals and satellites according to a given logic, for instance based on the best carrier-to-noise ratio.

CEM Encoding: *I* and *D* frame types

When dealing with vehicular networks and their strict latency and throughput requirements, optimized network usage is often an important challenge. Indeed, properly designing a communication protocol can noticeably improve the network resilience to a high vehicle density, and provide a better overall end-to-end latency.

With this aim, the CEM protocol employs a low-complexity differential encoding, with the definition of two types of CEM messages (also referred to as *CEM frames*):

- **Intra-message (*I*):** this message is one described earlier, and it contains the full value of measurements of both high and low frequency data;
- **Differential message (*D*):** this message encodes differential values with respect to the most recent I frame. These messages are transmitted in-between Is, and include only high frequency satellite data.

The I frames are transmitted at a fixed frequency of 1 Hz. In-between two I frames, up to 9 D frames can be transmitted with a basic interval of $100ms$. The actual rate of D frames (and, consequently, how many of them fit between Is) can be adjusted by the transmitting node, depending on the target CP application or on the congestion of the network. If no specific strategy is adopted, the node is expected to generate 9 Ds between two Is. A set of standard strategies which can be leveraged for the adjustment of the Ds transmission rate are currently being specified for the next version of the CEM protocol. The choice of such time intervals between frames has been done taking into consideration common vehicle dynamics in urban environment and the common rate of GNSS receivers. These values are also in line with the maximum frequency foreseen by ETSI for the CAM messages [63].

Figure 5.4 shows an example of transmission of I and D CEM messages according to the logic herein presented.

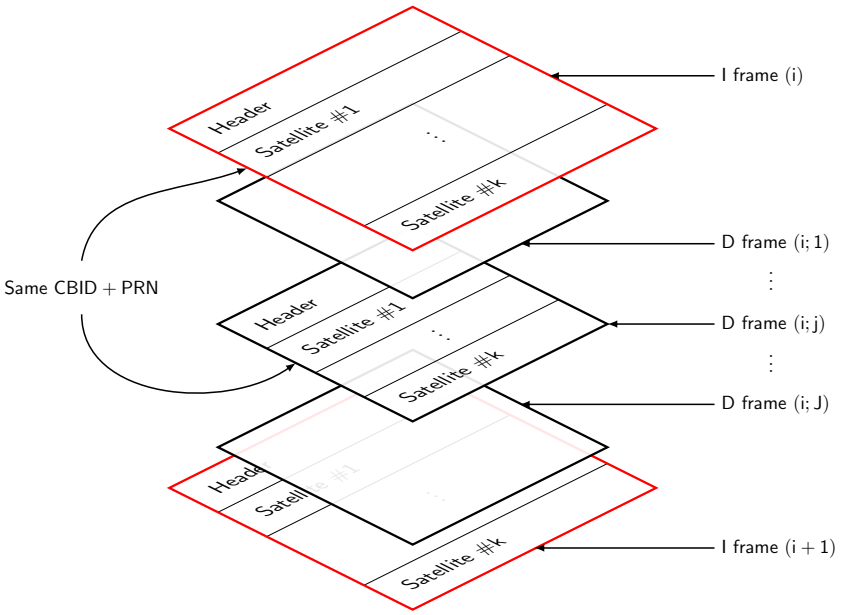


Figure 5.4: Example of a stream of CEMs with I and D frames. The node sends J Differential messages within two Intra-messages. Each message contains information related to k satellite signals. In absence of a specific rate adjustment strategy, $J = 9$.

It is worth mentioning that higher rates might facilitate time synchronization between received and local measurements. However, recent research on time calibration between GNSS and Ultra Wideband measurements reveal that time misalignment between multi-sensor range measurements can be compensated through measurement integration algorithms [225], [226]. Higher rates than the proposed ones would thus bring little benefit to the CP applications, while significantly increasing the load on the wireless channel.

I frames description

I frames are the ones depicted in Figure 5.3. The CEM header of Is contains a unique sequence identifier (SQ ID). Then, satellite containers include, as mandatory data, both the CB ID and PRN identifiers, together with the pseudorange measurements, while all the other fields are optional. These optional fields can be included depending on the availability of such data and/or on the requirements of the target CP application.

D frames description

The header of *D* frames is very similar to the one of *I* frames, with the addition of a second sequence number (the *Differential Sequence Identifier*, DSQ ID). Thus, a *D* frame CEM header includes:

- τ, as in *I* frames;
- SQ ID, as the identifier of the last *I*;
- DSQ ID, as its own unique message sequence identifier;
- Optional TR ID, as in *I* frames.

The addition of these sequence identifiers enable an easy sequence reconstruction by the receiver if needed, together with the association of *D* frames to the proper *I* frame, in case of out-of-order packets.

Concerning the container sequence, only the GNSS observables are sent within every *D* frame satellite container, and only differential pseudorange values are mandatory. Therefore, other low frequency data that does not need to be updated often, such as the variances and the carrier-to-noise ratio, is not included, and can be averaged by the transmitter in case of severe fluctuations. The CB ID and PRN values are also not encoded in *D* frames, as opposed to *I* frames, as measurements from different signals are designed to appear in the same order as in the most recent *I* frame. These values can be thus easily inferred from the last *I* frame. This allows the protocol to keep the size of *D* frames quite low.

Is it worth mentioning how the low complexity of this differential approach well suits the deployment on automotive embedded systems, such as the OBUs described in Section 3.1 and 3.5.

Optional CAM container

As mention in Section 5.1.4, CEMs are designed to complement the information already available thanks to the exchange of CAMs.

However, with the aim of making our protocol as much standalone as possible, it is envisioned to design an additional optional container, which can be included in case, for any reason, CAMs are not available or they are not being received for longer than a configurable amount of time. This container, which is going to be specified in the next CEM protocol version, is going to store *post-processed* GNSS and kinematic information, such as latitude, longitude, speed, heading, acceleration and estimates on their variance. All this information is going to be adapted to the transmission through the differential approach described earlier.

5.1.5 Sensor Containers for other ranging sensors

Other than satellite GNSS data, the CEM protocol is flexible enough to carry information from other on-board ranging sensors. This data is encoded in a sequence of optional *sensor containers*, which, if present, are always placed after the sequence of satellite containers.

Data from other ranging sensors can be very useful under harsh conditions, such as in deep urban canyons, in which the visibility of GNSS satellites may be reduced, causing a degradation of the quality of the received signal. In these cases, agents may want to rely on measurements from on-board sensors to improve their positioning capabilities. Sensor containers enable the exchange of such range measurements and sensors data between vehicles (and, possibly, also between vehicles and the infrastructure, depending on the CP application). The supported ranging sensor data can be obtained from several on-board sensors such as radars, LiDARs, cameras, or Ultra Wideband interfaces.

Other than providing useful data in harsh conditions, the addition of *sensor containers* also enables several CP applications which require combining both GNSS data and other ranging measurements to further enhance the positioning performance.

A sensor container includes additional cooperative information from other sensors available in a vehicle, and encodes the following values:

- τ_s *(mandatory)*: accurate time difference between GNSS and sensor measurements;
- Target ID (TG ID) *(mandatory)*: identifier of target vehicles relative to which the sensor measurements are taken;
- Sensor ID (SR ID) *(mandatory)*: identifier of the type of sensor or signal used to obtain the measurement; as only some values are defined for known sensors, the CEM protocol is flexible enough to easily accommodate data from new types sensors other than the ones mentioned earlier, which can use currently unused SR ID values;
- Range r *(mandatory)*: a measurement in meters of the distance between the transmitting vehicle and the target one;
- Angle α *(optional)*: angle of the target vehicle with respect to a common reference frame;
- Variance σ_s *(optional)*: an estimate of the measurements uncertainty (associated to r and α).

As mentioned earlier, an ID of the transmitting vehicle is included in the CEM header when one or more cooperative satellite or sensor containers are present in

the CEM packet. Thanks to the presence of this identifier, each of the sensor containers can include a TG ID (stored in 8 bits) to uniquely identify the target vehicle with respect to which the measurements are taken. In this way, transmitting and receiving vehicles can all unambiguously identify each other, and properly associate and collect all measurement regarding each vehicle. Each sensor container also encodes an SR ID, enabling any vehicle to integrate the received cooperative measurement using an error model based on the type of sensor from which the measurement was obtained. The protocol can define up to 16 different types of ranging sensors, only some of which are reserved for known sensors, i.e., `0x01` for radars, `0x02` for LiDARs, `0x03` for cameras, `0x04` for Ultra Wideband, while `0x00` is reserved. All the other values up to `0x0F` are available for future extensions of the protocol to new sensor types.

An additional scalar value, τ_s, is included in the sensor container to represent the difference between the CEM header timestamp (τ) and the time at which the measurements are obtained from the sensor. Indeed, other sensors may work asynchronously with respect to the GNSS receiver, and collect measurements with different rates. To optimize the network usage, this quantity is expressed as a differential scalar value τ_s, which is the time difference between the most recent measurement obtained from the sensor and the timestamp τ provided in the message header. This field encodes values in the range from 0 to 100 ms, with a granularity of 0.1 ms. Furthermore, sensors measurements are expected to be always less fresh than GNSS data carried by CEM, making τ_s always carry positive values. It should be noted that the minimum time resolution has been set based on current studies and investigations on GNSS and asynchronous sensor integration.

Finally, it is worth mentioning how sensor containers are designed to be inserted in CEMs after all available satellite containers. This, thanks to the UPER encoding rules, enables an easy message decoding at the receiver side, without the need for including a dedicated field carrying the type of each container.

Management of vehicle identifiers

When sensor containers are encoded in CEMs, the management of vehicle identifiers (i.e., TR ID and TG ID) becomes an important issue.

As mentioned earlier, each CEM message may contain an optional TR ID, when not relying on the ETSI *station ID* as vehicle identifier. The latter can indeed be fixed on a per-vehicle basis, or anonymized with different strategies [227], and its management according to the ETSI standards may not be suitable for all the CP approaches. We thus consider two possible scenarios for an ad-hoc assignment of TR IDs:

1. In case of a platoon of vehicles, which move together as described in Section 2.1.1, IDs can be fixed and assigned a-priori.

2. If vehicles are instead continuously moving in and out of range of each other, a dynamic allocation of IDs is required, such that two vehicles in range of each other cannot have the same ID.

Several strategies can be defined for any of these two cases, and they are currently not explicitly defined by the most recent CEM specifications. The specification of a set of standard strategies, to complement other user-defined solutions for the assignment of IDs, is currently undergoing for a future version of the CEM protocol.

Another issue that may arise when leveraging sensor containers is the data association problem. When a vehicle is measuring a distance with respect to another target vehicle using on-board sensors, it needs to know which vehicle it is, so that the corresponding ID can be associated to the measurement and included in the sensor container as TG ID. How to manage this task is out of the scope of this thesis and it is currently left to the specific CEM protocol implementation, but it is nevertheless an important challenge. A proper management of the IDs is indeed crucial for the correct utilization of cooperative sensor measurements.

5.1.6 Ranges of values

Satellite containers

The range of GNSS observable values in I frames, according to our analysis, are summarized in Table 5.1, which also shows the minimum and maximum values for CB ID and PRN. Likewise, the range of values for the D frames are described in Table 5.2.

Table 5.1: GNSS observables - CEM Intraframe (I)

Description	Symbol	Min. Value	Max. Value	Precision	Units	Bits
CB ID	n/a	1	31	n/a	n/a	5
PRN	n/a	1	31	n/a	n/a	5
Pseudorange	ρ	$1.9 \cdot 10^{10}$	$2.4 \cdot 10^{10}$	10^{-2}	m	30
Carrier phase	ϕ	$0.7 \cdot 10^{8}$	$1.6 \cdot 10^{8}$	10^{-3}	Cycles	40
Doppler shift	Δf	$-5.0 \cdot 10^{3}$	$5.0 \cdot 10^{3}$	10^{-3}	Hz	24
Carrier-to-Noise Ratio	C/N_0	20	70	0.25	$\mathrm{dB\,Hz^{-1}}$	8
Uncertainties	σ	0	20	10^{-2}	n/a	8

The ranges reported in the Tables are obtained through the analysis of real GNSS datasets. The last column of each Table shows the approximate amount of bits needed to represent the ranges with the corresponding accuracy, in order to provide a rough idea of the amount of data needed.

Table 5.2: GNSS observables - CEM differential frame (D)

Description	Symbol	Min. Value	Max. Value	Precision	Units	Bits
Pseudorange	ρ	$-1.0 \cdot 10^3$	$1.0 \cdot 10^3$	10^{-2}	m	18
Carrier phase	ϕ	$-5.5 \cdot 10^3$	$5.5 \cdot 10^3$	10^{-3}	Cycles	24
Carrier-to-Noise Ratio	Δf	$-3.0 \cdot 10^1$	$3.0 \cdot 10^1$	10^{-3}	Hz	16

Furthermore, the ranges for the *D* frames have been defined by considering the largest possible variation of observables over a time span of $900ms$, which is the maximum time between a *D* and the latest *I*.

All the uncertainties/variances are represented with an integer value from 0 to 200 (corresponding to values from 0 to 20 with a precision of 10^{-2}). The CEM protocol does not define any specific combinations of ranges and precision, as shown in the Tables, so that any implementation has the flexibility to map these values as needed. It should also be mentioned that, since information regarding the quality of signals is mainly exploited to obtain weighted estimates, its accuracy is not as crucial as for the measurements, and can be represented with only a few bits (roughly, 8 bits are needed).

Concerning the CB ID value, it can assume values from 1 to 31, with 0 being reserved for an unavailable value in case of errors (even though the CB ID is always expected to be filled in). The same applies to the PRN value, with 0 being reserved for "unavailable".

Table 5.3 summarizes the currently specified CB ID values for different satellite constellations. As can be seen, some ID numbers are valid, but currently unused (i.e., 4, 5, 9, 10, 16, 17, 21 to 31). These values can be reserved for a possible future addition of either new signals from the already included constellations or for supporting other constellations, such as the Japanese Quasi-Zenith Satellite System (QZSS).

Table 5.3: Constellations and Signal Bands (identifiers in round brackets are compliant with RINEX standard)

Constellation	SB #1	SB #2	SB#3	SB #4	SB #5
GPS	L1 (1)	L2 (2)	L5(3)	-	-
GLONASS	G1 (6)	G2 (7)	G3 (8)	-	-
Galileo	E1 (11)	E2 (12)	E5a (13)	E5b (14)	E6 (15)
BeiDou	B1 (18)	B2 (19)	B3 (20)	-	-

Finally, for each type of information listed in Tables 5.1 and 5.2, an additional value is always reserved with the meaning of "unavailable". As in CAMs for certain

data (e.g., acceleration and heading, which can indeed be unavailable), this value is usually leveraged in case the information is outside the defined bounds or not available at all. This could for example occur if a satellite is no longer visible, or if the receiver is unable to compute a certain value. In the first case, the container for that satellite will still be included (with unavailable data) in the following *D*s until the next *I*. The special value reserved for unavailable data in CEMs is always defined as the maximum value plus one time the corresponding precision (with the exception of the CB ID and PRN, as mentioned earlier), and it is therefore set to 201 for the uncertainties and the carrier-to-noise ratio.

Sensor containers

Concerning sensor containers, the ranges for I frames are summarized in Table 5.4.

Table 5.4: Sensor data - CEM Intraframe

Description	Symbol	Min. Value	Max. Value	Precision	Units	Bits
Range	r	0	300	10^{-2}	m	15
Angle	α	0	360	10^{-2}	Degree	15
Differential timestamp w.r.t. GNSS time	τ_s	0	100	10^{-1}	ms	10
Uncertainties	σ_s	0	20	10^{-2}	n/a	8

In particular:

- We considered a maximum value for the inter-vehicle range r, i.e., $300\,m$. This value has been determined by considering several factors, which include common vehicle dynamics in urban environments and the broadcasting range of the communication technology.
- Concerning the angle measurements, all values of α are possible, since, in general, vehicles could be anywhere around each other.
- The maximum value of tau_s has been set to $100\,ms$, considering the usual rate at which sensors obtain measurements and the basic rate of the CEM protocol.
- Variances (σ_s) are also defined for both r and α, as in satellite containers, to enable weighted estimation techniques. These uncertainties are represented again with a value from 0 to 200, with 201 being reserved for "unavailable".

The values for sensor containers in D frames are instead defined as follows:

- The full value of α is transmitted again at every D frame, due to the fast changing vehicle dynamics in urban environments.
- The maximum value for the differential range has been set to $80\,m$, considering realistic vehicle speeds and that a D frame can be generated at most 0.9 seconds after the corresponding I frame. An example of worst case calculation for the differential range can be found in [7].

It should be mentioned that, as in satellite containers, uncertainties are not encoded in D frames.

5.1.7 The SAMARCANDA dataset

With the aim of simulating and emulating vehicular scenarios with real GNSS data, affected by estimation errors, we build and disclose to the public an open dataset named Synthetic Accurate Multi-Agent RealistiC Assisted-gNss DatAset (SAMARCANDA). This dataset not only enables the simulation of vehicular scenarios with realistic positioning and kinematic data, but also the assessment of CP applications based on the CEM protocol and other positioning technologies for connected vehicles.

SAMARCANDA consists of accurate GNSS PVT data and RINEX files obtained from 19 different vehicular traces collected through a multi-band, multi-constellation Swift Piksi Multi GNSS/INS/RTK, high-accuracy receiver mounted on a car. More details on the used hardware and its setup are available in [7].

Furthermore, it comes in two versions:

- CSV version with accurate PVT data from all 19 vehicular traces, including geographical positions, speed, acceleration and heading of the different vehicles, each with a precise timestamp related to when the measurements were collected. ms-van3t can make use of this version thanks to the `gps-tc` module, which can be used, as mentioned in Chapter 4, as a mobility simulator instead of relying on the SUMO tool. This version has been made publicly available on GitHub[1].
- RINEX version with raw GNSS data from the vehicle traces, including a set of sample traces for the integration into ms-van3t-CAM2CEM.

The traces are related to a fleet of vehicles travelling in an area of approximately $50.34\,\text{km}^2$ around the city of Pinerolo, Italy. The trajectories of the 19 vehicular traces are depicted in Figure 5.5.

[1] https://raw.githubusercontent.com/francescoraves483/ms-van3t-CAM2CEM/master/src/gps-tc/examples/GPS-Traces-Sample/SAMARCANDA_dataset.csv

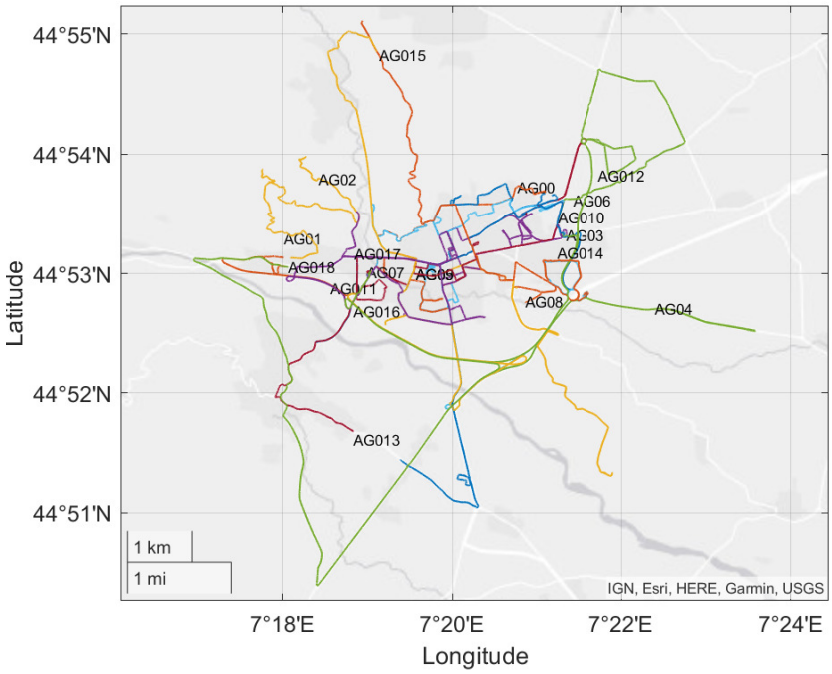


Figure 5.5: Map of real vehicular trajectories of the SAMARCANDA dataset. Labels and colors identify the different emulated vehicles.

These trajectories, although acquired asynchronously, are expected to be reproduced according to a synchronous scheduling (for instance, in ms-van3t) in order to simulate up to 19 vehicles travelling in urban and suburban areas.

5.1.8 ms-van3t-CAM2CEM

With the aim of extensively evaluating the CEM protocol, using at the same time an open source tool, we have developed a dedicated version of ms-van3t (described in Section 4.1). This version, named *ms-van3t-CAM2CEM*, comes with several enhancements with respect to the original framework.

Specifically, the ms-van3t framework has been enhanced as follows [7]:

- We have generated the encoding and decoding functions for the CEM messages, thanks to the *asn1c* tool [70]. *asn1c* enables the automatic generation of C/C++ message handling functions starting from any ASN.1 data structure. In our case, we fed the tool with the latest version of the CEM specifications, and integrated the resulting code into ms-van3t.

- We have implemented and integrated a novel *CE basic service* managing the CEM protocol, including *(i)* the management of the differential encoding

scheme and of the optional containers and fields, *(ii)* the transmission of CEM messages at the proper frequency, *(iii)* the reception of CEM messages and the extraction of the relevant data, which is then made available to the V2X applications; this service provides an easy-to-use interface to the ms-van3t user thanks to callback functions;

- We have adapted the ETSI BTP implementation to support CEMs, by coding a new port number, as defined in Section 5.1.4.
- We have developed a set of dedicated examples, which can be used as a starting point to simulate the exchange of CEMs in urban scenarios, with different access technologies (IEEE 802.11p, Release 14 LTE-V2X);
- We have integrated the CSV version of the SAMARCANDA dataset, as mentioned earlier.
- We have implemented and integrated a special module, called `gps-raw-tc`, which is able to read raw GNSS traces (i.e., files containing raw GNSS observables coming from pre-recorded vehicular traces) and provide them as input to the CE basic service for the generation of CEMs. The newly created module can work in conjunction with SUMO, when sample raw GNSS traces are assigned to simulated vehicles (i.e., traces with exactly the same data types and ranges as real traces, but without application-layer informative content) for the evaluation of network-related parameters. It can also work with `gps-tc`, to simulate a real scenario, where vehicles are generating CEMs with actual application-layer content, for the evaluation of full-fledged CP approaches;
- Finally, we integrated a sample raw GNSS trace from the SAMARCANDA dataset, with the aim of simulating the transmission and reception of GNSS observables and evaluate the network performance when CEMs are being transmitted besides other standard-compliant messages (mainly, CAMs).

ms-van3t-CAM2CEM is specifically designed for the evaluation of CP approaches. We have made it available on GitHub with an open source license [25].

It should be remarked that ms-van3t-CAM2CEM can not only be used to gather network KPIs, such as the ones presented in Section 5.1.9, but also to test CP applications relying on the CEM protocol. It could also be used in emulation mode, as part of a real testbed, to send and receive CEMs (and CAMs) from an external entity implementing our protocol. The implementation of a CE Basic Service and of the related encoding and decoding function is thus one of the major features of this special version of ms-van3t.

5.1.9 Evaluation of the CEM protocol with IEEE 802.11p and LTE-V2X

After the definition of the CEM protocol, we exploited ms-van3t-CAM2CEM to perform an extensive simulation campaign, aimed at validating the CEM protocol from a networking point of view, when vehicles exchange both CAMs and CEMs in a realistic urban scenario.

After an initial set of tests performed using the SAMARCANDA dataset, we present a detailed scalability analysis, in order to assess the network behaviour when the number of connected vehicles grows large and with different access technologies. For such a purpose, we set up a dedicated simulation scenario on the ms-van3t-CAM2CEM framework, which in turn uses SUMO to represent the vehicles' mobility. In particular, since our main goal is to thoroughly evaluate the network performance when CP approaches are enabled by the CEM protocol, we have used SUMO to simulate the vehicles trajectories, instead of directly relying on the SAMARCANDA dataset. This choice was led by the possibility of increasing the number of vehicles in SUMO without any hard limit (other than the one imposed by the hardware running ms-van3t-CAM2CEM), and, hence, of analyzing the network behaviour with a higher number of vehicles than what would be available in SAMARCANDA.

Following this approach, each vehicle is assigned a sample raw GNSS trace from the SAMARCANDA dataset, as SUMO cannot provide such information. Even if the data is sample data, not directly related to the simulated positions in SUMO, this methodology allows us to simulate the exact range of values, data types, and data size that we would expect in real-world CEMs. The urban scenario we considered for this evaluation campaign is depicted in Figure 5.6. As can be seen, each vehicle is equipped with *(i)* a CA Basic Service, for the exchange of CAMs, *(ii)* a CE Basic Service, for the exchange of CEMs, *(iii)* a sample CP application receiving data from both Basic Services and storing it locally, *(iv)* an ETSI ITS-G5 stack with the GeoNetworking and BTP layers, and *(v)* a radio interface (including the MAC and physical layers) either based on IEEE 802.11p or on C-V2X Mode 4. All the CAMs and CEMs are broadcasted after being encapsulated into the BTP and GeoNetworking layers, following the rules defined in Section 5.1.4.

To evaluate our proposed protocol in different operational conditions, we mainly consider two access technologies for V2X communications, i.e., IEEE 802.11p and Release 14 LTE-V2X Mode 4 (which will be referred to, in the next Sections, as C-V2X Mode 4). Both technologies are fully supported by ms-van3t-CAM2CEM, through state-of-the-art models, as in the standard version of ms-van3t [32]. Even though both Mode 3 and Mode 4 are foreseen for LTE-V2X, in our analysis we focus on Mode 4, as it is supported in ms-van3t and allows us to obtain a direct comparison against IEEE 802.11p while exchanging CEMs in V2V scenarios.

The performance metrics that we investigate are: *(i)* the total transmission rate,

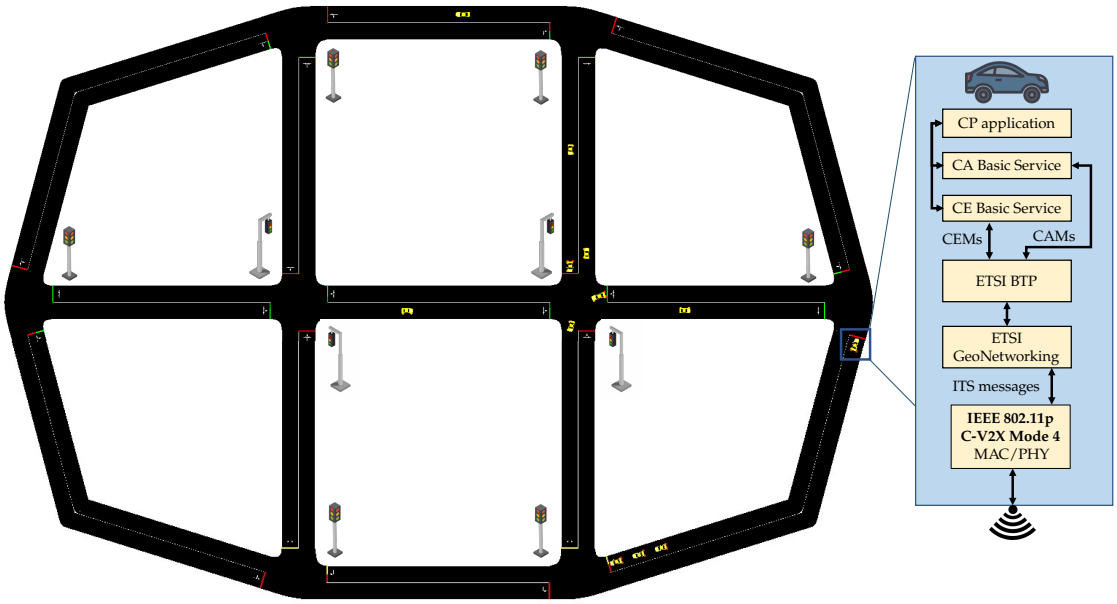


Figure 5.6: The simulated urban scenario of choice for the CEM evaluation from a network standpoint. It is composed of several intersections regulated by traffic lights, with two central intersections and an outer ring with one lane per direction of travel. The scheme also depicts the different modules which are deployed on each simulated vehicle.

expressed as the number of bytes per second transmitted in the wireless medium by vehicles as their density grows larger, *(ii)* the average one-way latency between a transmitting vehicle and all the vehicles receiving the message (we recall that both CAMs and CEMs are normally broadcasted), and *(iii)* the Packet Delivery Ratio (PDR), also called Packet Reception Ratio (PRR) by 3GPP [178] (in the following we will use PRR to refer to such metric). As described in Section 4.1, the latter represents a measure of network reliability, as it accounts for the number of packets that are lost due to channel collisions, interference, or harsh propagation conditions. Both latency and PRR are computed thanks to the ms-van3t `PRRsupervisor` module. It should be recalled that this module computes the PRR following the 3GPP TR 36.885 specifications [178], which report a way to compute the packet loss, or better, the overall PRR, when packets are broadcasted. The PRR computation baseline has been varied, in this case, from 100 m to 200 m.

All the metrics (except for the transmission rate) are investigated for an increasing number of vehicles, up to a density of about 33 veh/km, and a varying baseline distance and transmission power. Indeed, two transmission power levels are considered: 23 dBm, as it represents a typical value for the exchange of vehicular messages [73], and 33 dBm. For each access technology and scenario, results are

averaged over 10 different experiments, each corresponding to a randomly selected traffic pattern. Each plot also reports the 95% confidence intervals.

Finally, concerning the LTE-V2X Mode 4 configuration, we selected an RRI of 20 ms (i.e., the minimum available value in Release 14, to try to reduce as much as possible the latency due to the SPS scheduling) and a probability of keeping the current resources $P = 0.4$.

Total transmission rate

The first set of results is related to the total transmission rate as the vehicle density increases. These results do not depend upon the actual access technology and show how much the wireless medium is potentially used when transmitting both CAMs and CEMs in an urban scenario. The obtained values are depicted in Figure 5.7, in which the y axis represents the total transmission rate, in Mbit/s, computed over all vehicles in the considered scenario.

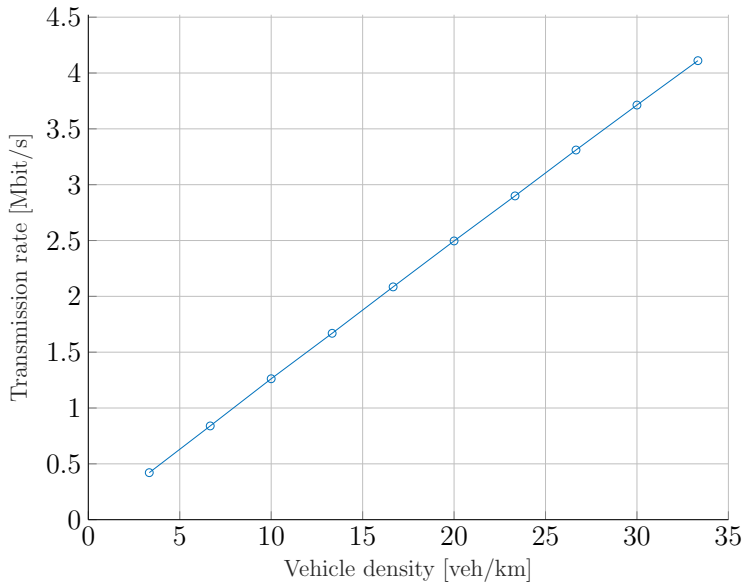


Figure 5.7: Total transmission rate, representing the amount of bytes transmitted per second as a function of the vehicle density, when both CAMs and CEMs are being actively transmitted by vehicles.

The first important outcome is the almost linear proportionality with respect to the vehicle density, which shows how the communication complexity of the proposed protocol is relatively low. Indeed, the system scales almost linearly with the vehicle density, and thus also with the number of nodes participating in the CP process. This is due to CEMs being mainly designed to be broadcasted by vehicles.

A second relevant outcome is represented by the maximum transmission rate that is achieved under very high vehicle density, namely, 4.1 Mbit/s. As this is a fairly low value, this figure confirms that the proposed solution can work well even when the network is congested and the available throughput is limited.

Latency

The latency results are instead depicted in Figure 5.8. It should be noted that these results do not depend on the value of baseline distance (i.e., either 100 m, 150 m or 200 m), which is a parameter for the computation of the PRR only.

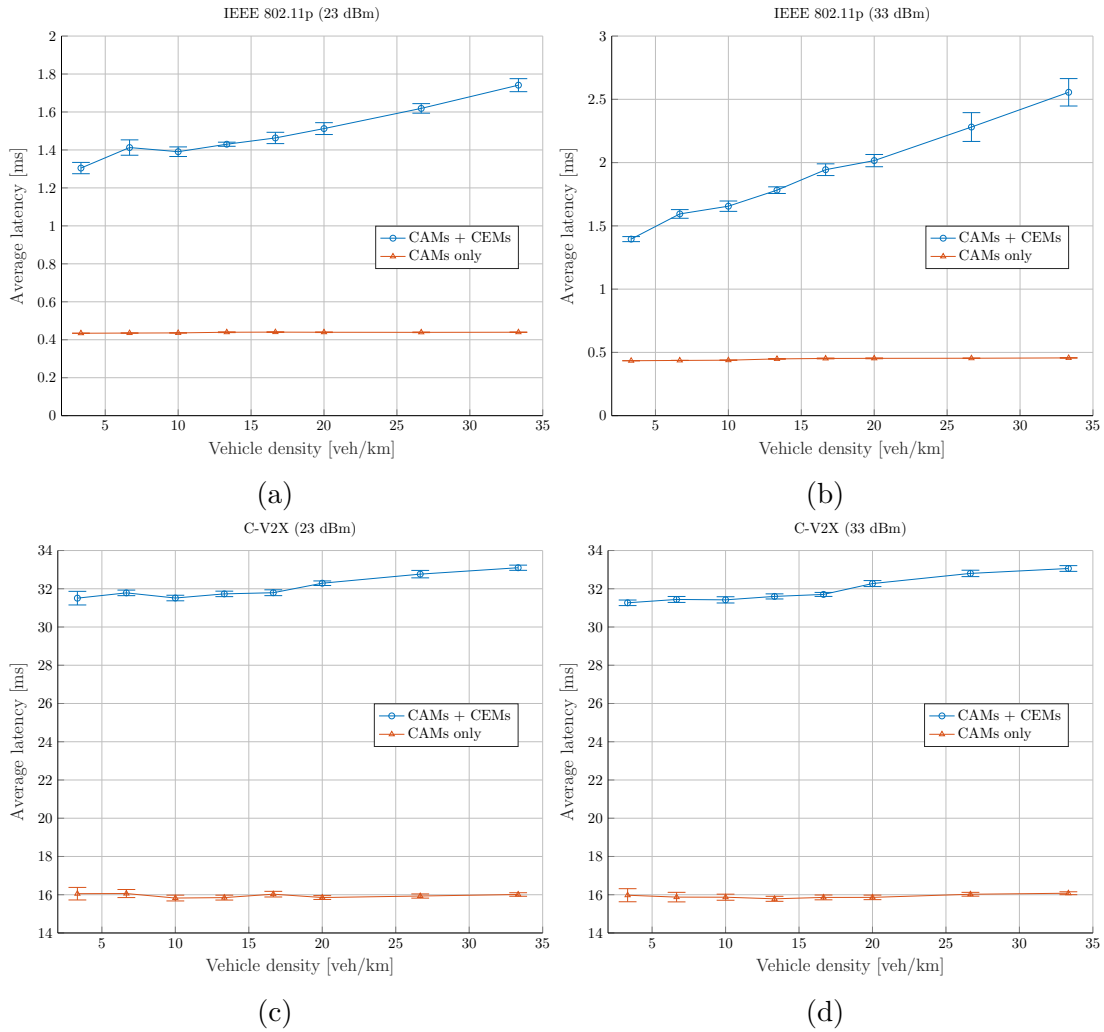


Figure 5.8: Average one-way latency with respect to vehicle density. (a) IEEE 802.11p at 23 dBm (b) IEEE 802.11p at 33 dBm (c) C-V2X at 23 dBm (d) C-V2X at 33 dBm

As can be seen, IEEE 802.11p can provide an overall average latency which

is significantly lower than the one provided by C-V2X, even for a relatively large vehicle density. In particular, the IEEE-based access technology can provide a latency of around 0.4 ms when only CAMs are transmitted, which increases to, on average, 1.3-2.5 ms (depending upon the vehicle density) when both CAMs and CEMs are used. This is due to the larger size of CEMs, compared to the one of standard CAMs. Indeed, in our simulation, the size of the CEMs can range from 496 B (smallest D frames) to 860 B (largest I frames), while CAMs reach up to a maximum of 121 B. However, despite the increased latency, the delays remain very low and compatible with all the safety critical applications which may require, according to ETSI, a maximum end-to-end latency between 50 and 300 ms (depending on the specific application) [8]–[10].

The latency observed under C-V2X Mode 4 is instead higher, with values around 16 ms for the transmission of CAMs only, and 32 ms in the case of concurrent transmission of CAMs and CEMs. Such increase in latency is again due to the larger size of the CEM messages. As the amount of data required to exchange raw GNSS information is higher than what is normally transmitted within CAMs, this latency increase also shows the importance of adopting a differential encoding to reduce as much as possible the impact of the transfer of GNSS raw data on the underlying network.

These results are also in line with past simulation studies comparing the performance of C-V2X and IEEE 802.11p [79]. It is also worth noticing that increasing the transmission power does not lead to changes in the order of magnitude of the measured values, but it makes the IEEE 802.11p latency increase more as the vehicle density grows, due to higher interference levels. This effect is not observed in the case of C-V2X, which suffers less severely from highly congested network conditions. Thus, when transmitting CEMs over C-V2X, the latency remains almost constant, regardless of the selected transmission power level. This suggests that, when leveraging C-V2X and transmitting both CAMs and CEMs, a higher transmission power would be desirable, as it has a minimal impact on latency, while guaranteeing a higher PRR, as described below.

Packet Reception Ratio

The PRR results are depicted in Figure 5.9.

Each plot depicts the average PRR as the vehicle density grows from low values (around 3.4 vehicle/km) to a relatively high traffic scenario (with 33.4 vehicle/km). The lines corresponding to the transmission of CAMs only are marked by triangles, while those corresponding to the concurrent transmission of CAMs and CEMs are marked by circles. Each color corresponds to a different baseline value, i.e., 100 m, 150 m and 200 m.

The first interesting result comes from the trend of the PRR as the vehicle density grows larger, as different access technologies exhibit different behaviour for

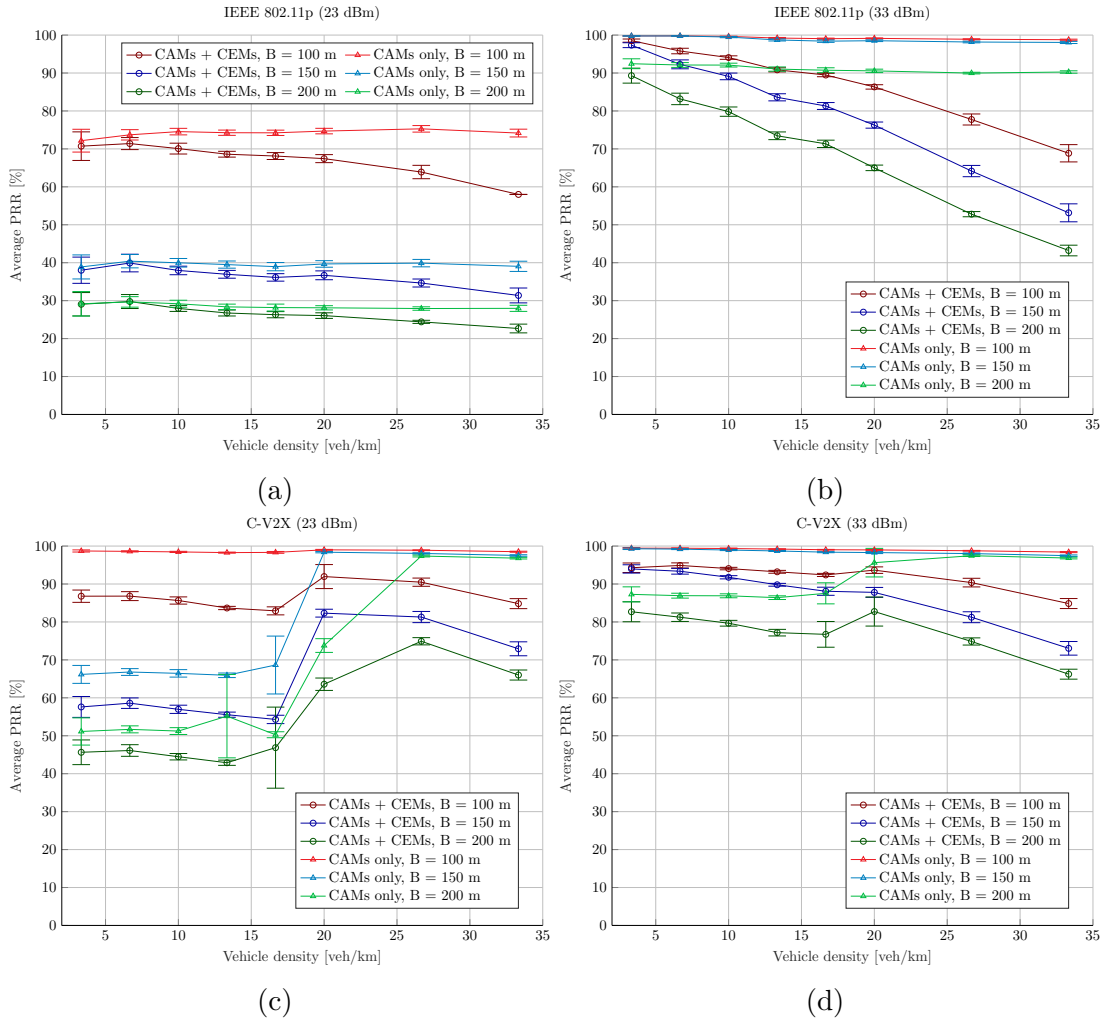


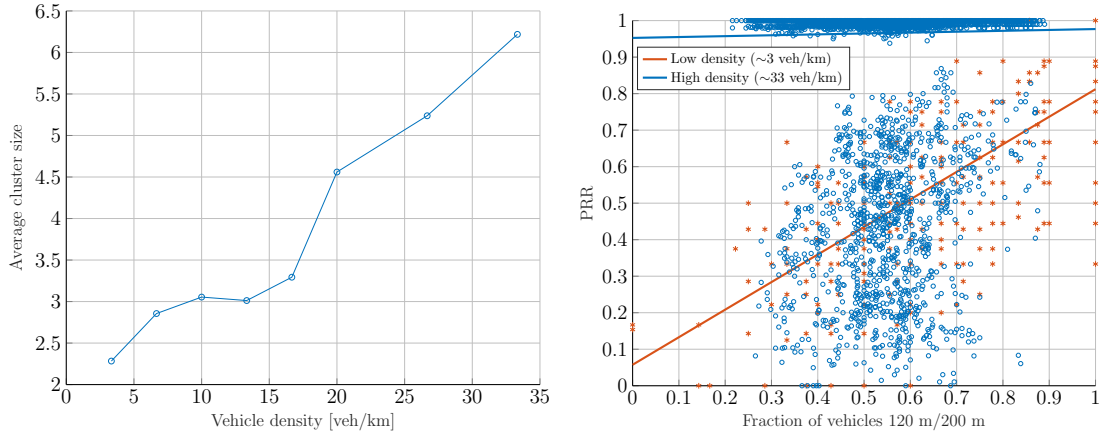
Figure 5.9: Packet Reception Ratio, in percentage as a function of vehicle density, for three different values of baseline distance, i.e., 100 m, 150 m and 200 m. (a) IEEE 802.11p at 23 dBm (b) IEEE 802.11p at 33 dBm (c) C-V2X at 23 dBm (d) C-V2X at 33 dBm.

highly congested scenarios. Indeed, IEEE 802.11p experiences a monotonically reduced PRR when the vehicle density increases, which is especially evident when transmitting both CAMs and CEMs. This is due to the contention-based channel access, which suffers from increased collisions and backoff times when a large number of vehicles try to access the same shared wireless medium. Interestingly, C-V2X Mode 4 experiences instead a slight reduction until 16.67 vehicle/km, followed by a sudden increase for higher vehicle densities, when a lower transmission power (e.g., 23 dBm) is used. Taking the baseline of 200 m as a reference, for the 23 dBm case, the *CAMs + CEMs* PRR increases from 46.86% with 16.67 veh/km to 63.58% with 20 veh/km. Such effect is less evident when increasing the value of transmission

power.

The observed trend is mainly due to the resilience of C-V2X when there is a large number of communicating nodes. Simulating the same scenarios but using a very high transmission power (i.e., 42 dBm, which is not shown here as it would represent an unrealistically high value) led, in both the *CAM only* and *CAMs + CEMs* scenarios, to PRR values always around 100%. The behaviour of the PRR when the vehicle density increases is thus mainly due to the selected level of transmission power, which may not be sufficient to reach all the vehicles within the desired baseline. This fact also explains why high transmission power levels (i.e., 33 dBm), or, equivalently, smaller baseline values (i.e., 100 m), make this effect much less noticeable.

Taking this into account, we were able to evaluate how in urban scenarios, like the one considered here, vehicles tend to form clusters. This occurs especially in correspondence of regulated intersections. Figure 5.10a depicts the mean cluster size as a function of the vehicle density, taking as reference one of the ten traffic patterns used to gather the PRR results. A cluster is defined here as a group of vehicles with a reciprocal distance lower than 20 m. As can be seen, the mean cluster size almost doubles between 3 veh/km and 33 veh/km, making the number of vehicles within the baseline that can be reached by C-V2X Mode 4 grow larger.



(a) (b)

Figure 5.10: Analysis of the C-V2X Mode 4 PRR behaviour, taking as reference one of the traffic patterns in the urban scenario depicted in Figure 5.6, a transmission power of 23 dBm and a baseline of 200 m. (a) Mean vehicle cluster size (b) PRR, for each transmitted packet, as a function of the fraction of vehicles within 120 m, with respect to the total amount within the 200 m baseline; the straight lines show a linear interpolation of the single PRR values.

Figure 5.10b shows instead the relationship between the obtained PRR values and the ratio between the number of vehicles within 120 m and the total number

within a 200 m baseline. As a reference, a transmission power of 23 dBm is considered, as the C-V2X PRR increase is much more evident for low transmission power levels. The dependence between the PRR values and the fraction of vehicles is significantly less when high density scenarios are considered, with higher values of PRR becoming more likely even when the fraction of vehicles is relatively low.

Next, we look at the comparison between the PRR values obtained with a 23 dBm transmission power and those achieved at 33 dBm. A transmission power higher than 23 dBm is always desirable, as it provides higher values of PRR in all cases and under any of the selected technologies. Indeed, a transmission power of 23 dBm is not always sufficient to reach a transmission range greater than 100 m, thus leading to low packet reception ratios for both the baseline values of 150 m and 200 m. Even though a range of 100 m may be enough for enabling several CP approaches, it can be important to take this into account, especially when targeting use cases which may require larger single-hop ranges. It can also be observed that the C-V2X technology provides, overall, better PRR results than IEEE 802.11p, especially when the vehicle density grows, hence more devices try to access the shared wireless channel. C-V2X with 33 dBm-transmission power can indeed provide a PRR which is always higher than 77%, for a vehicle density up to 16.67 veh/km (which already represents a quite congested scenario for most CP approaches), no matter the selected baseline distance. Furthermore, the same transmission power level can guarantee an average PRR higher than 66%, even for a density of 33.4 veh/km and a large baseline of 200 m.

Looking instead at the comparison between the case where both CAMs and CEMs are transmitted and the one where only CAMs are used, one can notice how the use of CEMs slightly reduces the overall PRR, due to the larger size of our messages and of their more frequent transmission. This reduction is, however, limited, especially in the presence of mid-to-low connected vehicle densities, and does not significantly impact several relevant CP use cases enabled by our protocol. In particular, a range of 100 m with a PRR higher than 90% can enable the development and testing of different cooperative approaches for high-precision localisation. Furthermore, focusing on the highest level of transmission power level (i.e., 33 dBm) and on the exchange of both CAMs and CEMs, IEEE 802.11p showcases a PRR which is much lower than the one of C-V2X, when the number of connected vehicles increases. As an example, for 16.67 veh/km density and a 100 m baseline distance, we observe a reduction on the PRR, with respect to the transmission of CAMs only, by 9.67% for IEEE 802.11p and 6.71% for C-V2X. Once again, this is due to the contention-based nature of the IEEE 802.11p access technology, which suffers from increased collisions and longer backoff procedures as more vehicles try to access the shared medium.

The PRR results also provide useful insights on how the CEM protocol specifications can be improved in the near future. In particular, a future version of the CEM protocol should include:

- The possibility of transmitting the I and D frames at varying periodicity, depending on the actual GNSS raw data, which may help to improve the overall PRR, as less D frames would be transmitted when the deviation from the preceding I frame is not big enough; this would also help to reduce the likelihood of concurrent CAMs and/or CEMs transmissions from different vehicles;
- The possibility of dynamically reducing the number of D frames in case of detected channel congestion;
- A DCC mechanism to automatically reduce, or increase, the transmission power depending on the real-time PRR performance and on the channel congestion, thus leading to an optimal setting of the transmission power. The same mechanism could be realized in an agnostic way with respect to the underlying access technology (i.e., either IEEE 802.11p or LTE-V2X), which would just need to tune the maximum transmission power depending on the inputs from a CEM DCC module.

Take-away messages

Our results highlight how the CEM protocol is an effective way of exchanging raw GNSS data between connected vehicles, using a fully open protocol. Furthermore, the CEM protocol showed to be well-suited to work with both the IEEE 802.11p and C-V2X Mode 4 technologies, depending on the latency and reliability requirements of the overlying CP application. We have also demonstrated that, in order to reach all vehicles within a range of at least 100 m with a reliable PRR, a transmission power of 23 dBm may not be sufficient. On the contrary, 33 dBm can guarantee a PRR higher than 90% at a 100 m-distance, for up to 16.67 veh/km, which represents an already congested value for most CP approaches. We have also provided an insight on how IEEE 802.11p yields a better latency, when it is used to transmit CEMs to nearby vehicles, while C-V2X showed to be more resilient to channel congestion.

Finally, the design, implementation and evaluation of the CEM protocol may contribute to ongoing standardization efforts in ETSI ITS-G5, where the exchange of raw GNSS data could represent a fundamental enabler for next-generation cooperative positioning applications.

5.2 The Server Local Dynamic Map (S-LDM) for Enhanced Collective Perception

The open platforms and novel protocols presented in this thesis can enable a plethora of different innovative V2X use cases, in which vehicles cooperate thanks to the exchange of data through either V2V or V2I/V2N communication.

The aim of this Section it thus to present an innovative service for automated and connected vehicles, which can run on a MEC platform as a MEC service. MEC services can receive ETSI-compliant messages from vehicles, either thanks to the communication with a RSU, via either IEEE 802.11p or C-V2X, which has been evaluated in Chapter 4, or to the transmission of messages via 4G/5G connectivity over the *Uu* interface. The transmission of messages through 5G is gaining more and more interest in the scientific community, especially when considering its combination with messaging protocols such as AMQP 1.0, as mentioned earlier.

Thanks to the technological evolution and to the MEC paradigm, which has been standardized by ETSI, as described in Section 3.7, it becomes now feasible to create and implement centralized services that can acquire up-to-date information about the situation on the road, through low-latency and high-throughput connectivity.

Furthermore, the highest SAE automation levels are hardly achievable when relying solely on V2V communication. Indeed, in decentralized approaches, vehicles are often characterized by a limited overall "view" of the road and by limited on-board computational capabilities. Centralized services are therefore being actively studied, tested and deployed alongside V2V communication, with the aim of overcoming these limitations.

In general, 5G-enabled MEC services for connected and automated vehicles require the reception of data from vehicles, which send standard-compliant messages such as CAMs. These services, however, very often require only a subset of pre-processed data, especially when dealing with centralized maneuver management, which often involves a limited number of vehicles among all the ones circulating in a given stretch of road. This subset of data is related to a "context" on the road, corresponding for instance to information relevant to road users located only in a specific geographic area, or only about vehicles and other non-connected objects involved in a highly automated maneuver. This "context" on the road correspond, in practice, to an up-to-date "map" of a portion of the road, with connected and detected vehicles, road users, and, if needed, other objects.

On this basis, we propose the novel Server Local Dynamic Map (S-LDM) service, based on the Local Dynamic Map (LDM) concept by ETSI [6], [26]. The S-LDM is an open source 5G-enabled MEC service storing a centralized local dynamic map of the road, containing the most up-to-date and historical data of all vehicles (and other non-connected objects detected thanks to sensors) travelling in a given area. This service acts as "middleware" and can very efficiently provide a filtered and processed version of this data to other MEC services, when it detects certain "triggering" conditions on the road (e.g., a vehicle trying to perform a centralized lane merge). Messages are received from vehicles thanks to 5G connectivity and through one or more AMQP 1.0 brokers, and the pre-processed data is stored into a very efficient in-memory database. Then, when a "triggering" condition occurs, the S-LDM can provide the required information with low latency and high reliability, offloading the safety-critical maneuver management services from the burden of

receiving and processing a huge amount of raw data from vehicles.

The S-LDM is designed to run on Linux, and it can be easily containerized for the deployment to MEC platforms based on Docker containers and managed by orchestration frameworks such as Kubernetes. It has also been designed to support high scalability when the number of connected vehicles in a given geographic area increases.

Finally, our service has been developed as part of the 5G-CARMEN project, and it thus supports cross-border operations, which are a focus of the European project.

As detailed in the next Sections, the S-LDM has been deployed on the 5G-CARMEN infrastructure, and extensively evaluated both in a laboratory environment and on-road, by leveraging MEC platforms of actual Mobile Network Operators (MNOs) in Italy, Austria and Germany, and V2X-equipped Maserati vehicles provided by Stellantis.

5.2.1 Review on LDM and existing centralized approaches

The idea behind the Local Dynamic Map (LDM) first came from the SAFESPOT project, and it has been later standardized by ETSI [228]. The LDM, as foreseen by ETSI, is a facility storing information about vehicles and other road users, from which ITS applications can retrieve information on demand. Furthermore, the LDM architecture has been designed to be deployed on each ITS Station (ITS-S), such as vehicles, relying on the different on-board sensors to describe the surrounding environment [26].

Several messages can be used to provide data to a vehicle LDM, such as CAMs and DENMs. However, the perception capabilities of vehicles, together with the data encoded in CAMs and DENMs, may be not sufficient for all the applications. Several studies are thus investigating the usage of collective perception [229]–[231], in which vehicles exchange information about their perceived environment (including the object and road users they detect through their on-board sensors), thanks to CPM messages [69].

The combination of CAMs and CPMs represents a valuable approach for the construction of the LDM by vehicles in a decentralized manner, but the computational burden and the potential channel load required to provide up-to-date information to the LDMs introduce numerous limitations.

With the emergence of the MEC paradigm, thanks to the computing power made available at the edge, it becomes possible to deploy centralized LDMs to overcome the limitations of a purely decentralized approach. Different implementations of LDM services on a MEC architecture can thus be found in the literature. In [232] the authors propose a centralized LDM which stores information received from CAM and DENM messages in a SQL-based database, and processes at the same time the information for the triggering of DENMs in case of possible collisions. A

more modular LDM service is instead presented in [233]. The proposed approach receives the CAM messages from vehicles and stores them for usage in an intersection control application. Both the solutions proposed by [233] and [232] store the entire messages in the database. The work presented in [234], instead, performs a pre-processing of the message information prior to storage, and leverages a JavaScript Object Notation (JSON) text format to serialize CAM and DENM messages, that can then be sent to other interested parties by means of the MQTT protocol.

All the aforementioned approaches rely on SQL-based databases, and do not consider custom databases optimized for storing vehicle data in-memory, which could improve the performance of data management. As mentioned in the next Sections, the S-LDM includes a custom database specifically developed for the purpose of storing vehicle information on a MEC-based LDM. Since the information about objects on the road is changing very rapidly, due to the high dynamism of vehicular networks, this real-time database follows an in-memory approach, and does not store permanent information on the disk (which would significantly slow down each *INSERT* operation). Indeed, CAMs can be sent with a maximum periodicity of 100 ms, and projects such as 5G-CARMEN are investigating the usage of an even higher periodicity (up to 50 ms), for each single vehicle. This can lead, in presence of a high vehicle density, to messages coming with a periodicity lower than 1 ms. According to a database evaluation and comparison between our solution, the SQLite database and WhiteDB (an efficient NoSQL database on shared memory) [235], which is not reported here for the sake of brevity, writing on disk with a Write-Ahead Log (WAL) mechanism can cause up to a *57825x* INSERT delay increase for a mechanical hard drive (increasing the INSERT time from the 0.48 microseconds of our solution to around 27 milliseconds of SQLite with WAL), and up to a *3705x* increase for an SSD. We thus decided to opt for an in-memory solution.

Furthermore, the implementation proposed in [233] is able to detect possible relevant events on the road, but, in contrast to our service, the LDM itself executes the application entirely and does not take into consideration support for other MEC services.

Finally, to the best of our knowledge, the ability to store data about non-connected objects on a MEC service has not yet been investigated, since prior literature only focused on messages related to connected vehicles.

5.2.2 Service description and architecture

The S-LDM is able to receive messages from a large number of vehicles, and use the data encoded in each message to populate an enriched and pre-processed map of the road, containing both historical and real-time data of vehicles and other non-connected objects. As a fundamental requirement, this map should be able

to promptly deliver data to other MEC services as well as to vehicles, as needed, by timely receiving ETSI-compliant messages from the vehicles themselves. Other than providing a much wider and more precise view of the situation on the road than a decentralized architecture (relying only, for instance, on communication via IEEE 802.11p or LTE-V2X/NR-V2X Mode 4/Mode 2), the S-LDM represents a novelty over standard V2N approaches, which envision the direct transmission of data from vehicles to MEC services. Indeed, as mentioned, the S-LDM can be considered as a "middleware", which processes the data from a large amount of vehicles, stores it in a very efficient database, and provide a filtered and processed version of this information to other safety-critical MEC services, when needed. It should be stressed how this proves to be very useful, as it offloads both vehicles and other services from the necessity of quickly processing a large number of messages, which could reduce the performance of the actual latency-critical algorithms (e.g., control algorithms for centralized automated lane merge).

The S-LDM has been designed as part of the 5G-CARMEN project architecture, which is based on the one standardized in the C-Roads project. Indeed, the S-LDM is able to receive messages from vehicles through AMQP 1.0 and the deployment of one or more AMQP broker. These messages include CAMs, and, as soon as they are fully standardized, CPMs, since the combination of the two appears to be one of the best choice for populating the LDM database, as mentioned in the previous Section. The S-LDM has been developed and tested by taking as reference the Apache ActiveMQ broker, which is the broker of choice in 5G-CARMEN, but it can potentially work with any broker compliant with the latest AMQP 1.0 specifications. In the 5G-CARMEN architecture (see also Figure 2.9, by replacing "MEC service 1" with the S-LDM), the brokers are deployed on the respective MEC platforms of the MNOs participating the project, guaranteeing low latency information provision to the S-LDM.

The internal architecture of the S-LDM is depicted in Figure 5.11.

As can be seen, the S-LDM is able to receive messages from one or more AMQP broker, thanks to the subscription to one or more topics. For this purpose the S-LDM integrates the Apache Qpid Proton library, which is able to efficiently and quickly handle the reception of messages from a broker compliant with AMQP 1.0. Each message is then decoded by a custom ETSI ITS-G5 stack, developed to be as much efficient as possible with the goal of enabling very high update rates of the internal database. As mentioned before, the S-LDM can receive information about both connected and non-connected objects that are occupying the carriageway. Information about connected object can be received thanks to CAM messages. In particular, the S-LDM has been designed to receive CAMs from a large number of vehicles, each sending CAMs at 20 Hz. Even though the maximum frequency foreseen by ETSI, as mentioned multiple times in this thesis, is 10 Hz, the 5G-CARMEN project is experimenting the transmission at 20 Hz, to enable fine-grained

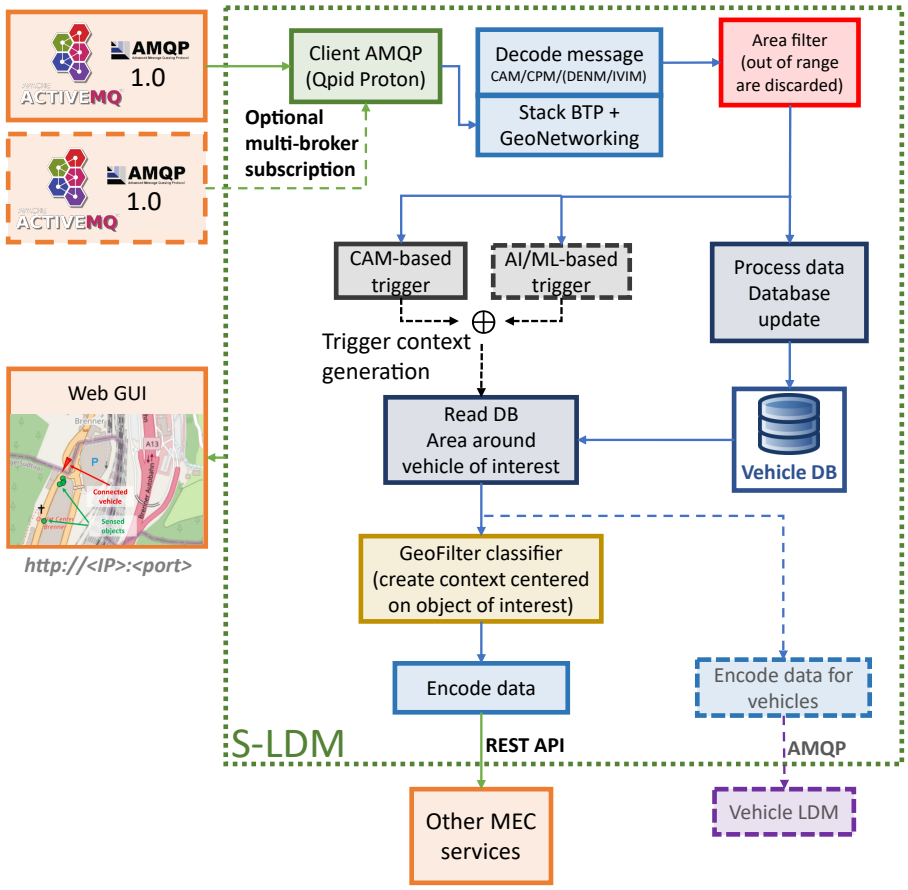


Figure 5.11: S-LDM service internal architecture and interfaces to other components and services.

updates for highly automated maneuver management. Concerning instead the non-connected objects, the S-LDM is expected to leverage the CPMs, as soon as they are fully standardized. However, since the standardization process is still in progress, we developed the so-called *Virtual CAMs*, which carry information about detected objects and are sent by connected vehicles. These special messages have been chosen as a preliminary implementation before CPMs are standardized. Indeed, remote objects can be detected by vehicle sensor, and this information can be provided to the V2X OBU through the vehicle CAN bus. The OBU can then encode this data in Virtual CAMs which are used for tracking remote objects. Instead of sending a list of objects as would be required by CPMs, the absolute position of the detected objects is calculated based on sensed data and encoded as a standard CAM. These fabricated Virtual CAMs are sent to the AMQP broker to which the S-LDM is subscribed, by connected vehicles. Furthermore, the S-LDM is also able to decode other types of messages, such ad DENMs and IVIMs, which can be possibly exploited to further enrich the centralized LDM.

The received information is then filtered (more details on the *Area filter* module will be provided later), processed and used to update a custom, highly-efficient, in-memory, thread-safe database, storing the content of the centralized local dynamic map. According to a database evaluation performed on a Desktop PC equipped with an Intel Core i5-10600k CPU, our database is able to perform an *INSERT* operation in less than one microsecond, when no other parallel operations are pending on the same database.

As mentioned earlier, one of the most important features of the S-LDM is its capability to detect the occurrence of "triggering conditions", i.e., either a vehicle performing some action to signal that a maneuver with a high level of automation has been requested, or a situation on the road which potentially requires the intervention of a centralized MEC service. When the S-LDM detects one of these conditions, it efficiently reads the database and computes a context around a reference vehicle or other object. This context includes all the information on vehicles and other non-connected objects within a certain configurable radius around the reference node. The triggering condition detection is currently performed in a deterministic way, by analyzing a set of fields in the received messages (mainly, CAMs). For instance, the intention of performing a lane marge is detected when a vehicle switches on one of its turn indicators, encoding this information in the *Exterior Lights* field of the CAMs it sends to the S-LDM. It is also planned to include an AI/ML-based trigger detection system, which is currently under development.

After the creation of the context, it is made available to other MEC services through a dedicated REST API, which encodes the pre-processed and filtered data into an easily manageable JSON format.

Optionally, the S-LDM is also able to leverage the content of the database to periodically update the information available to each vehicle with wide and precise centralized information related to objects on the road. This operation occurs through the generation of proper update messages and to the transmission of such information to an AMQP broker, to which vehicles can subscribe to receive the periodic updates. Even though it has not been fully implemented yet, it will be integrated in the next version of the service.

Finally, the S-LDM provides a web-based interface showing the content of the local dynamic map in real-time, with a configurable update rate (the default value is 500 ms, not to overload the database with continuous read operations while it is also being updated by vehicles). This interface lets both end-users and road operators monitor the situation on the road in real-time and with highly-accurate data.

With the aim of tackling scalability and supporting cross-border use cases, each S-LDM instance has been specifically thought to cover a limited portion of the road, and filter out all the messages of vehicles coming from outside this area, thanks to the *Area Filter* module. In order to extensively cover the road, several S-LDM instances should be created, each covering an area starting from where the

previous S-LDMs areas end. The overall areas covered by neighboring S-LDMs should also superimpose, in order to let each instance store a complete view of the road, even when the context generation involves a vehicle travelling near the border of the coverage area. A 2D representation of this mechanism is shown in Figure 5.12.

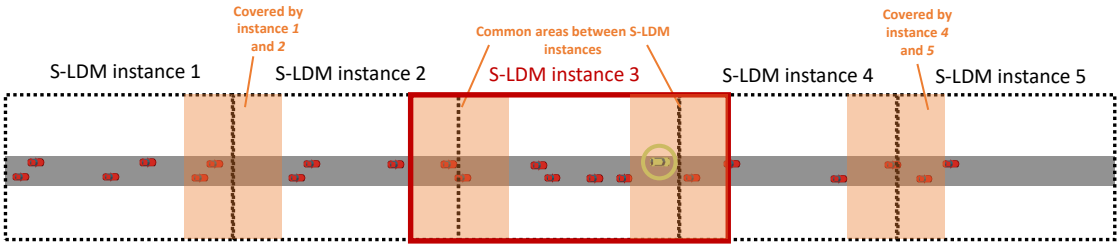


Figure 5.12: Multiple S-LDM architecture for cross-border and in-country scalability.

As can be seen, thanks to the superimposing areas, some stretches of road are covered by more than one S-LDM instance. This enables a proper context generation even when a vehicle, like the highlighted car, is travelling near the coverage area border, and needs to be informed about all nearby road users, including the preceding red car already travelling inside the *S-LDM instance 4* area.

Receiving all messages from all vehicles on the road, and then filtering them inside the *Area Filter* module (which requires decoding each single message) could be a computationally expensive operation, especially under high traffic conditions. Therefore, each AMQP client of each S-LDM instance is set to receive only the messages of vehicles travelling in an area slightly larger than the actual coverage area, limiting the number of messages discarded at the Area Filter module and improving the performance of the S-LDM itself. The message pre-filtering happens at an AMQP broker level thanks to the Quadkeys, as described in Section 2.4.1.

The S-LDM has been released under an open-source license, and it is available on GitHub [27].

5.2.3 In-lab pre-deployment evaluation

Before the deployment as a MEC service on the 5G-CARMEN architecture, a series of in-lab tests have been performed on a local instance of the S-LDM. This testing campaign was mainly aimed at evaluating the performance of the S-LDM, connected to a local instance of an ActiveMQ broker, on well-known hardware running Ubuntu 20.04 LTS.

During the development and in-lab pre-deployment tests, we resorted on the emulation capabilities of ms-van3t. It should be recalled that the messages coming from emulated vehicles can be either directly broadcasted, or transmitted to an

external server via UDP. We relied on this last possibility, combined with a custom AMQP client relaying the UDP messages to ActiveMQ, to emulate the presence of vehicles sending their messages to an AMQP broker. Concerning the emulated scenario, we configured ms-van3t to emulate vehicles travelling on a stretch of the A22 motorway, between Trento Nord and San Michele all'Adige, on which the 5G-CARMEN project focused for most of the in-country pilots. It is worth highlighting that, since the S-LDM receives the messages directly from the AMQP broker, its behaviour will be exactly the same, no matter whether these messages originate from a simulated scenario or from real vehicles.

The testbed setup for the in-lab tests is depicted in Figure 5.13.

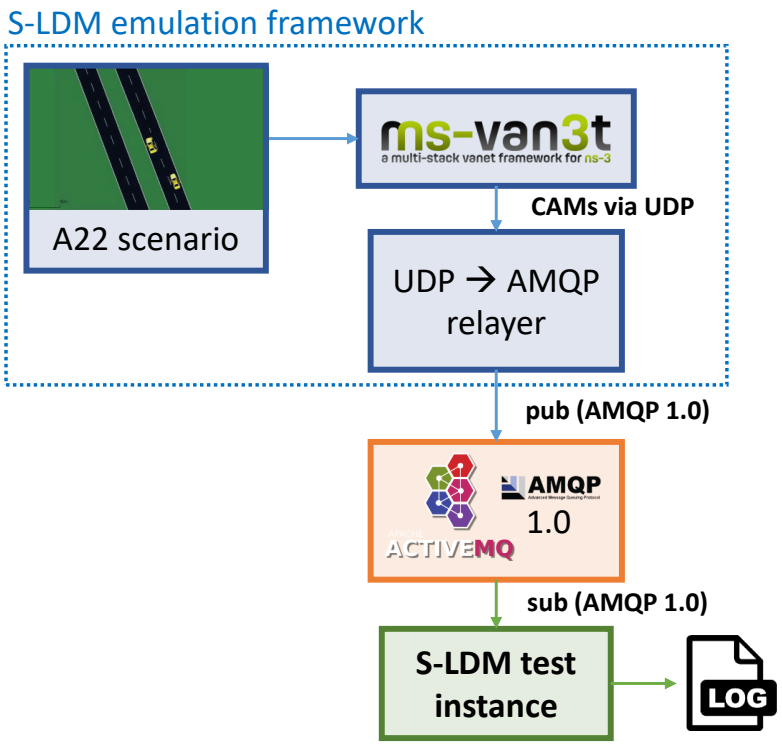


Figure 5.13: Testbed setup for the laboratory pre-deployment performance tests.

The main aim of the in-lab tests was to assess the S-LDM performance, in terms of needed average computation time, focusing on three main sub-modules (with reference to Figure 5.11): *(i)* Message Decoder, *(ii)* Area Filter and *(iii)* Database Update.

All in-lab measurements have been performed by setting a local instance of the S-LDM to cover an area around the chosen stretch of road. It should be noted that the number of vehicles in simulation is limited by the hardware used for the tests. Indeed, a non-negligible single-core computational effort is required by ms-van3t, when the number of vehicles increases beyond a certain amount. The device used

for the tests, running an Intel i7-10750H CPU, was able to properly emulate up to 25 vehicles without slowing down the S-LDM or any other process running on it. This justifies why we chose to test up to 25 vehicles, while more extensive scalability tests are currently ongoing.

The most relevant results are reported in Figure 5.14, together with the main ms-van3t emulation parameters. The plot shows the average computation time, in microseconds, for each of the considered sub-modules.

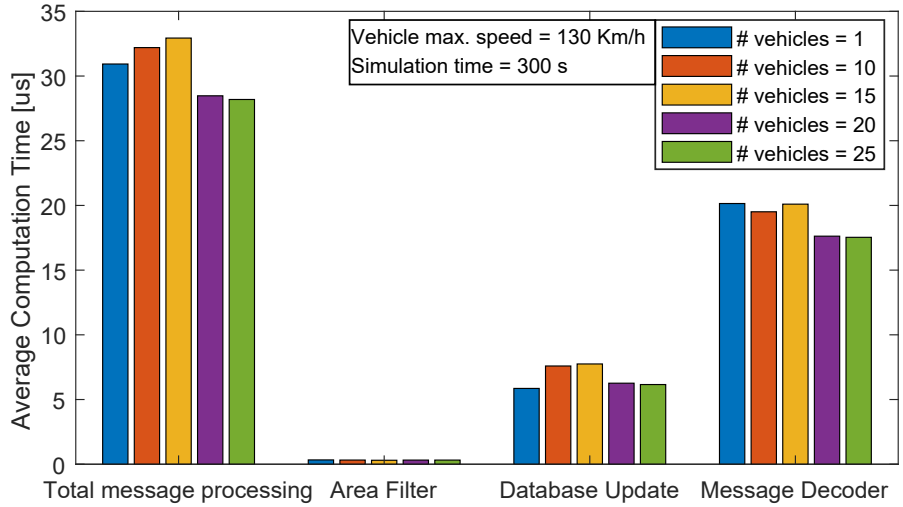


Figure 5.14: Results of the S-LDM pre-deployment sub-modules performance tests.

As can be seen, the S-LDM performance remains stable up to at least 25 vehicles, with few very small variations between the different numbers of vehicles. These oscillations are nevertheless very limited (up to a maximum of 3 microseconds) and can be ascribed to small kernel scheduling time oscillations. This is a good indication on how the S-LDM can scale well, at least when taking into account a limited number of subscribers.

Furthermore, the results show how the most demanding sub-module is the Message Decoder, further justifying the Quadkey-based filtering approach described earlier.

Finally, all the measured times are much lower than 10 ms, which is considered as an upper bound to the end-to-end latency in the 5G-CARMEN project, as mentioned in Section 2.4. Indeed, the measured values reveal how few tens of microseconds are needed to fully process each message. This is also noticeably lower that the latency contribution introduced by other components in the 5G-CARMEN architecture, demonstrating how the S-LDM can be an efficient low-latency 5G enabler for high levels of automation.

A second set of laboratory tests was also aimed at determining the time needed to transfer the context data to other MEC services, through the dedicated REST API. Six tests have been performed, each under the same configuration, by transmitting

the context data to Python Tornado sample server, acting as MEC service receiving the information from the S-LDM. 100 POST requests have been performed for each test, with a periodicity of 1 second. The results are reported in Table 5.5.

TEST #1	TEST #2	TEST #3	TEST #4	TEST #5	TEST #6
latency (avg) [ms]	latency (avg) [ms]	latency (avg) [ms]	latency (avg) [ms]	latency (avg) [ms]	latency (avg) [ms]
1.006	0.975	1.029	0.987	0.982	0.991
latency (max) [ms]	latency (max) [ms]	latency (max) [ms]	latency (max) [ms]	latency (max) [ms]	latency (max) [ms]
2.158	2.208	2.121	2.167	2.245	2.329
latency (RTT - avg) [ms]	latency (RTT - avg) [ms]	latency (RTT - avg) [ms]	latency (RTT - avg) [ms]	latency (RTT - avg) [ms]	latency (RTT - avg) [ms]
2.384	2.330	2.341	2.214	2.215	2.256
Excluding the first request					
latency (max) [ms]	latency (max) [ms]	latency (max) [ms]	latency (max) [ms]	latency (max) [ms]	latency (max) [ms]
1.544	1.199	1.539	1.219	1.135	2.329

Table 5.5: Results of the S-LDM REST API one-way and RTT latency measurements.

The Table reports, from top to bottom, and for each test, the average and maximum one-way latency, and the average RTT (i.e., from the POST request to the reception of the reply from the other MEC service). The last section of the Table reports instead the maximum one-way REST API latency after the exclusion of the first POST request, which usually requires more time due to the need of establishing the connection.

As can be seen, the results show that the average one-way delay is slightly less than 1 ms, with few peaks up to around 2 ms. This additional delay, when sending context data to other MEC services, can be considered completely acceptable while ensuring, at the same time: *(i)* the target of keeping the full-chain latency under 10 ms and *(ii)* a standard, interoperable, REST interface between containerized components in the 5G-CARMEN MEC platforms.

5.2.4 In-country on-road evaluation

After the first in-lab measurements, the S-LDM component has been integrated as a container inside the MEC platforms of three network operators participating in the 5G-CARMEN project (i.e., TIM of Italy, Magenta Telekom of Austria and Deutsche Telekom AG of Germany). These MEC platforms run a Kubernetes-based orchestrated edge platform, developed by 5G-CARMEN [236], and designed to run and orchestrate 5G-enabled applications and services for cooperative and automated driving. Is it worth mentioning how the S-LDM can be easily containerized and configured for deployment thanks to a dedicated Dockerfile available in the GitHub repository [27].

The deployment of our service on actual MEC platforms allowed us to begin a number of road test campaigns, letting us validate the S-LDM capabilities in real-world scenarios. During each test, data about connected and non-connected objects was continuously received via AMQP 1.0, thanks to a 5G NSA *Uu* link. Furthermore, all measurements have been performed with two Stellantis vehicles equipped with V2X OBUs. The OBUs installed on the connected and automated vehicles are based on L3 Pilot [237] Maserati prototypes, upgraded in the 5G-CARMEN project with a perception system for High Automated Driving (HAD). These V2X OBUs enable short range communication via *PC5* and *Uu*. Specifically, the *Uu* network connectivity is guaranteed by a 5G New Radio (NR) modem to ensure the needed data rate performance and benefit from the 5G-CARMEN cross-border features. The connectivity towards the S-LDM has been designed such that: *(i)* the data transfer matches the on-board detection frequency (i.e., there is no bottleneck effect due to the transmission of data via V2X), at a higher rate than what is normally foreseen by the standards for V2V communication [72], *(ii)* it guarantees as much as possible service continuity when crossing borders as well as *(iii)* the shortest path are always used to reach the MEC services, i.e., the S-LDM. Concerning the application layer deployed on the OBUs, a dedicated AMQP client is used for the exchange of V2N packets (i.e., CAMs and virtual CAMs) with the AMQP broker. This client is also populating the header of each AMQP message with all the required C-Roads properties, as described in Section 2.4.1.

The primary focus of this Section are the results of the most significant test involving the MEC platform managed by TIM in Italy. The next Section, related to cross-border tests, involves instead both the edge platform in Italy and the one managed by Magenta Telekom (MTA) in Austria. It is also to be noted that the S-LDM deployed in Germany has been used for several additional test campaigns, which yielded similar results as the ones presented here.

The two main aims of the road tests can be resumed as follows:

1. Evaluate the S-LDM capability of receiving CAMs (and virtual CAMs) at a very high rate, i.e., 20 Hz, through AMQP 1.0 and with the support of a 5G network

2. Evaluate the performance of the S-LDM when dealing with real-world connected vehicles and other non-connected objects.

Figure 5.15 reports the CDF of the overall messaging processing times, for each message received by the S-LDM during the whole duration of a test performed on the Italian side of the E45 motorway (i.e., the A22 motorway). This test has been performed between Sterzing-Vipiteno and Brennerpass, and the reported values are related to the two Maserati Prototypes (herein labelled as “Vehicle 1” and “Vehicle 2”).

As can be seen, most message decoding and database update operations occurred in less than 50 microseconds, with very few spikes up to around 0.5-0.7 ms

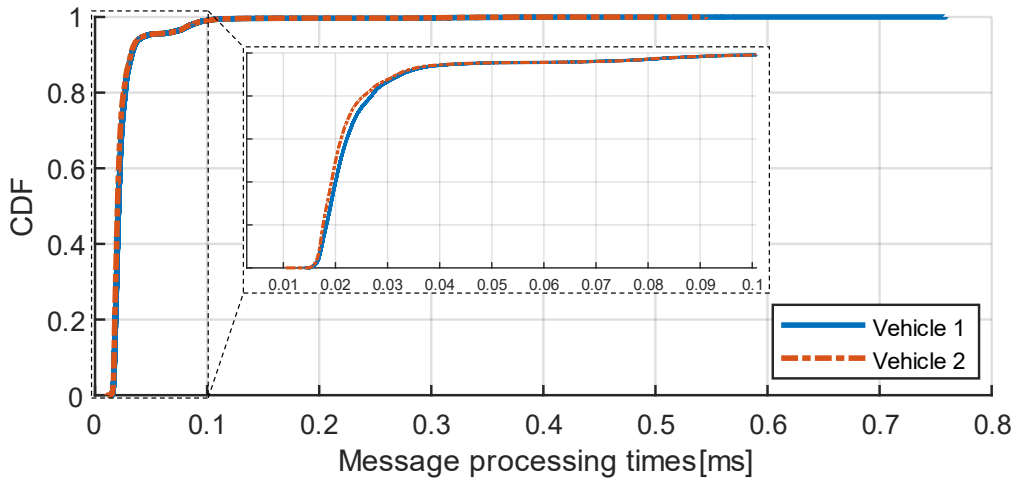


Figure 5.15: Empirical CDF of the S-LDM message processing times during the whole road test duration.

(these peaks, however, always involve a single message over thousands of processed messages). The outcome, in terms of performance, is thus good. Furthermore, despite the numbers are slightly different (due to the different resources assigned to the S-LDM process), the orders of magnitude are consistent with pre-deployment results, as the great majority of messages is decoded and processed in a few tens of microseconds.

It was also possible to measure how much time occurs between two consecutive database updates for the same connected vehicle or detected object. This metric (which we named "*Instantaneous Update Rate*") was gathered to understand how often the database was actually being updated, considering the processing times and network conditions on the A22 motorway. It is indeed crucial to be able to update the local dynamic map as quickly and promptly as possible, since it provides data to other 5G-enabled latency-critical services. The results are plotted in Figure 5.16.

As can be seen, the database is being updated on average every 50 ms, which is consistent with the high-frequency CAM generation at 20 Hz. It can then be noted that few spikes were observed, with a very low probability. These spikes are likely due to the jitter and delay caused by the underlying network architecture, as no significant performance issues were detected throughout the whole test duration, as reported in Figure 5.15. They are nevertheless occasional and affect single packets, thus they should not cause any noticeable issue in the services relying on the S-LDM to gather an up-to-date map of the road.

By examining the CDF, it is also possible to determine how, in more than 95% of cases, the database can be updated in under 70 ms after the previous update. As this periodicity is highly affected by the jitter of the underlying network architecture, which is based on 5G Non-Standalone, this result can be considered

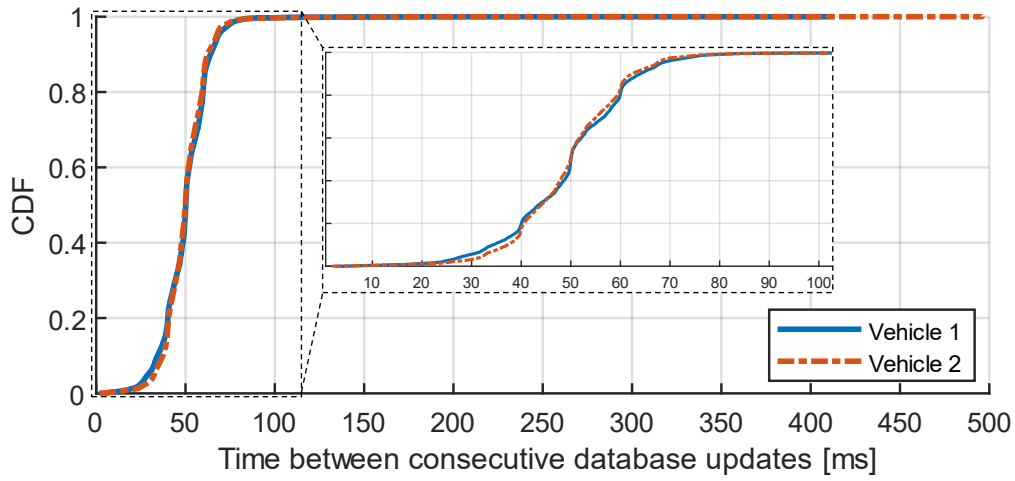


Figure 5.16: Empirical CDF of the time between consecutive database updates for each vehicle during the whole road test duration.

quite promising. This is especially true when considering that, according to ETSI, the standard highest CAM transmission frequency is set to 10 Hz [63]. According to the tests presented here, a network-intensive transmission of messages at 20 Hz is viable and can lead to tangible advantages, enabling the creation of an overall map of the road with update rates higher than before.

5.2.5 Cross-border on-road evaluation

The in-country road tests paved the way for a number of cross-border measurements, among which the most significant are reported here.

The cross-border tests involved the same Maserati vehicles as before, travelling on different stretches of road between different countries. This Section focuses on the results while travelling along the Brennerpass border, on the E45/A22 motorway between Austria and Italy.

Two S-LDM instances were deployed respectively in the MEC platforms managed by TIM in Italy and MTA in Austria. The same instances were then configured as subscribers to two AMQP 1.0 brokers, one deployed on the Italian MEC platform, and one on the Austrian MEC platform. Both S-LDM instances were set to cover superimposing areas around the border, such that the whole border area was well-covered by both S-LDMs.

The S-LDM can handle cross-border scenarios thanks to the subscription to multiple AMQP brokers, which, in the 5G-CARMEN architecture, is in turn enabled by a dedicated inter-MEC communication infrastructure. Indeed, in this case, each S-LDM instance was connected to both the Italian and the Austrian broker, to receive messages from vehicles located in the two countries, enabling the creation

of a full context around the border line.

The main aim of the cross-border tests was *(i)* to evaluate the performance of the S-LDM in cross-border scenarios and when deployed to different MEC platforms, and *(ii)* to evaluate the disconnection time due to the MNO network reselection time at the border, from the S-LDM point of view. It should be mentioned that a fast reselection mechanism was developed and deployed by 5G-CARMEN, to enable low reselection delays when switching between operators (e.g., between MTA and TIM).

The tests reported here have been performed on the E45 stretch of road between Nößlach (Austria) and Brennerpass (Italy), focusing on the Austria to Italy direction. The path followed by one of the Maserati vehicles is depicted in Figure 5.17. The map takes as reference the S-LDM instance in Italy, and shows, in blue, the positions recorded through messages received from the Italian broker, and, in red, the ones gathered thanks to the CAM messages received from the Austrian broker.

This plot shows how the S-LDM can be used, as an additional feature, to understand under which network a vehicle is connected to, given its geographical position. Indeed, thanks to the multi-broker subscription, it is possible to distinguish which messages are coming from which broker (and, consequently, which messages are coming from which network).

Finally, before presenting the results, it should be mentioned that the S-LDM includes the so-called *ageCheck* feature. This feature, as part of the message decoding sub-module, leverages the GeoNetworking timestamp to check the age of the data stored inside the database before updating it with the new received data. This allows our service to discard potential outdated messages received after crossing the border, due to out-of-order packets.

The overall outcome of the tests was positive, since our service was able to receive real-time information to track connected vehicles and detected objects on both sides of the border, by processing the messages from the respective AMQP brokers. Even though several runs have been performed, only one of them is reported (being the most significant) in order to make a more descriptive outline of the results.

Figure 5.18 shows the normalized message delay as a function of the time since the beginning of the test, taking as reference the S-LDM instance running on the TIM MEC platform. The normalized message delay represents a measurement of the delay between message transmission from vehicles and reception by the S-LDM, minus the minimum observed during the whole test. If d_i is the delay measured for packet i, then the normalized message delay for the same packet is computed as:

$$\hat{d}_i = d_i - \min_i d_i \tag{5.1}$$

The minimum normalized delay thus corresponds to a value of "0". Using a normalized delay value was necessary as the time synchronization between the MEC platforms of the different MNOs was not guaranteed, thus not ensuring consistent measurements in case of absolute delay values.

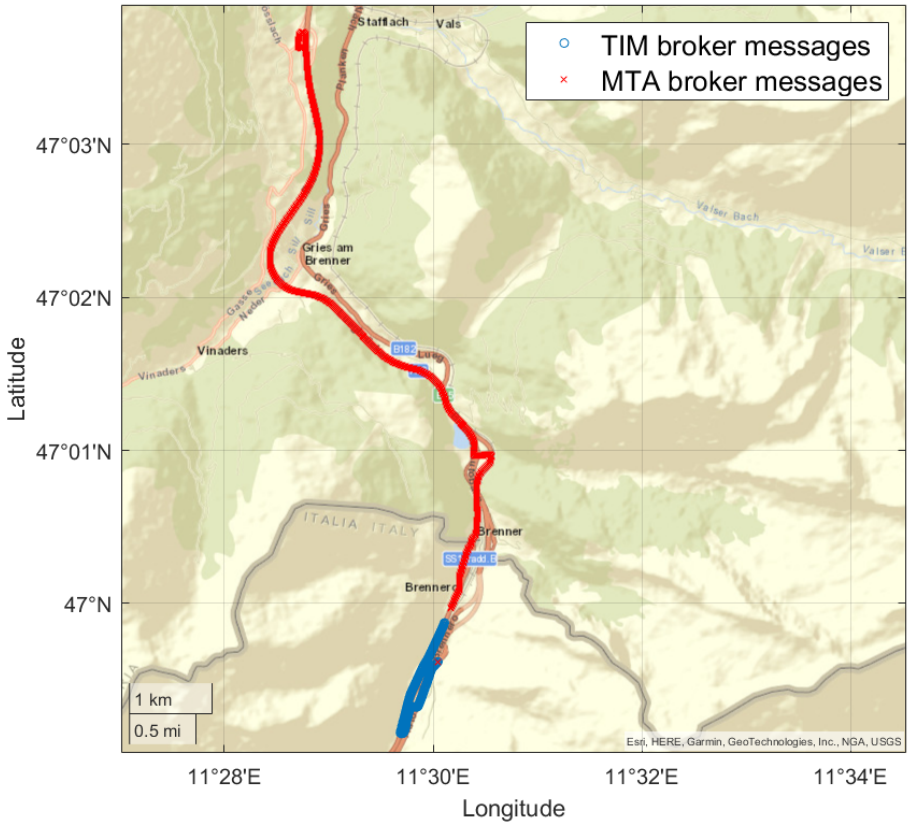


Figure 5.17: Path followed by one of the two Stellantis vehicles during the cross-border road tests from Austria to Italy. The messages received by the S-LDM from the TIM network and AMQP broker are depicted in blue, while the messages from the MTA network and broker are represented by red points.

As can be seen, the normalized delay remains mostly stable during the whole test duration, with very few peaks, whose duration in time is always relatively short and under a second. These spikes are probably due to the jitter and delay caused by the underlying 5G NSA network, and are expected to be noticeably reduced when moving to a full-fledged 5G SA architecture. Furthermore, it is possible to observe how the average normalized delay for the MTA broker is, overall, slightly higher. This is expected, as the data shown in Figure 5.18 is taken from the S-LDM instance in Italy, with respect to which all messages from the MTA broker in Austria experience an additional inter-MEC communication delay.

Finally, it was possible to observe a disconnection interval of only 13 seconds, when moving from Italy to Austria, which remained quite stable for all the test runs along the border. Indeed, focusing on the direction from Austria to Italy, we

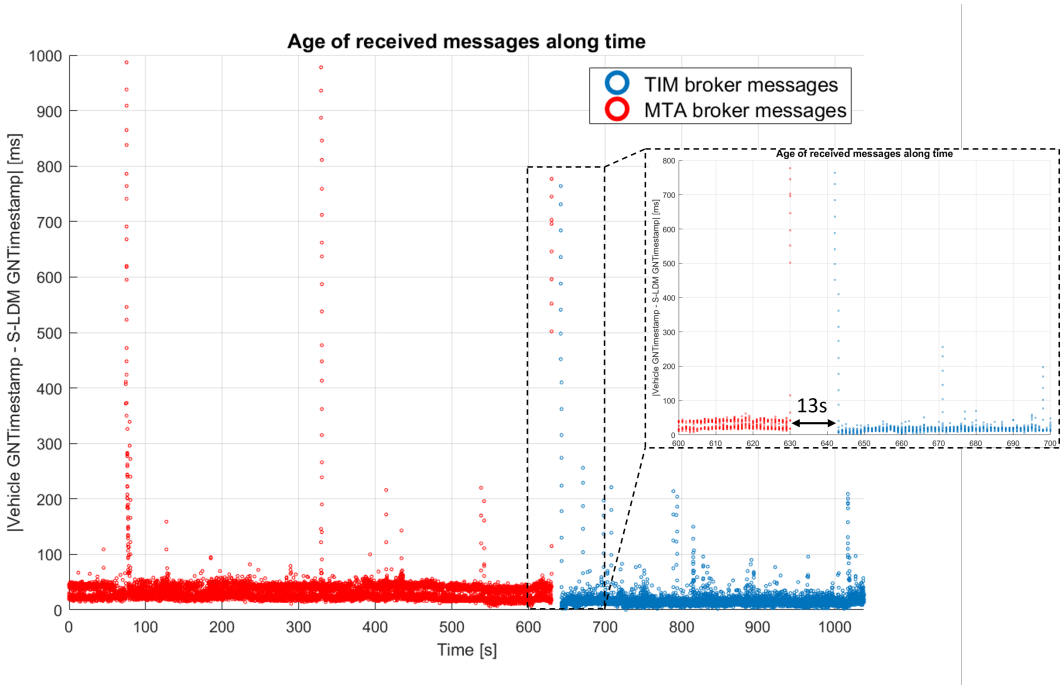


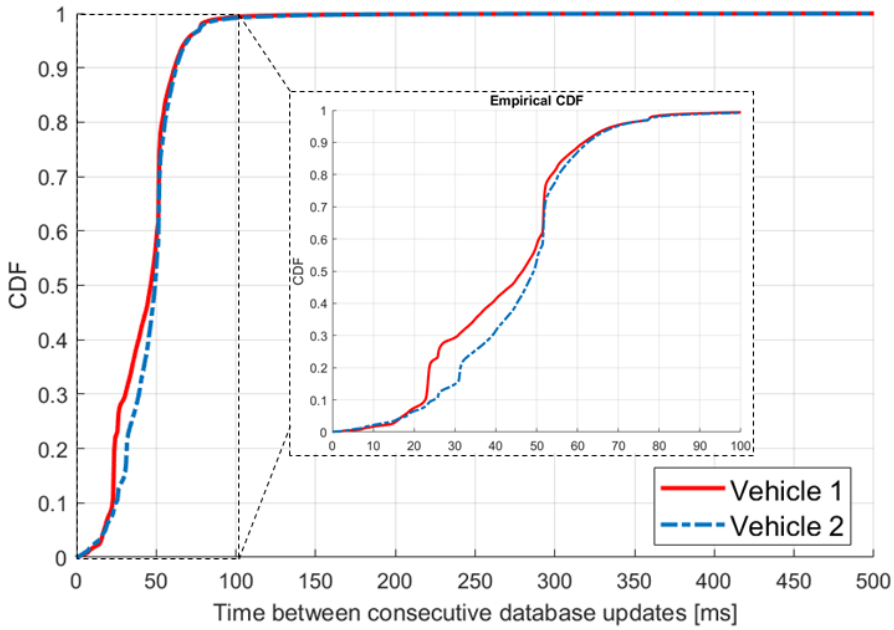
Figure 5.18: Normalized message delay (i.e., a measurement of the delay between message transmission from vehicles and reception by the S-LDM, normalized to the minimum observed during the whole test) along the duration of a single test run, for a single vehicle. The S-LDM instance running on the TIM MEC platform has been taken as reference.

measured an S-LDM average disconnection time, due to fast network reselection, of 13.3729 s, with a standard deviation of 2.0203 s, and a 95% confidence interval of $\pm 0.7062\,s$.

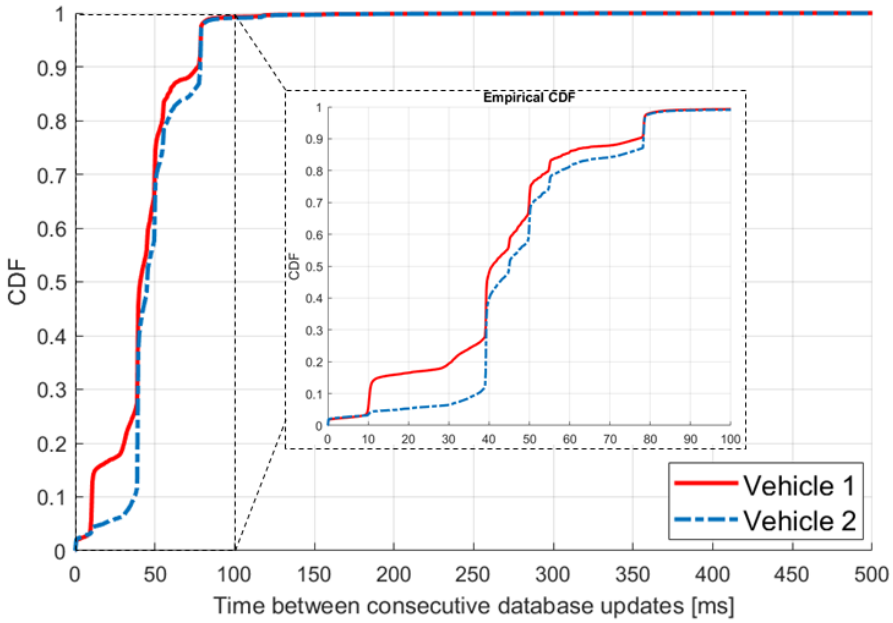
The CDFs of the *instantaneous update rate* of the S-LDM, concerning both the TIM and MTA instances and during the most significant Austria-to-Italy cross-border test, are reported in Figure 5.19.

As can be seen, and in line with the in-country tests, it is possible to observe how around 80% of the times, for a given vehicle, the database is updated with a periodicity of 50 ms or less (lower values are possible due to the underlying network jitter). As in the previous case, this is in accordance with the 20 Hz CAM frequency. Very similar results could be observed concerning both the S-LDM instance deployed to the TIM MEC platform, and the one deployed to the MTA MEC platform, proving how the S-LDM can efficiently store and provide a real-time map of the road in cross-border scenarios.

Finally, it was possible to evaluate the overall message processing times during the whole duration of the same cross-border test. The CDFs related to both S-LDM instances are depicted in Figure 5.20.

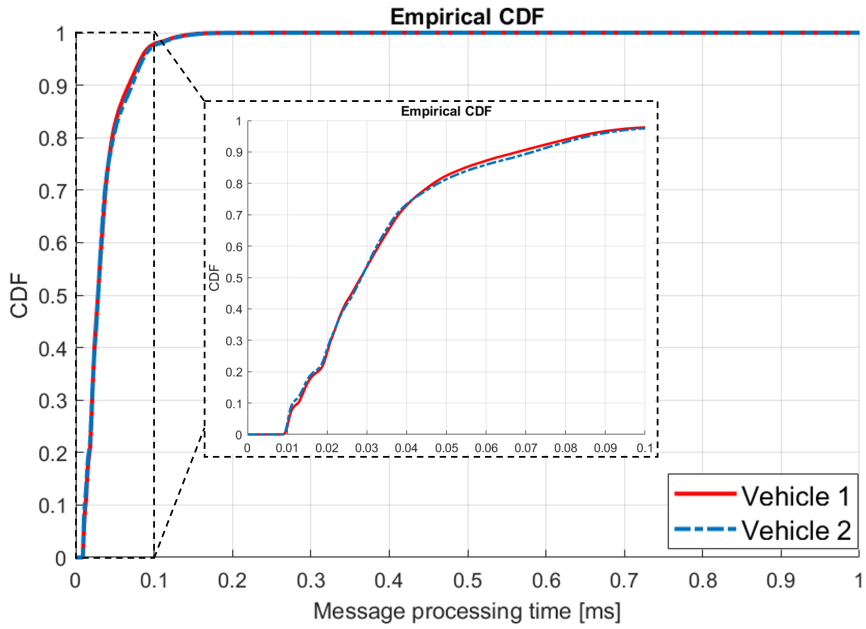


(a) TIM MEC platform S-LDM instance.

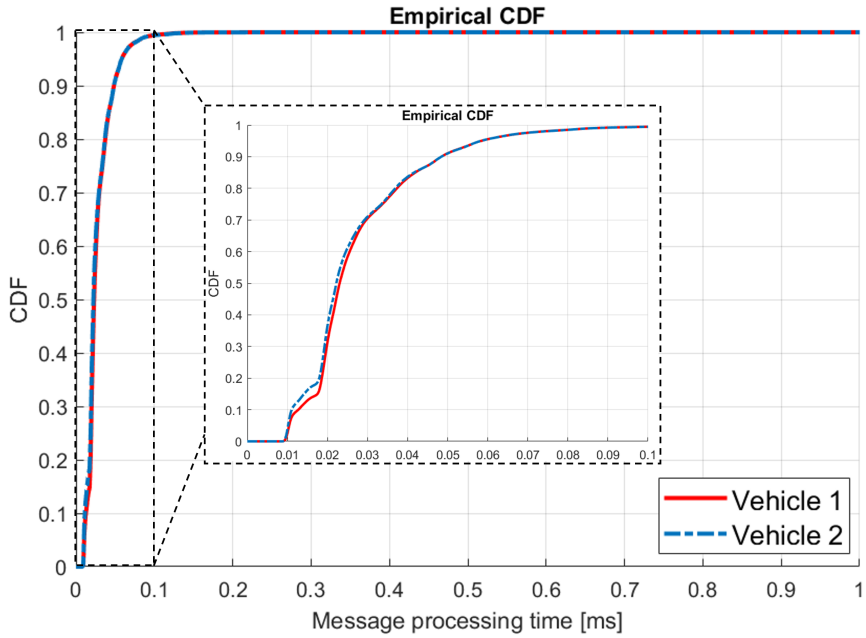


(b) MTA MEC platform S-LDM instance.

Figure 5.19: Cumulative Distribution Function of the time between consecutive database updates for each of the Maserati vehicles during the most significant Austria-to-Italy cross-border test.



(a) TIM MEC platform S-LDM instance.



(b) MTA MEC platform S-LDM instance.

Figure 5.20: Cumulative Distribution Function of the message processing times for each of the Maserati vehicles during the most significant Austria-to-Italy cross-border test.

As in the previous case, the outcome is good, as 90% of messages are processed in under 72 microseconds, concerning the instance at TIM MEC, and 50 microseconds concerning the instance at MTA MEC. The difference between the two, given the microsecond-scale of all values, can be explained with the different hardware configuration supporting the two MEC platforms. Furthermore, the average processing time proved to be around 35 microseconds for the S-LDM deployed in Italy, and 28 microseconds for the one instantiated in Austria. These values are consistent with the in-lab and in-country road tests, and show how the S-LDM can perform very well under different circumstances, including cross-border scenarios.

Furthermore, all the results presented in this Section, and in the previous one, allowed us to validate our component in the field, showing its effectiveness in processing messages with very low latency, storing a highly efficient map of the road and providing a filtered and processed context to other MEC services when needed.

5.2.6 Scalability and future directions

As mentioned earlier, scalability represents one of the major challenges for most V2X MEC services, especially when the connected vehicle density grows high. As each S-LDM instance, to tackle scalability, is designed to cover a limited portion of the road, multiple instances are expected to be deployed in future practical scenarios covering a whole motorway or urban road, possibly adapting the coverage areas in real-time, depending on the number of active message producers (i.e., vehicles) and on the resource usage.

An extensive scalability study of the S-LDM is thus currently ongoing, considering both different vehicle densities and the CPU/RAM resource usage. Preliminary results show that a single instance of the S-LDM can handle up to at least 550 vehicles without any noticeable performance loss, when CAMs are sent at 20 Hz and with the current AMQP broker settings. In a two-lane per direction of travel motorway with vehicles moving at 130 km/h and spaced 36 m apart, a single S-LDM instance could therefore cover at least, roughly, 4.9 km of road.

5.3 Publications

This Section reports our publications, related to the topics presented in this Chapter.

5.3.1 Conferences

S-LDM

- F. Raviglione, C. M. Risma Carletti, C. Casetti, F. Stoffella, G. M. Yilma and F. Visintainer, “S-LDM: Server Local Dynamic Map for Vehicular Enhanced

Collective Perception", IEEE VTC2022-Spring, Helsinki, Finland, June 2022, pp. 1-5 [6]

The CEM protocol

- A. Minetto, S. Zocca, F. Raviglione, M. Malinverno, C. Casetti, C. F. Chiasserini and F. Dovis, "Cooperative Localization Enhancement through GNSS Raw Data in Vehicular Networks", 2021 IEEE Globecom Workshops, Madrid, Spain (held remotely), December 2021, pp. 1-6 [34]

5.3.2 Journals

The CEM protocol

- F. Raviglione, S. Zocca, A. Minetto, M. Malinverno, C. Casetti, C. F. Chiasserini and F. Dovis, "From Collaborative Awareness to Collaborative Information Enhancement in Vehicular Networks", Elsevier Vehicular Communications, vol. 36, June 2022 [7]

Chapter 6

Conclusions

The automotive field is experiencing an incredible evolution in the past decade, with the emergence of the first partially autonomous vehicles and the definition of the first standards for connected cars.

This evolution has pushed both academia and industry to investigate several innovative use cases for automated and connected vehicles, towards progressively high levels of automation. These applications aim at making future transportation systems smarter, safer and greener. However, the most advanced applications, such as Collision Avoidance, would he hardly feasible when relying solely on on-board sensors and ADAS system.

Vehicular networks have been thus proposed as a solution to make vehicles exchange data between them, and between them and the infrastructure, to provide applications such as centralized maneuver management, Virtual Traffic Lights, GLOSA, platooning, See Through and many more. This thesis has thus analyzed, in Chapter 2 the main use cases for vehicular networks, providing details on their implementation and describing the main technologies and protocols for connected vehicles. These include DSRC-based technologies, i.e., IEEE 802.11p, and cellular-based technologies, such as LTE-V2X Mode 4 or NR-V2X Mode 2.

When developing, deploying and testing a V2X application, open platforms play a fundamental role, for both the research and industrial communities. Indeed, they enable the reproducibility of results, fostering further research, they can be easily customized and upgraded for testing new protocols and applications, and they provide a low-cost and very effective way of working with vehicular networks. Open platforms provide transparency and reliability, thanks to the architecture and code being public and open to contributions by users. They also significantly reduce the risk of vendor lock-in, i.e., being locked in by a specific technology or device manufacturer, which may stop selling its products or providing support to its commercial libraries.

As there is currently a scarcity of open source solutions for connected vehicles, this thesis has focused on open frameworks, applications and services for vehicular

networks, looking at the future directions and providing novel insights on different technologies thanks to both simulations and field tests.

Three main wide areas can be defined when developing, deploying and testing an open V2X platform. These areas have been analyzed through three different Chapters in this thesis. Chapter 2 tackles the development of open source platforms able to provide vehicular connectivity and enable V2X use cases and applications, based on standardized (or promising) access technologies and networking stacks. First, it presents and evaluate an open DSRC platform based on low-cost, customizable, PC Engines APU embedded boards, and on a special version of the OpenWrt Linux distribution. Then, it presents a lower-layer agnostic latency measurement protocol, called Latency Measurement Protocol (LaMP), together with the first LaMP-compliant tool, LaTe [121]. LaTe is used to perform several measurements and provide interesting insights on the latency performance of IEEE 802.11p, such as how selecting a high priority Access Category (AC) can guarantee a better latency in presence of interfering traffic, as opposed to when a low priority AC is used.

Starting from LaTe, an open source long term network testing platform, called Long Term Network Tester (LTNT) has been developed, which has been used to assess the performance of a 5G automotive edge system. The system was tested to provide a one-way latency lower than 7 ms in both the UL and DL directions, accounting for the stringent latency requirements of vehicular networks. Then, the same Chapter presents an ongoing project related to the development of a full-fledged, plug and play, On-Board Unit, entirely based on open source software. A first prototype has been developed and tested in both urban and motorway scenarios, showing how it can interoperate with commercial devices and provide multi-stack connectivity to vehicles for research and experimentation purposes. Thanks to this project, it was also possible to gather interesting statistics on the CAM dynamic frequency management [63] in a real-world urban scenario.

Finally, two open frameworks are presented and extensively evaluated. The first, called Edge-V, is the first framework combining different technologies working in unlicensed spectrum bands (i.e., IEEE 802.11p, mmWave and standard Wi-Fi at 5 GHz) to enable Vehicular Edge Intelligence (VEI). It has been designed together with the definition of a mathematical optimization problem for task offloading in V2X scenarios. Then, a prototype based on off-the-shelf hardware and open source software has been developed to perform both laboratory and road tests. The obtained results have demonstrated the advantages of decentralized local offloading with respect to cloud-based approaches relying on cellular networks, and proved how Edge-V can enable high throughput and very low latency communication. The second framework, Open Radio Network Information Exchange (ONIX), is an open implementation of a Radio Network Information Service (RNIS) for 4G and 5G networks, targeted at vehicular applications. ONIX has been designed to be flexible with respect to different RAN deployment models and has been evaluated on

a dedicated testbed, focusing on low latency, resource usage and scalability. Besides the design and evaluation of ONIX, this thesis also presents a set of requirements for an effective RNI service.

Chapter 4 tackles instead the performance assessment of different access technologies and protocols, both in large scale simulations and in real-world outdoor scenarios. The measurements and their further analysis has been performed thanks to the open platforms and tools described in the previous Chapter, and to a novel simulation and emulation framework named ms-van3t (Multi-Stack VANET framework for ns-3). This thesis has presented our framework, released under an open source license [23], together with a comparison with other existing solutions for V2X simulations. ms-van3t has then been used to perform a comparison study, through simulations, between Release 14 LTE-V2X and IEEE 802.11p, showing how the latter provides the best average latency, while C-V2X appears to deliver a better overall Packet Reception Ratio (PRR) than IEEE 802.11p given a fixed transmission power level. Furthermore, ms-van3t comes with two sample applications, which have been evaluated both to show how our framework can be leveraged to test V2X applications, and the advantages brought by V2X connectivity to road safety and traffic efficiency scenarios. The open DSRC platform described in the previous Chapter has also been used to perform several field tests, aimed at determining the performance of IEEE 802.11p both in a laboratory environment, and on the road. The main focus was on latency, throughput, RSSI and maximum reachable distance, both in LOS and NLOS scenarios. An extensive field test campaign also allowed us to evaluate IEEE 802.11p in V2I scenarios, comparing it with two other promising technologies, i.e., IEEE 802.11ac and mmWave (IEEE 802.11ad @ 60 GHz). The results proved how mmWave can provide the best latency and throughput, while IEEE 802.11p appears to be the best technology in terms of connection stability until a limiting distance is reached (a quite stable connection can indeed be maintained until -87 dBm are reached, with is the lowest value among all the tested IEEE-based technologies).

Finally, Chapter 5 tackles the last important aspects related to the development of actual V2X services and novel protocols for connected and automated vehicles. These services and protocols can be deployed on the open platforms described in Chapter 3 and leverage the technologies tested in Chapter 4. First, we presented a novel approach for the exchange of raw GNSS data between vehicles, namely the Cooperative Enhancement Message (CEM) protocol, designed to be compatible with all the other protocols and message types already defined by ETSI. Our proposal has been extensively evaluated thanks to a special version of ms-van3t [25] and to a novel open dataset of 19 vehicular traces recorded by means of a high-accuracy GNSS RTK receiver. Then, we proposed the Server Local Dynamic Map (S-LDM) [27], an innovative centralized 5G-enabled MEC service targeted at collecting messages from a large number of vehicles. Our service stores a processed version of the data into a highly-efficient in-memory database, practically keeping

a centralized, enriched "map" of the road. When an "interesting" condition on the road is detected, the S-LDM is able to provide only the needed subset of data to other MEC services, for instance managing highly automated maneuvers in a centralized way. These services are thus offloaded from the necessity of processing a large amount of raw messages, which could reduce the performance of the actual latency-critical algorithms (e.g., control algorithms for centralized automated lane merge). The "interesting" conditions determining the transmission of data to other services include either a vehicle performing some action to signal that a maneuver with a high level of automation has been requested, or a situation on the road which potentially requires the intervention of a centralized MEC service. The S-LDM has been evaluated both in-lab and through the deployment to the MEC platforms of actual MNOs, as part of the 5G-CARMEN project. This enabled the execution of both in-country and cross-border tests with two Stellantis vehicles equipped with V2X OBUs. We were able to prove how the S-LDM is able to process each message with a very low latency, on average around a few tens of microseconds.

This thesis has thus proposed several open source solutions for V2X, including tools for assessing the performance of V2X applications and access technologies, together with the performance assessment of different technologies both in simulation and in the field. Furthermore, both a novel protocol and innovative MEC service have been proposed to the automotive research community for the next generation of connected vehicles.

The main focus has been put on open and customizable solutions, as they foster knowledge sharing among researchers and can be leveraged to provide fundamental insights for pushing research ahead towards the sough-after SAE automation levels 4 and 5.

List of acronyms

A-MPDU	Aggregated MAC Protocol Data Unit
A-MSDU	Aggregated MAC Service Data Unit
ADAS	Advanced Driver-Assistance System
AF	Application Function
AI	Aritifical Intelligence
ASA	Area Speed Advisor
ASIC	Application Specific Integrated Circuit
BIPW	Bump-In-The-Wire
BPSK	Binary Phase-Shift Keying
BSM	Basic Safety Message
BSS	Basic Service Set
BTP	Basic Transport Protocol
CA	Cooperative Awareness
CAM	Cooperative Awareness Message
CBR	Channel Busy Ratio
CCA	Clear Channel Assessment
CCH	Control Channel
CDF	Cumulative Distribution Function
CEM	Cooperative Enhancement Message
COCO	Common Objects in Context
CPS	Cooperative Perception Service
CQI	Channel Quality Indicator
CSMA	Carrier Sense Multiple Access
CSR	Common Safety Request
C-V2X	Cellular Vehicle-to-Everything

DCC	Decentralized Congestion Control
DEN	Decentralized Environmental Notification
DENM	Decentralized Environmental Notification Messages
DGNSS	Differential GNSS
DL	Deep Learning
DLC	Data Length Code
DNS	Domain Name System
DSRC	Dedicated Short-Range Communcations
eNB	E-UTRAN NodeB
ECU	Electronic Control Unit
EDCA	Enhanced Distributed Channel Access
EIRP	Equivalent Isotropic Radiated Power
EPC	Evolved Packet Core
EVA	Emergency Vehicle Alert
GLONASS	Globalnaya Navigazionnaya Sputnikovaya Sistema
GLOSA	Green Light Optimized Speed Advisory
GNSS	Global Navigation Satellite System
GPC	GNSS Positioning Correction
GPL	General Public License
GPLv2	General Public License version 2.0
GPS	Global Positioning System
GUI	Graphical User Interface
HAD	High Automated Driving
HIL	Hardware-in-the-Loop
HMI	Human-Machine Interface
ICA	Intersection Collision Avoidance
ICMP	Internet Control Message Protocol
ICRW	Intersection Collision Risk Warning
IMSI	International Mobile Subscriber Identity
INS	Inertial Navigation System
ISO	International Organization for Standardization
ITS	Intelligent Transport Systems

ITS-S	Intelligent Transport Systems Station
IVI	Infrastructure to Vehicle Information
IVIM	Infrastructure to Vehicle Information Message
JSON	JavaScript Object Notation
LaMP	Latency Measurement Protocol
LaTe	Latency Tester
LKA	Lane Keep Assist
LOS	Line-Of-Sight
LTNT	Long Term Network Tester
mmWave	Millimeter Wave
MAC	Medium Access Control
MAPEM	MAP (topology) Extended Message
MBMS	Multimedia Broadcast Multicast Service
MCS	Modulation and Coding Scheme
MIMO	Multiple-Input and Multiple-Output
MNO	Mobile Network Operator
ML	Machine Learning
MLME	MAC subLayer Management Entity
MTA	Magenta Telekom
MU-MIMO	Multi-User Multiple-Input and Multiple-Output
NEF	Network Exposure Function
NIC	Network Interface Card
NLOS	Non-Line-Of-Sight
NRZ	Non-Return-To-Zero
NR-V2X	New Radio Vehicle-to-Everything
NSA	Non-Standalone
NTP	Network Time Protocol
OBU	On-Board Unit
OCB	Outside the Context of a BSS
OEM	Original Equipment Manufacturer
ONIX	Open radio Network Information eXchange
ProSe	Proximity Services

PDCP	Packet Data Convergence Protocol
PHY	Physical (layer)
POC	Proof-Of-Concept
PPPP	ProSe Per Packet Priority
PPS	Packets-Per-Second
PRR	Packet Reception Ratio
PSID	Provider Service Identifier (PSID)
PTP	Precision Time Protocol
PVT	Position, Velocity and Time
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-shift Keying
RAN	Radio Access Network
REST	REpresentational State Transfer
RINEX	Receiver Independent Exchange Format
RNI	Radio Network Information
RNIS	Radio Network Information Service
RSA	Road Side Alert
RSRP	Reference Signals Received Power
RSRQ	Reference Signal Received Quality
RSSI	Received Signal Strength Indicator
RSU	Road Side Units
RTCM	Radio Technical Commission for Maritime Services
RTCMEM	Radio Technical Commission for Maritime Services Extended Message
RTK	Real-Time Kinematic
RTT	Round-Trip Time
S-LDM	Server Local Dynamic Map
SA	Standalone
SAE	Society of Automotive Engineers
SAEM	Services Announcement Essential Message
SAMARCANDA	Synthetic Accurate Multi-Agent RealistiC Assisted-gNss DatAset

SC-FDMA	Single-Carrier Frequency-Division Multiple Access
SCH	Service Channel
SDN	Software Defined Networking
SDR	Software-Defined Radio
SPATEM	Signal Phase And Timing Extended Message
SPS	Semi-Persistent Scheduling
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UE	User Equipment
URLLC	Ultra-Reliable Low Latency Communication
UPER	Unaligned Packed Encoding Rules
UPF	User Plane Function
UTC	Coordinated Universal Time
VANET	Vehicular Ad-Hoc Network
VDP	Vehicle Data Provider
VEI	Vehicular Edge Intelligence
VIF	Virtual Interface
VNMF	Virtualized Network Measurements Function
VRU	Vulnerable Road User
VUT	Vehicle Under Test
V2I	Vehicle-to-Infrastructure
V2N	Vehicle-to-Network
V2P	Vehicle-to-Pedestrian
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
WAVE	Wireless Access in Vehicular Environments
WLAN	Wireless Local Area Network
WSMP	WAVE Short Message Protocol

Bibliography

[1] M. Galvani, "History and future of driver assistance", *IEEE Instrumentation Measurement Magazine*, vol. 22, no. 1, pp. 11–16, 2019. DOI: 10.1109/MIM.2019.8633345.

[2] S. Chamraz and R. Balogh, "Two approaches to the adaptive cruise control (ACC) design", in *2018 Cybernetics Informatics (KI)*, 2018, pp. 1–6. DOI: 10.1109/CYBERI.2018.8337542.

[3] A. Mammeri, G. Lu, and A. Boukerche, "Design of lane keeping assist system for autonomous vehicles", in *2015 7th International Conference on New Technologies, Mobility and Security (NTMS)*, 2015, pp. 1–5. DOI: 10.1109/NTMS.2015.7266483.

[4] SAE, "SAE J3016 - Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles", en, Society of Automotive Engineers, Standard SAE J3016, 2021.

[5] F. Cunha, L. Villas, A. Boukerche, G. Maia, A. Viana, R. A. F. Mini, and A. A. F. Loureiro, "Data communication in VANETs: Protocols, applications and challenges", *Ad Hoc Networks*, vol. 44, pp. 90–103, 2016, ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2016.02.017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870516300580.

[6] F. Raviglione, C. M. R. Carletti, C. Casetti, F. Stoffella, G. M. Yilma, and F. Visintainer, "S-LDM: Server Local Dynamic Map for Vehicular Enhanced Collective Perception", in *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, 2022, pp. 1–5. DOI: 10.1109/VTC2022-Spring54318.2022.9860701.

[7] F. Raviglione, S. Zocca, A. Minetto, M. Malinverno, C. Casetti, C. Chiasserini, and F. Dovis, "From collaborative awareness to collaborative information enhancement in vehicular networks", *Vehicular Communications*, vol. 36, p. 100 497, 2022, ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2022.100497. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209622000444.

[8] ETSI, "ETSI TS 101 539-1 V1.1.1 (2013-08) - Intelligent Transport Systems (ITS); V2X Applications; Part 1: Road Hazard Signalling (RHS); application requirements specification", en, European Telecommunications Standards Institute, Standard ETSI TS 101 539-1 V1.1.1, 2013.

[9] ——, "ETSI TS 101 539-2 V1.1.1 (2018-06) - Intelligent Transport Systems (ITS); V2X Applications; Part 2: Intersection Collision Risk Warning (ICRW); application requirements specification", en, European Telecommunications Standards Institute, Standard ETSI TS 101 539-2 V1.1.1, 2018.

[10] ——, "ETSI TS 101 539-3 V1.1.1 (2013-11) - Intelligent Transport Systems (ITS); V2X Applications; Part 3: Longitudinal Collision Risk Warning (LCRW); application requirements specification", en, European Telecommunications Standards Institute, Standard ETSI TS 101 539-2 V1.1.1, 2013.

[11] Z. Li, T. Yu, R. Fukatsu, G. K. Tran, and K. Sakaguchi, "Towards Safe Automated Driving: Design of Software-Defined Dynamic MmWave V2X Networks and PoC Implementation", *IEEE Open Journal of Vehicular Technology*, vol. 2, pp. 78–93, 2021. DOI: 10.1109/OJVT.2021.3049783.

[12] F. Raviglione, M. Malinverno, S. Feraco, G. Avino, C. Casetti, C. F. Chiasserini, N. Amati, and J. Widmer, "Experimental Assessment of IEEE 802.11-Based V2I Communications", in *Proceedings of the 18th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, Ubiquitous Networks*, ser. PE-WASUN '21, Alicante, Spain: Association for Computing Machinery, 2021, pp. 33–40, ISBN: 9781450390781. DOI: 10.1145/3479240.3488506. [Online]. Available: https://doi.org/10.1145/3479240.3488506.

[13] Eurescom. "V2X in Europe or not? - Interview with 5GAA CTO Maxime Flament". https://www.eurescom.eu/eurescom-messages/winter-2019/v2x-in-europe-or-not/. (2022).

[14] National Highway Traffic Safety Administration, "National Highway Traffic Safety Administration Traffic Safety Facts - Early Estimates of Motor Vehicle Traffic Fatalities And Fatality Rate by Sub-Categories in 2021", Tech. Rep., May 2022. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/813298.

[15] 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. DOI: 10.23919/EETA.2018.8493191.

[16] M. Rapelli, C. Casetti, and M. Sgarbi, "A Distributed V2V-Based Virtual Traffic Light System", in *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*, 2020, pp. 122–128. DOI: 10.1109/COMSNETS48256.2020.9027339.

[17] 5G-CARMEN consortium, "Deliverable D5.1 - 5G-CARMEN Pilot plan", Tech. Rep., Dec. 2019. [Online]. Available: https://5gcarmen.eu/wp-content/uploads/2020/09/5G_CARMEN_D5.1_FINAL.pdf.

[18] S. Mumtaz, J. M. Jornet, J. Aulin, W. H. Gerstacker, X. Dong, and B. Ai, "Terahertz Communication for Vehicular Networks", *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, 2017.

[19] M. Karoui, V. Mannoni, B. Denis, and S. Mayrargue, "Performance Analysis of V2X-based Systems for Improved Vulnerable Road Users Safety", in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 3368–3373. DOI: 10.1109/ITSC55140.2022.9921841.

[20] F. Raviglione. "OpenWrt-V2X [online]". https://github.com/francescoraves483/OpenWrt-V2X. (2021).

[21] ETSI, "ETSI EN 302 636-4-1 V1.4.1 (2020-01) - 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", en, European Telecommunications Standards Institute, Standard ETSI EN 302 636-4-1, 2020.

[22] F. Raviglione. "LTNT (Long Term Network Tester) [online]". https://github.com/francescoraves483/LTNT. (2022).

[23] GitHub. "ms-van3t [online]". https://github.com/ms-van3t-devs/ms-van3t. (2022).

[24] nsnam. "ns-3 Network Simulator [online]". https://www.nsnam.org/. (2022).

[25] GitHub. "ms-van3t-CAM2CEM [online]". https://github.com/francescoraves483/ms-van3t-CAM2CEM. (2022).

[26] ETSI, "ETSI EN 302 895 V1.1.1 (2014-09) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM)", en, European Telecommunications Standards Institute, Standard ETSI EN 302 895 V1.1.1, 2014.

[27] GitHub. "GitHub - francescoraves483/S-LDM [online]". (2022), [Online]. Available: https://github.com/francescoraves483/S-LDM.

[28] 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. DOI: 10.1109/WoWMoM.2019.8793023.

[29] ——, "Open Source Testbed for Vehicular Communication", in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '19, Catania, Italy: Association for Computing Machinery, 2019, pp. 405–406, ISBN: 9781450367646. DOI: 10.1145/3323679.3326623. [Online]. Available: https://doi.org/10.1145/3323679.3326623.

[30] ——, "A Flexible, Protocol-Agnostic Latency Measurement Platform", in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5. DOI: 10.1109/VTCFall.2019.8891076.

[31] E. Coronado, F. Raviglione, M. Malinverno, C. Casetti, A. Cantarero, G. Cebrián-Márquez, and R. Riggio, "ONIX: Open Radio Network Information eXchange", *IEEE Communications Magazine*, vol. 59, no. 10, pp. 14–20, 2021. DOI: 10.1109/MCOM.101.2000900.

[32] 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*, ser. DIVANet '20, Alicante, Spain: Association for Computing Machinery, 2020, pp. 17–24, ISBN: 9781450381215. DOI: 10.1145/3416014.3424603. [Online]. Available: https://doi.org/10.1145/3416014.3424603.

[33] 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*, ser. TOP-Cars '19, Catania, Italy: Association for Computing Machinery, 2019, pp. 13–18, ISBN: 9781450368070. DOI: 10.1145/3331054.3331548. [Online]. Available: https://doi.org/10.1145/3331054.3331548.

[34] A. Minetto, S. Zocca, F. Raviglione, M. Malinverno, C. E. Casetti, C. F. Chiasserini, and F. Dovis, "Cooperative Localization Enhancement through GNSS Raw Data in Vehicular Networks", in *2021 IEEE Globecom Workshops (GC Wkshps)*, 2021, pp. 1–6. DOI: 10.1109/GCWkshps52748.2021.9682163.

[35] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration", *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017. DOI: 10.1109/COMST.2017.2705720.

[36] D. Patil and E. Al-Masri, "Seamless Service Migration across Multi-access Edge Computing (MEC) Environments", in *2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE)*, 2021, pp. 369–375. DOI: 10.1109/ECICE52819.2021.9645681.

[37] ETSI, 3GPP, "ETSI TS 122 185 V17.0.0 (2022-04) - LTE; Service requirements for V2X services (3GPP TS 22.185 version 17.0.0 Release 17)", en, European Telecommunications Standards Institute, Standard ETSI TS 122 185 V17.0.0, 2022.

[38] R. Zhang, F. Schmutz, K. Gerard, A. Pomini, L. Basseto, S. B. Hassen, A. Ishikawa, I. Ozgunes, and O. Tonguz, "Virtual Traffic Lights: System Design and Implementation", in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–5. DOI: 10.1109/VTCFall.2018.8690709.

[39] G. Sidorenko, J. Thunberg, K. Sjöberg, A. Fedorov, and A. Vinel, "Safety of Automatic Emergency Braking in Platooning", *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 2319–2332, 2022. DOI: 10.1109/TVT.2021.3138939.

[40] V. Group. "Volvo Trucks and FedEx demonstrate Truck Platooning [online]". https://www.volvogroup.com/en/news-and-media/news/2018/jun/news-2971141.html. (2018).

[41] R. L. Thompson, Z. Hu, J. Cho, J. Stovall, and M. Sartipi, "Enhancing Driver Awareness Using See-Through Technology", *SAE Technical Paper Series*, vol. 1, 2018.

[42] M. Giordani, A. Zanella, T. Higuchi, O. Altintas, and M. Zorzi, "On the Feasibility of Integrating mmWave and IEEE 802.11p for V2V Communications", in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–7. DOI: 10.1109/VTCFall.2018.8690697.

[43] ISTAT - Istituto Nazionale di Statistica, "Press release. Road accidents. Year 2021", ISTAT, Tech. Rep., Jul. 2022. [Online]. Available: https://www.istat.it/en/archivio/273327.

[44] WHO - World Health Organization, "Global Status Report on Road Safety - Time for Action", WHO, Tech. Rep., Jun. 2017, p. 301. [Online]. Available: https://www.afro.who.int/sites/default/files/2017-06/vid_global_status_report_en.pdf.

[45] ARIB, "ARIB STD-T109 1.3 - 700 MHz Band Intelligent Transport Systems", Association of Radio Industries and Businesses, Standard ARIB STD-T109, 2012.

[46] "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", *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021. DOI: 10.1109/IEEESTD.2021.9363693.

[47] ETSI, "ETSI ES 202 663 V1.1.0 (2010-01) - Intelligent Transport Systems (ITS); European profile standard for the physical and medium access control layer of Intelligent Transport Systems operating in the 5 GHz frequency band", en, European Telecommunications Standards Institute, Standard ETSI ES 202 663 V1.1.0, 2010.

[48] W.-F. Alliance. "Wi-Fi Certified TDLS: Easy-to-use, security-protected direct links to improve performance of Wi-Fi devices [online]". https://www.wi-fi.org/system/files/20120808_TDLS_White_Paper_FINAL.pdf. (2012).

[49] G. Naik, B. Choudhury, and J.-M. Park, "IEEE 802.11bd & 5G NR V2X: Evolution of Radio Access Technologies for V2X Communications", *IEEE Access*, vol. 7, pp. 70 169–70 184, 2019. DOI: 10.1109/ACCESS.2019.2919489.

[50] "IEEE Guide for Wireless Access in Vehicular Environments (WAVE) Architecture", *IEEE Std 1609.0-2019 (Revision of IEEE Std 1609.0-2013)*, pp. 1–106, 2019. DOI: 10.1109/IEEESTD.2019.8686445.

[51] "IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation", *IEEE Std 1609.4-2016 (Revision of IEEE Std 1609.4-2010)*, pp. 1–94, 2016. DOI: 10.1109/IEEESTD.2016.7435228.

[52] "IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Networking Services", *IEEE Std 1609.3-2016 (Revision of IEEE Std 1609.3-2010)*, pp. 1–160, 2016. DOI: 10.1109/IEEESTD.2016.7458115.

[53] "IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Identifiers", *IEEE Std 1609.12-2019 (Revision of IEEE Std 1609.12-2016)*, pp. 1–17, 2019. DOI: 10.1109/IEEESTD.2019.8877516.

[54] "IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages", *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)*, pp. 1–240, 2016. DOI: 10.1109/IEEESTD.2016.7426684.

[55] Society of Automotive Engineers, "Surface Vehicle Standard J2735 MAR2016 Dedicated Short Range Communications (DSRC) Message Set Dictionary", en, European Telecommunications Standards Institute, Standard SAE J2735, 2016.

[56] ——, "Surface Vehicle Standard J2945/1 MAR2016 On-Board System Requirements for V2V Safety Communications", en, European Telecommunications Standards Institute, Standard SAE J2945/1, 2016.

[57] ETSI, "ETSI EN 303 613 V1.1.1 (2020-01) - Intelligent Transport Systems (ITS); LTE-V2X Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band", en, European Telecommunications Standards Institute, Standard ETSI EN 303 613 V1.1.1, 2020.

[58] R. Molina-Masegosa, J. Gozalvez, and M. Sepulcre, "Configuration of the C-V2X Mode 4 Sidelink PC5 Interface for Vehicular Communication", in *2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, 2018, pp. 43–48. DOI: 10.1109/MSN.2018.00014.

[59] ETSI, "ETSI EN 302 636-5-1 V2.2.1 (2019-05) - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol", en, European Telecommunications Standards Institute, Standard ETSI EN 302 636-5-1, 2019.

[60] ——, "ETSI TS 103 248 V1.2.1 (2018-08) - Intelligent Transport Systems (ITS); GeoNetworking; Port Numbers for the Basic Transport Protocol (BTP)", en, European Telecommunications Standards Institute, Standard ETSI TS 103 248 V1.2.1, 2018.

[61] ——, "ETSI EN 302 665 V1.1.1 (2010-09) - European Standard (Telecommunications series) Intelligent Transport Systems (ITS); Communications Architecture", en, European Telecommunications Standards Institute, Standard ETSI EN 302 665, 2010.

[62] ——, "ETSI TS 102 894-1 V1.1.1 (2013-08) - Intelligent Transport Systems (ITS); Users and applications requirements; Part 1: Facility layer structure, functional requirements and specifications", en, European Telecommunications Standards Institute, Standard ETSI TS 102 894-1 V1.1.1, 2013.

[63] ——, "ETSI EN 302 637-2 V1.4.1 (2019-04) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service", en, European Telecommunications Standards Institute, Standard ETSI EN 302 637-2 V1.4.1, 2019.

[64] ——, "ETSI EN 302 637-3 V1.3.1 (2019-04) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service", en, European Telecommunications Standards Institute, Standard ETSI EN 302 637-3 V1.3.1, 2019.

[65] ——, "ETSI TS 102 894-2 V1.3.1 (2018-08) - Intelligent Transport Systems (ITS); Users and applications requirements; Part 2: Applications and facilities layer common data dictionary", en, European Telecommunications Standards Institute, Standard ETSI TS 102 894-2 V1.3.1, 2018.

[66] ——, "ETSI TS 103 301 V2.1.1 (2021-03) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Facilities layer protocols and communication requirements for infrastructure services; Release 2", en, European Telecommunications Standards Institute, Standard ETSI TS 103 301 V2.1.1, 2021.

[67] ISO, "BS EN ISO 14823:2017 - Intelligent transport systems - Graphic data dictionary", en, International Organization for Standardization, Standard ISO 14823, 2017.

[68] ETSI, "ETSI TS 103 300-3 V2.1.2 (2021-04) - Intelligent Transport Systems (ITS); Vulnerable Road Users (VRU) awareness; Part 3: Specification of VRU awareness basic service; Release 2", en, European Telecommunications Standards Institute, Standard ETSI TS 103 300-3 V2.1.2, 2021.

[69] ——, "ETSI TR 103 562 V2.1.1 (2019-12) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS); Release 2", en, European Telecommunications Standards Institute, Standard ETSI TR 103 562 V2.1.1, 2019.

[70] GitHub. "asn1c [online]". https://github.com/vlm/asn1c. (2021).

[71] D. Mundy and D. Chadwick, "An XML alternative for performance and security: ASN.1", *IT Professional*, vol. 6, no. 1, pp. 30–36, 2004. DOI: 10.1109/MITP.2004.1265540.

[72] ETSI, "ETSI TS 102 687 V1.1.1 (2011-07) - Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part", en, European Telecommunications Standards Institute, Standard ETSI TS 102 687 V1.1.1, 2011.

[73] N. Lyamin, A. Vinel, D. Smely, and B. Bellalta, "ETSI DCC: Decentralized Congestion Control in C-ITS", *IEEE Communications Magazine*, vol. 56, no. 12, pp. 112–118, 2018. DOI: 10.1109/MCOM.2017.1700173.

[74] ETSI, "ETSI TS 102 687 V1.2.1 (2018-04) - Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part", en, European Telecommunications Standards Institute, Standard ETSI TS 102 687 V1.2.1, 2018.

[75] 3GPP, "Vocabulary for 3GPP Specifications", 3rd Generation Partnership Project (3GPP), Technical Report (TR) 21.905.

[76] ——, "Architecture enhancements for V2X services", 3rd Generation Partnership Project (3GPP), Technical Specifications (TS) 23.285.

[77] 5GAA - 5G Automotive Association, "5G Automotive Association; Working Group System Architecture and Solution Development; 5GAA V2X Terms and Definitions", 5GAA, Tech. Rep., Jul. 2017. [Online]. Available: https://5gaa.org/wp-content/uploads/2017/08/5GAA-V2X-Terms-and-Definitions110917.pdf.

[78] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A Tutorial on 5G NR V2X Communications", *IEEE Communications Surveys Tutorials*, vol. 23, no. 3, pp. 1972–2026, 2021. DOI: 10.1109/COMST.2021.3057017.

[79] R. Molina-Masegosa, J. Gozalvez, and M. Sepulcre, "Comparison of IEEE 802.11p and LTE-V2X: An Evaluation With Periodic and Aperiodic Messages of Constant and Variable Size", *IEEE Access*, vol. 8, pp. 121 526–121 548, 2020. DOI: 10.1109/ACCESS.2020.3007115.

[80] G. P. Wijesiri N.B.A., J. Haapola, and T. Samarasinghe, "A Discrete-Time Markov Chain Based Comparison of the MAC Layer Performance of C-V2X Mode 4 and IEEE 802.11p", *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2505–2517, 2021. DOI: 10.1109/TCOMM.2020.3044340.

[81] W. Anwar, N. Franchi, and G. Fettweis, "Physical Layer Evaluation of V2X Communications Technologies: 5G NR-V2X, LTE-V2X, IEEE 802.11bd, and IEEE 802.11p", in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–7. DOI: 10.1109/VTCFall.2019.8891313.

[82] P. Valerio. "IoT Times - NXP, Volkswagen, and several European projects bet on WiFi DSRC for V2X [online]". https://iot.eetimes.com/nxp-volkswagen-and-several-european-projects-bet-on-wifi-dsrc-for-v2x/. (2021).

[83] V. Todisco, S. Bartoletti, C. Campolo, A. Molinaro, A. O. Berthet, and A. Bazzi, "Performance Analysis of Sidelink 5G-V2X Mode 2 Through an Open-Source Simulator", *IEEE Access*, vol. 9, pp. 145 648–145 661, 2021. DOI: 10.1109/ACCESS.2021.3121151.

[84] M. Harounabadi, D. M. Soleymani, S. Bhadauria, M. Leyh, and E. Roth-Mandutz, "V2X in 3GPP Standardization: NR Sidelink in Release-16 and Beyond", *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 12–21, 2021. DOI: 10.1109/MCOMSTD.001.2000070.

[85] ETSI, "ETSI TS 103 574 V1.1.1 (2018-11) - Intelligent Transport Systems (ITS); Congestion Control Mechanisms for the C-V2X PC5 interface; Access layer part", en, European Telecommunications Standards Institute, Standard ETSI TS 103 574 V1.1.1, 2018.

[86] K. F. Haque, A. Abdelgawad, V. P. Yanambaka, and K. Yelamarthi, "A LoRa Based Reliable and Low Power Vehicle to Everything (V2X) Communication Architecture", in *2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, 2020, pp. 177–182. DOI: 10.1109/iSES50453.2020.00047.

[87] Apache. "How does a Queue compare to a Topic [online]". https://activemq.apache.org/how-does-a-queue-compare-to-a-topic. (2022).

[88] Advanced Message Queuing Protocol (AMQP) Working Group, "AMQP Specification v1.0 - Revision: 1350 (07 Oct 2011)", en, Standard, 2011. [Online]. Available: https://www.amqp.org/sites/amqp.org/files/amqp.pdf.

[89] J. Schwartz. "Bing Maps Tile System [online]". https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system. (2022).

[90] 5G-CARMEN consortium, "5G-CARMEN D3.3, Intermediate report on 5G Technological Enablers for CCAM", Jul. 2021. [Online]. Available: https://5gcarmen.eu/wp-content/uploads/2021/10/D3.3-July-2021.pdf.

[91] CSS Electronics, *CAN bus - the ultime guide*. CSS Electronics, 2022.

[92] Cohda Wireless. "MK5 OBU [online]". https://www.cohdawireless.com/solutions/hardware/mk5-obu/. ().

[93] ——, "MK6C EVK [online]". https://www.cohdawireless.com/solutions/hardware/mk6c-evk/. ().

[94] Commsignia. "Powerful V2X Onboard Unit [online]". https://www.commsignia.com/products/obu/. ().

[95] Danlaw. "AutoLink - V2X Aftermarket Safety Device [online]". https://www.danlawinc.com/wp-content/uploads/DS_Aftermarket_Safety_Device-V10.pdf. ().

[96] Lacroix. "Neavia V2V Unit - On Board Unit for Cooperative Vehicles [online]". https://www.lacroix-city.com/wp-content/uploads/sites/7/2019/10/datasheet-Neavia_V2V_Unit-US-03182019.pdf. ().

[97] Harman. "HARMAN Savari MobiWAVE [online]". https://car.harman.com/solutions/connectivity/harman-savari-mobiwave. ().

[98] Quectel. "C-V2X AG15 [online]". https://www.quectel.com/product/c-v2x-ag15. ().

[99] Unex. "Innovative V2X System-on-Module [online]". https://www.unex.com.tw/som/. ().

[100] T. M. Sales. "Toyota and Lexus to Launch Technology to Connect Vehicles and Infrastructure in the U.S. in 2021 [online]". (Apr. 2018), [Online]. Available: https://pressroom.toyota.com/toyota-and-lexus-to-launch-technology-connect-vehicles-infrastructure-in-u-s-2021/.

[101] D. Richter, L. Pirl, J. Beilharz, C. Werling, and A. Polze, "Performance of Real-Time Wireless Communication for Railway Environments with IEEE 802.11p", in *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS)*, Jan. 8, 2019, ISBN: 978-0-9981331-2-6. DOI: 10.24251/HICSS.2019.907. [Online]. Available: http://scholarspace.manoa.hawaii.edu/handle/10125/60190.

[102] 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)*, Jun. 2012, pp. 127–132. DOI: 10.1109/MedHocNet.2012.6257112.

[103] Z. Qin, Z. Meng, X. Zhang, B. Xiang, and L. Zhang, "Performance evaluation of 802.11p WAVE system on embedded board", in *The International Conference on Information Networking 2014 (ICOIN2014)*, Feb. 2014, pp. 356–360. DOI: 10.1109/ICOIN.2014.6799704.

[104] 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)*, Apr. 2012, pp. 1–4. DOI: 10.1109/CCECE.2012.6334893.

[105] 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)*, Dec. 2016, pp. 1–2. DOI: 10.1109/VNC.2016.7835955.

[106] G. P. 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)*, Jul. 2010, pp. 449–453.

[107] 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)*, Jul. 2017, pp. 1–5. DOI: 10.1109/TENCONSpring.2017.8070033.

[108] J. de Jongh, J. van de Sluis, D. Heuven, A. Voronov, and I. Passchier, "IEEE 802.11p [CTU-IIG] on PCEngines APU1D running Voyage", Feb. 2016.

[109] P. Engines. "PC Engines APU2E4 product file [online]". (2021), [Online]. Available: https://www.pcengines.ch/apu2e4.htm.

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

[111] GitHub. "GitHub - Rawsock library [online]". (2020), [Online]. Available: https://github.com/francescoraves483/Rawsock_lib.

[112] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Performance Assessment of IEEE 802.11p with an Open Source SDR-Based Prototype", *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1162–1175, May 2018, ISSN: 1536-1233. DOI: 10.1109/TMC.2017.2751474.

[113] GitHub. "GitHub - spectools-dsrc [online]". (2019), [Online]. Available: https://github.com/francescoraves483/spectools-dsrc.

[114] F. Raviglione. "V2X-LaTe-GUI-tools [online]". https://github.com/francescoraves483/V2X-LaTe-GUI-tools. (2019).

[115] ——, "LaTe + LaMP home page [online]". https://francescoraves483.github.io/LaMP_LaTe/. (2020).

[116] D. Rossi Mafioletti *et al.*, "Metherxis: Virtualized Network Functions for Micro-second Grade Latency Measurements", in *ACM LANCOMM'16*, Aug. 2016, pp. 22–24. DOI: 10.1145/2940116.2940131.

[117] K. Hedayat *et al.*, "A Two-Way Active Measurement Protocol (TWAMP)", RFC Editor, RFC 5357, Oct. 2018, pp. 1–26. DOI: 10.17487/RFC5357. [Online]. Available: https://www.rfc-editor.org/rfc/rfc5357.txt.

[118] J. Sherry, "Applications of the IP Timestamp Option to Internet Measurement", Dec. 2010.

[119] M. Kerrisk. "icmp(7) - Linux manual page [online]". https://man7.org/linux/man-pages/man7/icmp.7.html. (2017).

[120] F. Raviglione, M. Malinverno and C. Casetti, "Latency Measurement Protocol - Custom protocol specifications - Revision 2.0", en, Politecnico di Torino, Standard Latency Measurement Protocol Revision 2.0, 2019. [Online]. Available: https://francescoraves483.github.io/LaMP_LaTe/LaMP/LaMP_specifications_rev2.0.pdf.

[121] GitHub. "LaTe main repository - development branch [online]". https://github.com/francescoraves483/LaMP_LaTe/tree/development. (2022).

[122] M. Kerrisk. "cmsg(3) - Linux manual page [online]". https://man7.org/linux/man-pages/man3/cmsg.3.html. (2021).

[123] "Graphite [online]". https://graphiteapp.org/. (2022).

[124] "Grafana [online]". https://grafana.com/. (2022).

[125] The perfSONAR Project and contributors. "perfSONAR Home [online]". https://www.perfsonar.net/. (2022).

[126] K. Saputra, N. Nazaruddin, D. H. Yunardi, and R. Andriyani, "Implementation of Haversine Formula on Location Based Mobile Application in Syiah Kuala University", in *2019 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, 2019, pp. 40–45. DOI: 10.1109/CYBERNETICSCOM.2019.8875686.

[127] T. Shimizu, V. Va, G. Bansal, and R. W. Heath, "Millimeter Wave V2X Communications: Use Cases and Design Considerations of Beam Management", in *2018 Asia-Pacific Microwave Conference (APMC)*, IEEE, 2018, pp. 183–185.

[128] M. Kutila, P. Pyykonen, Q. Huang, W. Deng, W. Lei, and E. Pollakis, "C-V2X Supported Automated Driving", in *Proceedings of IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2019, pp. 1–5.

[129] D. Maggiorini, C. Quadri, and L. A. Ripamonti, "Opportunistic Mobile Games Using Public Transportation Systems: a Deployability Study", *Multimedia systems*, vol. 20, no. 5, pp. 545–562, 2014.

[130] J. Choi, V. Va, N. Gonzalez-Prelcic, R. Daniels, C. R. Bhat, and R. W. Heath, "Millimeter-wave vehicular communication to support massive automotive sensing", *IEEE Communications Magazine*, vol. 54, no. 12, pp. 160–167, 2016.

[131] M. Drago, T. Zugno, M. Polese, M. Giordani, and M. Zorzi, "MilliCar: An ns-3 module for mmWave NR V2X networks", in *Proceedings of the 2020 Workshop on ns-3*, 2020, pp. 9–16.

[132] S. A. A. Shah, E. Ahmed, M. Imran, and S. Zeadally, "5G for Vehicular Communications", *IEEE Communications Magazine*, vol. 56, no. 1, pp. 111–117, 2018.

[133] 5G Americas, "5g: The future of iot", 5G Americas, Bellevue, Washington, Tech. Rep., Jul. 2019, p. 121. [Online]. Available: https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Americas_White_Paper_on_5G_IOT_FINAL_7.16.pdf.

[134] P. Dai, K. Hu, X. Wu, H. Xing, and Z. Yu, "Asynchronous Deep Reinforcement Learning for Data-Driven Task Offloading in MEC-Empowered Vehicular Networks", in *IEEE INFOCOM 2021*, 2021, pp. 1–10. DOI: 10.1109/INFOCOM42981.2021.9488886.

[135] A. Molina-Galan, B. Coll-Perales, and J. Gozalvez, "C-V2X Assisted mmWave V2V Scheduling", in *2019 IEEE 2nd Connected and Automated Vehicles Symposium (CAVS)*, 2019, pp. 1–5. DOI: 10.1109/CAVS.2019.8887840.

[136] M. Polese, M. Giordani, T. Zugno, A. Roy, S. Goyal, D. Castor, and M. Zorzi, "Integrated Access and Backhaul in 5G mmWave Networks: Potential and Challenges", *IEEE Communications Magazine*, vol. 58, no. 3, pp. 62–68, 2020. DOI: 10.1109/MCOM.001.1900346.

[137] L. Liang, H. Ye, and G. Y. Li, "Toward Intelligent Vehicular Networks: A Machine Learning Framework", *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 124–135, 2018.

[138] W. Miao, G. Min, X. Zhang, Z. Zhao, and J. Hu, "Performance Modelling and Quantitative Analysis of Vehicular Edge Computing with Bursty Task Arrivals", *IEEE Transactions on Mobile Computing*, 2021.

[139] H. Ye, L. Liang, G. Ye Li, J. Kim, L. Lu, and M. Wu, "Machine Learning for Vehicular Networks: Recent Advances and Application Examples", *IEEE Vehicular Technology Magazine*, vol. 13, no. 2, pp. 94–101, 2018. DOI: 10.1109/MVT.2018.2811185.

[140] S. Wang, J. Huang, and X. Zhang, "Demystifying Millimeter-Wave V2X: Towards Robust and Efficient Directional Connectivity under High Mobility", in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, New York, NY, USA: ACM, 2020, ISBN: 9781450370851. [Online]. Available: https://doi.org/10.1145/3372224.3419208.

[141] E. Krijestorac, A. Memedi, T. Higuchi, S. Ucar, O. Altintas, and D. Cabric, "Hybrid Vehicular and Cloud Distributed Computing: A Case for Cooperative Perception", in *GLOBECOM 2020*, 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9322247.

[142] B.-J. Qiu, C.-Y. Hsieh, J.-C. Chen, and F. Dressler, "DCOA: Double-Check Offloading Algorithm to Road-Side Unit and Vehicular Micro-Cloud in 5G Networks", in *GLOBECOM 2020*, 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9348224.

[143] C. Tang, C. Zhu, X. Wei, Q. Li, and J. J. P. C. Rodrigues, "Task Caching in Vehicular Edge Computing", in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–6. DOI: 10.1109/INFOCOMWKSHPS51825.2021.9484498.

[144] T. Higuchi, S. Ucar, and O. Altintas, "Offloading Tasks to Vehicular Virtual Edge Servers", in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW)*, 2019, pp. 162–163. DOI: 10.1109/MASSW.2019.00040.

[145] Y. He, D. Zhai, R. Zhang, J. Du, G. S. Aujla, and H. Cao, "A Mobile Edge Computing Framework for Task Offloading and Resource Allocation in UAV-assisted VANETs", in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–6. DOI: 10.1109/INFOCOMWKSHPS51825.2021.9484643.

[146] MiktroTik. "MiktroTik Routers and Wireless - Products: wAP 60G [online]". https://mikrotik.com/product/wap_60g. (2022).

[147] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context", in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 740–755, ISBN: 978-3-319-10602-1.

[148] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3", *CoRR*, vol. abs/1905.02244, 2019. arXiv: 1905.02244. [Online]. Available: http://arxiv.org/abs/1905.02244.

[149] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021", *arXiv preprint arXiv:2107.08430*, 2021.

[150] ETSI, "ETSI GS MEC 003 V2.2.1 (2020-12) - Multi-access Edge Computing (MEC); Framework and Reference Architecture", en, European Telecommunications Standards Institute, Standard ETSI GS MEC 003 V2.2.1, 2020.

[151] S. Arora, P. A. Frangoudis, and A. Ksentini, "Exposing radio network information in a MEC-in-NFV environment: the RNISaaS concept", in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 306–310. DOI: 10.1109/NETSOFT.2019.8806647.

[152] M. Nasimi, M. A. Habibi, B. Han, and H. D. Schotten, "Edge-Assisted Congestion Control Mechanism for 5G Network Using Software-Defined Networking", in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*, 2018, pp. 1–5. DOI: 10.1109/ISWCS.2018.8491233.

[153] Y. Tan, C. Han, M. Luo, X. Zhou, and X. Zhang, "Radio network-aware edge caching for video delivery in MEC-enabled cellular networks", in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2018, pp. 179–184. DOI: 10.1109/WCNCW.2018.8368984.

[154] ETSI, "ETSI GR MEC 031 V2.1.1 (2020-10) - Multi-access Edge Computing (MEC) MEC 5G Integration", en, European Telecommunications Standards Institute, Standard ETSI GR MEC 031 V2.1.1, 2020.

[155] ——, "ETSI GS MEC 012 V2.1.1 (2019-12) - Multi-access Edge Computing (MEC); Radio Network Information API", en, European Telecommunications Standards Institute, Standard ETSI GS MEC 012 V2.1.1, 2019.

[156] ——, "ETSI GS NFV-IFA 011 V4.1.1 (2020-11) - Network Functions Virtualisation (NFV) Release 4; Management and Orchestration; VNF Descriptor and Packaging Specification", en, European Telecommunications Standards Institute, Standard ETSI GS NFV-IFA 011 V4.1.1, 2020.

[157] O-RAN Alliance, "O-RAN Architecture Description", O-RAN Alliance, Tech. Rep., 2020.

[158] European Telecommunications Standards Institute, "MEC Deployments in 4G and Evolution Towards 5G, White Paper 24", ETSI, Tech. Rep., Feb. 2018, p. 24. [Online]. Available: https://www.etsi.org/images/files/etsiwhitepapers/etsi_wp24_mec_deployment_in_4g_5g_final.pdf.

[159] E. Coronado, S. N. Khan, and R. Riggio, "5G-EmPOWER: A Software-Defined Networking Platform for 5G Radio Access Networks", *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 715–728, 2019. DOI: 10.1109/TNSM.2019.2908675.

[160] S. R. Systems. "srsRAN - Your own mobile network [online]". https://www.srslte.com/. (2022).

[161] "Open5GS - Open source project of 5GC and EPC (Release-16) [online]". https://open5gs.org/. (2022).

[162] E. Coronado, Z. Yousaf, and R. Riggio, "LightEdge: Mapping the Evolution of Multi-Access Edge Computing in Cellular Networks", *IEEE Communications Magazine*, vol. 58, no. 4, pp. 24–30, 2020. DOI: 10.1109/MCOM.001.1900690.

[163] Apache. "Apache ActiveMQ - Flexible Powerful Open Source Multi-Protocol Messaging [online]". https://activemq.apache.org/. (2022).

[164] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond Throughput: A 4G LTE Dataset with Channel and Context Metrics", in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18, Amsterdam, Netherlands: Association for Computing Machinery, 2018, pp. 460–465, ISBN: 9781450351928. DOI: 10.1145/3204949.3208123. [Online]. Available: https://doi.org/10.1145/3204949.3208123.

[165] ETSI, "ETSI TS 136 331 V8.18.0 (2013-02) - LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification (3GPP TS 36.331 version 8.18.0 Release 8)", en, European Telecommunications Standards Institute, Standard ETSI TS 136 331 V8.18.0, 2013.

[166] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis", *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, 2011. DOI: 10.1109/TMC.2010.133.

[167] 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.

[168] R. Riebl, H.-J. Gunther, C. Facchi, and L. Wolf, "Artery: Extending Veins for VANET applications", Jun. 2015, pp. 450–456. DOI: 10.1109/MTITS.2015.7223293.

[169] F. Eckermann, M. Kahlert, and C. Wietfeld, "Performance Analysis of C-V2X Mode 4 Communication Introducing an Open-Source C-V2X Simulator", Sep. 2019, pp. 1–5. DOI: 10.1109/VTCFall.2019.8891534.

[170] Z. Ali, S. Lagén, L. Giupponi, and R. Rouil, "3GPP NR V2X Mode 2: Overview, Models and System-Level Evaluation", *IEEE Access*, vol. 9, pp. 89 554–89 579, 2021. DOI: 10.1109/ACCESS.2021.3090855.

[171] A. Wegener, M. Piorkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: An Interface for Coupling Road Traffic and Network Simulators", *Proceedings of the 11th Communications and Networking Simulation Symposium, CNS'08*, Apr. 2008. DOI: 10.1145/1400713.1400740.

[172] A. Abunei, C.-R. 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.

[173] M. Rondinone, J. Maneros, D. Krajzewicz, R. Bauza, P. Cataldi, F. Hrizi, J. Gozalvez, V. Kumar, M. Röckl, L. Lin, O. Lazaro, J. Leguay, J. Härri, S. Vaz, Y. Lopez, M. Sepulcre, M. Wetterwald, R. Blokpoel, and F. Cartolano, "iTETRIS: A modular simulation platform for the large scale evaluation of cooperative ITS applications", *Simulation Modelling Practice and Theory*, vol. 34, pp. 99–125, 2013, ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2013.01.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X1300018X.

[174] G. Shah, R. Valiente Romero, N. Gupta, S. M. O. Gani, B. Toghi, Y. Fallah, and S. Gupta, "Real-Time Hardware-In-the-Loop Emulation Framework for DSRC-based Connected Vehicle Applications", May 2019.

[175] B. Mafakheri, P. Gonnella, A. Bazzi, B. Masini, M. Caggiano, and R. Verdone, "Optimizations for Hardware-in-the-Loop-Based V2X Validation Platforms", Feb. 2021. DOI: 10.1109/VTC2021-Spring51267.2021.9448667.

[176] M. Amoozadeh, B. Ching, C.-N. Chuah, D. Ghosal, and H. M. Zhang, "VENTOS: Vehicular Network Open Simulator with Hardware-in-the-Loop Support", *Procedia Computer Science*, vol. 151, pp. 61–68, 2019, The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.04.012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919304739.

[177] C. Obermaier, R. Riebl, and C. Facchi, "Fully Reactive Hardware-in-the-Loop Simulation for VANET Devices", in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3755–3760. DOI: 10.1109/ITSC.2018.8569663.

[178] 3GPP, "Technical Specification Group Radio Access Network; Study on LTE-based V2X Services; Release 14", 3rd Generation Partnership Project (3GPP), Technical Report (TR) 36.885, Apr. 2015, Version 14.0.0.

[179] GitHub. "ns3 module for bidirectional coupling with SUMO". https://github.com/vodafone-chair/ns3-sumo-coupling. (2019).

[180] S. Ollander, F. A. Schiegg, F.-W. Bode, and M. Baum, "Dual-frequency Collaborative Positioning for Minimization of GNSS Errors in Urban Canyons", in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 2020, pp. 1–8. DOI: 10.23919/FUSION45008.2020.9190612.

[181] ETSI, "ETSI TS 103 301 V1.1.1 (2016-11) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Facilities layer protocols and communication requirements for infrastructure services", en, European Telecommunications Standards Institute, Standard ETSI TS 103 301, 2016.

[182] V. Vukadinovic, K. Bakowski, P. Marsch, I. D. Garcia, H. Xu, M. Sybis, P. Sroka, K. Wesolowski, D. Lister, and I. Thibault, "3GPP C-V2X and IEEE 802.11 p for Vehicle-to-Vehicle communications in highway platooning scenarios", *Ad Hoc Networks*, vol. 74, pp. 17–29, 2018.

[183] V. Maglogiannis, D. Naudts, S. Hadiwardoyo, D. van den Akker, J. Marquez-Barja, and I. Moerman, "Experimental v2x evaluation for c-v2x and its-g5 technologies in a real-life highway environment", *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1521–1538, 2022. DOI: 10.1109/TNSM.2021.3129348.

[184] Y. Xiao and J. Rosdahl, "Throughput and delay limits of IEEE 802.11", *IEEE Communications Letters*, vol. 6, no. 8, pp. 355–357, 2002. DOI: 10.1109/LCOMM.2002.802035.

[185] W. Vandenberghe, I. Moerman, and P. Demeester, "Approximation of the IEEE 802.11p standard using commercial off-the-shelf IEEE 802.11a hardware", in *2011 11th International Conference on ITS Telecommunications*, 2011, pp. 21–26. DOI: 10.1109/ITST.2011.6060057.

[186] M. Shi, Y. Zhang, D. Yao, and C. Lu, "Application-oriented performance comparison of 802.11p andLTE-V in a V2V communication system", *Tsinghua Science and Technology*, vol. 24, no. 2, pp. 123–133, 2019. DOI: 10.26599/TST.2018.9010075.

[187] A. Mednis, "Development of 802.11p testbed - experiences", in *2014 14th Biennial Baltic Electronic Conference (BEC)*, 2014, pp. 137–140. DOI: 10.1109/BEC.2014.7320575.

[188] X. Chen and D. Yao, "An empirically comparative analysis of 802.11n and 802.11p performances in CVIS", in *2012 12th International Conference on ITS Telecommunications*, 2012, pp. 848–851. DOI: 10.1109/ITST.2012.6425303.

[189] M. Klapez, C. A. Grazia, and M. Casoni, "Application-Level Performance of IEEE 802.11p in Safety-Related V2X Field Trials", *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 3850–3860, 2020. DOI: 10.1109/JIOT.2020.2967649.

[190] Z. Liu, Z. Liu, Z. Meng, X. Yang, L. Pu, and L. Zhang, "Implementation and performance measurement of a V2X communication system for vehicle and pedestrian safety", *International Journal of Distributed Sensor Networks*, vol. 12, no. 9, p. 1 550 147 716 671 267, 2016. DOI: 10.1177/1550147716671267.

[191] V. P. Sarvade and S. A. Kulkarni, "Performance analysis of IEEE 802.11ac for vehicular networks using realistic traffic scenarios", in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 137–141. DOI: 10.1109/ICACCI.2017.8125830.

[192] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer, "IEEE 802.11 ad: directional 60 GHz communication for multi-Gigabit-per-second Wi-Fi", *IEEE Communications Magazine*, vol. 52, no. 12, pp. 132–141, 2014.

[193] M. Giordani, A. Zanella, and M. Zorzi, "LTE and millimeter waves for V2I communications: An end-to-end performance comparison", in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, IEEE, 2019, pp. 1–7.

[194] C. Perfecto, J. Del Ser, and M. Bennis, "Millimeter-wave V2V communications: Distributed association and beam alignment", *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 9, pp. 2148–2162, 2017.

[195] G. Bhanage, *AMSDU vs AMPDU: A Brief Tutorial on WiFi Aggregation Support*, 2017. DOI: 10.48550/ARXIV.1704.07015. [Online]. Available: https://arxiv.org/abs/1704.07015.

[196] A. Blanco, P. J. Mateo, F. Gringoli, and J. Widmer, "Augmenting MmWave Localization Accuracy through Sub-6 GHz on off-the-Shelf Devices", in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, ser. MobiSys '22, Portland, Oregon: Association for Computing Machinery, 2022, pp. 477–490, ISBN: 9781450391856. DOI: 10.1145/3498361.3538920. [Online]. Available: https://doi.org/10.1145/3498361.3538920.

[197] P. J. Mateo. "Researcher version of Openwrt for wAP 60G/LHGG-60ad/wAP 60Gx3 AP [online]". https://github.com/IMDEANetworksWNG/Mikrotik-researcher-tools. (2021).

[198] iPerf. "iPerf - The TCP, UDP and SCPT network bandwidth measurement tool [online]". https://iperf.fr/. (2019).

[199] Francesco Raviglione, "Setting up relayd to create a bridge between Ethernet and 802.11 in OCB mode", Politecnico di Torino, Tech. Rep., Jul. 2019, p. 8. [Online]. Available: https://raw.githubusercontent.com/francescoraves483/OpenWrt-V2X/OpenWrt-V2X-21.02.1/docs/Setting%5C%20up%5C%20relayd%5C%20bridge%5C%20802_11p.pdf.

[200] Z. Amjad, A. Sikora, B. Hilt, and J.-P. Lauffenburger, "Low Latency V2X Applications and Network Requirements: Performance Evaluation", in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 220–225. DOI: 10.1109/IVS.2018.8500531.

[201] N. Alam, A. Kealy, and A. G. Dempster, "Cooperative Inertial Navigation for GNSS-Challenged Vehicular Environments", *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1370–1379, 2013. DOI: 10.1109/TITS.2013.2261063.

[202] C. Gao, G. Zhao, and H. Fourati, *Cooperative localization and navigation: theory, research, and Practice*. CRC Press, 2019.

[203] G. Thandavarayan, M. Sepulcre, and J. Gozalvez, "Generation of Cooperative Perception Messages for Connected and Automated Vehicles", *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 336–16 341, 2020. DOI: 10.1109/TVT.2020.3036165.

[204] F. A. Schiegg, N. Brahmi, and I. Llatser, "Analytical Performance Evaluation of the Collective Perception Service in C-V2X Mode 4 Networks", in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 181–188. DOI: 10.1109/ITSC.2019.8917214.

[205] F. A. Schiegg, D. Bischoff, J. R. Krost, and I. Llatser, "Analytical Performance Evaluation of the Collective Perception Service in IEEE 802.11p Networks", in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020, pp. 1–6. DOI: 10.1109/WCNC45663.2020.9120490.

[206] GSA working group, *Using GNSS Raw Measurements on Android Devices*, https://www.gsa.europa.eu/system/files/reports/gnss_raw_measurement_web_0.pdf, 2018.

[207] A. Privat, M. Pascaud, and D. Laurichesse, "Innovative smartphone applications for Precise Point Positioning", in *2018 SpaceOps Conference*. 2018. DOI: 10.2514/6.2018-2324. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2018-2324. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2018-2324.

[208] X. Zhang, X. Tao, X. S. F. Zhu, and F. Wang, "Quality assessment of GNSS observations from an Android N smartphone and positioning performance analysis using time-differenced filtering approach", *GPS Solutions*, vol. 22, no. 3, p. 70, May 2018, ISSN: 1521-1886. DOI: 10.1007/s10291-018-0736-8. [Online]. Available: https://doi.org/10.1007/s10291-018-0736-8.

[209] A. Minetto, F. Dovis, A. Vesco, M. Garcia-Fernandez, À. López-Cruces, J. L. Trigo, M. Molina, A. Pérez-Conesa, J. Gáñez-Fernández, G. Seco-Granados, *et al.*, "A testbed for GNSS-based positioning and navigation technologies in smart cities: The HANSEL project", *Smart Cities*, vol. 3, no. 4, pp. 1219–1241, 2020.

[210] A. Minetto, M. C. Bello, and F. Dovis, "DGNSS Cooperative Positioning in Mobile Smart Devices: A Proof of Concept", *IEEE Transactions on Vehicular Technology*, vol. 71, no. 4, pp. 3480–3494, 2022. DOI: 10.1109/TVT.2022.3148538.

[211] N. Chukhno, S. Trilles, J. Torres-Sospedra, A. Iera, and G. Araniti, "D2D-based Cooperative Positioning Paradigm for Future Wireless Systems: A Survey", *IEEE Sensors Journal*, 2021.

[212] I. Skog and P. Handel, "In-Car Positioning and Navigation Technologies—A Survey", *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 1, pp. 4–21, 2009. DOI: 10.1109/TITS.2008.2011712.

[213] N. Alam and A. G. Dempster, "Cooperative Positioning for Vehicular Networks: Facts and Future", *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1708–1717, 2013. DOI: 10.1109/TITS.2013.2266339.

[214] M. Tahir, S. S. Afzal, M. S. Chughtai, and K. Ali, "On the Accuracy of Inter-Vehicular Range Measurements Using GNSS Observables in a Cooperative Framework", *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 682–691, 2019. DOI: 10.1109/TITS.2018.2833438.

[215] A. Minetto, A. Gurrieri, and F. Dovis, "A Cognitive Particle Filter for Collaborative DGNSS Positioning", *IEEE Access*, vol. 8, pp. 194 765–194 779, 2020. DOI: 10.1109/ACCESS.2020.3033626.

[216] A. Minetto and F. Dovis, "On the Information Carried by Correlated Collaborative Ranging Measurements for Hybrid Positioning", *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1419–1427, 2020. DOI: 10.1109/TVT.2019.2957015.

[217] S. Zocca, A. Minetto, and F. Dovis, "Adaptive Bayesian State Estimation Integrating Non-stationary DGNSS Inter-Agent Distances", in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–7. DOI: 10.1109/VTC2021-Spring51267.2021.9448952.

[218] S. Ma and H. Lee, "Improving Positioning Accuracy Based on Self-Organizing Map (SOM) and Inter-Vehicular Communication", *Trans. Emerg. Telecommun. Technol.*, vol. 30, no. 9, Sep. 2019, ISSN: 2161-3915. DOI: 10.1002/ett.3733. [Online]. Available: https://doi.org/10.1002/ett.3733.

[219] F. Yin, Z. Lin, Q. Kong, Y. Xu, D. Li, S. Theodoridis, and S. R. Cui, "FedLoc: Federated Learning Framework for Data-Driven Cooperative Localization and Location Data Processing", *IEEE Open Journal of Signal Processing*, vol. 1, pp. 187–215, 2020. DOI: 10.1109/OJSP.2020.3036276.

[220] J. Xiong, J. W. Cheong, Z. Xiong, A. G. Dempster, S. Tian, and R. Wang, "Integrity for Multi-Sensor Cooperative Positioning", *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 792–807, 2021. DOI: 10.1109/TITS.2019.2956936.

[221] K. F. Hasan, Y. Feng, and Y.-C. Tian, "GNSS Time Synchronization in Vehicular Ad-Hoc Networks: Benefits and Feasibility", *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3915–3924, 2018. DOI: 10.1109/TITS.2017.2789291.

[222] H. J. Jo, I. S. Kim, and D. H. Lee, "Reliable Cooperative Authentication for Vehicular Networks", *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1065–1079, 2018. DOI: 10.1109/TITS.2017.2712772.

[223] Y. Hao, Y. Cheng, C. Zhou, and W. Song, "A Distributed Key Management Framework with Cooperative Message Authentication in VANETs", *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 3, pp. 616–629, 2011. DOI: 10.1109/JSAC.2011.110311.

[224] A. Conti, S. Mazuelas, S. Bartoletti, W. C. Lindsey, and M. Z. Win, "Soft Information for Localization-of-Things", *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2240–2264, 2019. DOI: 10.1109/JPROC.2019.2905854.

[225] M. Li and A. I. Mourikis, "Online Temporal Calibration for Camera–IMU Systems: Theory and Algorithms", *Int. J. Rob. Res.*, vol. 33, no. 7, pp. 947–964, Jun. 2014, ISSN: 0278-3649. DOI: 10.1177/0278364913515286. [Online]. Available: https://doi.org/10.1177/0278364913515286.

[226] Y. Guo, O. Vouch, S. Zocca, A. Minetto, and F. Dovis, "Enhanced EKF-based Time Calibration for GNSS/UWB Tight Integration", May 2022. DOI: 10.36227/techrxiv.19823128.v1. [Online]. Available: https://www.techrxiv.org/articles/preprint/Enhanced_EKF-based_Time_Calibration_for_GNSS_UWB_Tight_Integration/19823128.

[227] ETSI, "ETSI TR 103 415 V1.1.1 (2018-04) - Intelligent Transport Systems (ITS); Security; Pre-standardization study on pseudonym change management", en, European Telecommunications Standards Institute, Standard ETSI TR 103 415 V1.1.1, 2018.

[228] H. Shimada, A. Yamaguchi, H. Takada, and K. Sato, "Implementation and Evaluation of Local Dynamic Map in Safety Driving Systems", *Journal of Transportation Technologies*, vol. 05, pp. 102–112, Jan. 2015. DOI: 10.4236/jtts.2015.52010.

[229] A. Rauch, F. Klanner, and K. Dietmayer, "Analysis of V2X communication parameters for the development of a fusion architecture for cooperative perception systems", in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 685–690. DOI: 10.1109/IVS.2011.5940479.

[230] H.-j. Gunther, O. Trauer, and L. Wolf, "The potential of collective perception in vehicular ad-hoc networks", in *ITST 2015*, 2015, pp. 1–5. DOI: 10.1109/ITST.2015.7377190.

[231] P. Sondi, L. Rivoirard, and M. Wahl, "Performance Evaluation of Vehicle-to-Vehicle Communications for a Collective Perception Application in Vehicular Ad hoc Networks", in *IEEE PIMRC 2018*, 2018, pp. 602–603. DOI: 10.1109/PIMRC.2018.8580753.

[232] A. Olmos, F. Vázquez-Gallego, R. Sedar, V. Samoladas, F. Mira, and J. Alonso-Zarate, "An Automotive Cooperative Collision Avoidance Service Based on Mobile Edge Computing", in Sep. 2019, pp. 601–607, ISBN: 978-3-030-31830-7. DOI: 10.1007/978-3-030-31831-4_43.

[233] G. Avino, P. Bande, P. A. Frangoudis, C. Vitale, C. Casetti, C. F. Chiasserini, K. Gebru, A. Ksentini, and G. Zennaro, "A MEC-Based Extended Virtual Sensing for Automotive Services", *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1450–1463, 2019. DOI: 10.1109/TNSM.2019.2931878.

[234] R. Vilalta, R. Casellas, R. Sedar, F. Vázquez-Gallego, R. Martínez, S. K. Datta, M. Lefebvre, F. Gardes, J.-M. Odinot, J. Härri, J. Alonso-Zarate, and R. Muñoz, "Vehicular Message Exchange in Cross-border Scenarios Using Public Cloud Infrastructure", in *IEEE 35GWF 2020*, pp. 251–256. DOI: 10.1109/5GWF49715.2020.9221221.

[235] "5G-CARMEN D5.2, Preliminary Pilot Report", Sep. 2021.

[236] N. Slamnik-Krijestorac, G. M. Yilma, M. Liebsch, F. Z. Yousaf, and J. Marquez-Barja, "Collaborative orchestration of multi-domain edges from a Connected, Cooperative and Automated Mobility (CCAM) perspective", *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021. DOI: 10.1109/TMC.2021.3118058.

[237] "Home page - L3 Pilot [online]". (2022), [Online]. Available: https://l3pilot.eu.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. 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.