ms-van3t: An integrated multi-stack framework for virtual validation of V2X communication and services

Computer Communications 217 (2024) 70–86

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

ms-van3t: An integrated multi-stack framework for virtual validation of V2X communication and services

F. Raviglione [a,*], C.M. Risma Carletti [b], M. Malinverno [c], C. Casetti [b], C.F. Chiasserini [a]

[a] Politecnico di Torino, Department of Electronics and Telecommunications (DET), C.so Duca degli Abruzzi 24, Turin, 10129, Italy
[b] Politecnico di Torino, Department of Control and Computer Engineering (DAUIN), C.so Duca degli Abruzzi 24, Turin, 10129, Italy
[c] Italdesign Giugiaro, Via Achille Grandi, 25, Moncalieri, 10024, Italy

ARTICLE INFO

Keywords:
Connected vehicles
V2X
Virtual validation
Hardware in the loop

ABSTRACT

The automotive field is evolving towards high levels of automation, requiring seamless data exchange between vehicles through Vehicle-to-Everything (V2X) communications. Direct V2X technology is already being deployed on commercial vehicles, and it has the potential to deliver a range of safety and efficiency benefits on the road. However, the deployment of V2X-based applications is a complex process that demands extensive testing before these systems can be widely used by the public; indeed, high costs and safety concerns are among the main hurdles to overcome before applications leveraging V2X communication can become a reality. It is thus critical to reliably validate through simulation and emulation both the V2X technologies and the applications in realistic scenarios, before performing large-scale road tests. To address this pressing need, we present an open source framework for the virtual validation of V2X-based applications, amenable to the development and testing not only of different access technologies within the same environment (IEEE 802.11p, LTE-V2X, 5G NR-V2X, and LTE), but also of any kind of V2X-based application using ETSI-compliant messages. Our framework, called ms-van3t, is based on the ns-3 and SUMO (Simulation of Urban MObility) simulators, it implements a full ETSI C-ITS stack for CAM, DENM and IVIM messages, and it provides several novel features not found elsewhere. Further, ms-van3t enables the testing of V2X-based applications in HIL (Hardware-In-the-Loop) scenarios, thanks to a dedicated emulation mode, and it allows users to easily select different physical and MAC layer models, seamlessly collecting performance statistics. To showcase the capabilities of the framework, we present three sample applications as well as the performance results we obtained in terms of both application-related and network-related key performance indicators.

1. Introduction

The automotive sector is recently undergoing a significant evolution towards safer, more efficient, greener vehicles. One of the most groundbreaking technologies in this context is represented by Vehicle-to-Everything (V2X) communications, i.e., the possibility of connecting vehicles between themselves as well as with the road and network infrastructure.

The analysis and deployment of V2X technologies, followed by the development of suitable applications to improve, for instance, road safety, is however a complex process. This is evident when an application needs to be tested and analyzed on large fleets of vehicles, which would imply non-negligible costs. Moreover, 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's internal logic. It is thus mandatory to leverage efficient simulation and emulation tools, to support the development of applications and the deployment of the underlying physical and access layer technologies.

This paper aims at addressing the above issues by introducing a novel V2X tool for virtual validation, named ms-van3t (Multi-Stack VANET framework for ns-3). ms-van3t, drawing on the well-established ns-3 and SUMO simulators and supporting the integration of hardware on the loop, manages both mobility and connectivity of vehicles.

Importantly, ms-van3t integrates several open-source, state-of-the-art 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 our knowledge, this is a feature that cannot yet be found in any other existing simulation tools with the same wide support for current and emerging V2X technologies. Further, ms-van3t provides a full ETSI (European Telecommunication

* Corresponding author.
E-mail addresses: francesco.raviglione@polito.it (F. Raviglione), carlos.rismacarletti@polito.it (C.M.R. Carletti), marco.malinverno@polito.it (M. Malinverno), claudio.casetti@polito.it (C. Casetti), carla.chiasserini@polito.it (C.F. Chiasserini).

https://doi.org/10.1016/j.comcom.2024.01.022
Received 15 November 2022; Received in revised form 2 June 2023; Accepted 22 January 2024
Available online 29 January 2024

Standard Institute) C-ITS stack, implemented from scratch, to best leverage the capabilities offered by ns-3, supporting CAM (Cooperative Awareness Messages), with the status and attribute information of vehicles [1], event-triggered DENM (Decentralized Environmental Notification Message) messages [2], and road signage information IVIMs (Infrastructure to Vehicle Information Messages) [3]. To the best of our knowledge, the latter is not found in any other open source simulation and emulation framework for V2X.

Additionally, other than simulating large scale scenarios, ms-van3t can emulate the presence of vehicles: once a mobility scenario is set up through SUMO, our framework can send and receive standard ETSI C-ITS messages from/to the external world, through a physical interface, enabling Hardware-in-the-Loop (HIL) testing of V2X applications and communications with real vehicles. Finally, ms-van3t offers the possibility to:

1. use real GNSS traces for the mobile nodes, with real GNSS errors, instead of relying solely on synthetic scenarios generated, e.g., using SUMO;
2. easily visualize the mobile nodes on real maps, thanks to a web-based visualizer, which can also be updated thanks to data coming from real vehicles and received in emulation mode;
3. efficiently compare different technologies, involving up to thousands of nodes to understand, e.g., the advantages and disadvantages of IEEE 802.11p versus Cellular-V2X (C-V2X, i.e., either LTE-V2X or NR-V2X) under different scenarios;
4. perform very large scale simulations, up to thousands of vehicles in simulation mode, and up to tens of vehicles in real-time emulation mode;
5. switch between ETSI C-ITS vehicular message versions thanks to an easy-to-use script;
6. rely on a fully open-source and up-to-date repository, available at the public GitHub repository in [4], working on the great majority of Debian-based Linux distributions.

The remainder of the paper is organized as follows. Section 2 discusses related studies, other available frameworks for V2X, and further highlights the novelty of our work. Section 3 introduces the architecture of the simulation and emulation framework we envisioned. A description of how applications can be developed by users in ms-van3t, together with the presentation of three sample applications, is provided in Section 4. Section 5 shows how ms-van3t can be used to investigate application and network-related performance and presents the evaluation of the four access-layer technologies available in ms-van3t. Finally, Section 6 concludes the paper and discusses future evolutions of ms-van3t.

2. Related work and novelty

Among the existing V2X-dedicated network simulators, Veins [5] is surely one of the most established ones. It couples SUMO and the network simulator OMNeT++, via one of the most complete implementations of the so-called TraCI (Traffic Control Interface) [6] interface. Beside its native IEEE WAVE/802.11p implementation, other projects, such as SimuLTE [7] and Simu5G [8], have been merged with Veins to extend its capabilities to the simulation of LTE and 5G networks respectively. Veins has also been extended by the Artery and Vanetza projects [9,10], introducing a fully functional ETSI C-ITS model and a flexible platform for V2X applications prototyping. It is worth mentioning that both Artery and Vanetza are not offered bundled together, and the user is expected to install each OMNet++ library manually in order to use them together with Veins. ms-van3t, as opposed to Veins, has been conceived to integrate both the most relevant ETSI C-ITS protocols and access layer technologies as a self-contained feature including everything needed to quickly develop and simulate V2X applications. Offering a full ETSI C-ITS stack as well, the iTETRIS project [11] relies on ns-3 to be coupled with SUMO, providing IEEE 802.11p, UMTS, WiMAX and DVB-H implementations for its access technologies features. However, ms-van3t is the first project *for ns-3* implementing a full ETSI C-ITS stack, while enabling communication on top of IEEE 802.11p, 3GGP LTE-V2X, LTE, and NR-V2X. Even though the WiLabV2Xsim [12] simulator, developed on MATLAB, offers the same access technologies as ms-van3t, it lacks the ETSI C-ITS stack support and can only leverage pre-recorded mobility traces. Indeed, WiLabV2Xsim is aimed at the analysis of Sidelink resource allocation performance rather than the testing and development of vehicular applications, as is the case of our framework.

It is worth noting that one of the advantages of ns-3, with respect to OMNeT++ and MATLAB, is that it provides an easier transition from coding a simulated application to its actual development for a real deployment into an On-Board Unit. This is due to the fact that ns-3 is UNIX-oriented and uses an event scheduling approach, together with functions with closely resemble Linux system calls (e.g., for sending packets, sockets need to be created and bound in a similar way as it is done in Linux software for V2X). Furthermore, despite OMNeT++-based solutions offer a more complete statistics collection framework, ms-van3t simplifies the collection of relevant V2X metrics under ns-3 thanks to a dedicated module described in Section 3.2.

As opposed to other open-source solutions, ms-van3t enables the use of pre-recorded GNSS traces, which can be used as an alternative to SUMO, thus providing paths based on real GNSS data to the simulated vehicles running the applications under test. Even though WiLabV2Xsim handles pre-recorded traces, it only supports traces recorded in Cartesian format (i.e., x and y) without offering compatibility with traces in geodesic format (i.e., Latitude and Longitude), as recorded by commercial GNSS receivers and supported by our framework. Furthermore, although the platform presented in [15] accounts for localization errors introduced by GNSS receivers, proposing a Gaussian model applied to ground-truth values retrieved from SUMO, it only considers ideal access protocols, neglecting packet collisions, as opposed to ms-van3t which offers the combination of both realistic access control models and real GNSS traces.

As mentioned earlier, ms-van3t includes an additional emulation mode, for V2X application HIL testing. Different emulation solutions for HIL testing can be found in the literature, most of them devoted solely to emulation, such as [14,15], leveraging custom components instead of relying on network simulators. In a similar approach to Veins but aimed towards the US market, the VENTOS project [13] couples SUMO and OMNeT++ with an implementation of the WAVE protocol stack, providing emulation capabilities. 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. An extension to Artery is proposed in [16], to monitor a Device Under Test able to send and receive messages from vehicles in a simulated scenario. The authors also analyze the limitations of OMNeT++ to generate traffic in real-time. However, none of these solutions, based on OMNeT++, specifies the limits in the number of vehicles supported in real-time when emulation mode is enabled. The authors in [17] showcased the capabilities of Simu5G to emulate up to hundreds of UEs and several gNBs by applying simplifications to the standard Simu5G models and limiting the functionalities of *background UEs*. Recently, an extension for the emulation of ETSI MEC entities has been presented in [18] where the mobility model available in the OMNeT++ INET library is used. Although Simu5G supports the integration with Veins for modeling the mobility of the emulated nodes, to the best of our knowledge, large-scale emulation of vehicular use cases have not been tested in the literature. It is worth noting that, despite introducing delays in the emulation process, microscopic mobility tools such as SUMO are needed for the accurate performance assessment of ITS applications. Our framework, instead, is specifically developed for vehicular use cases, offering an inherently embedded API for SUMO and GNSS traces for the mobility of nodes, as well as a state-of-the-art model for NR-V2X in Mode 2 [19], still not included in Simu5G at the time of

Table 1
Comparison of ms-van3t features with available vehicular networks simulation/emulation solutions.

Tool	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 [5]	*X* 802.11p	✓	✓	*X*	*X*	No ETSI stack	n.d.	*X*	✓
Artery [9]	*X* 802.11p	✓	✓	*X*	*X*	CAM, DENM, MAPEM, SPATEM, CPM	*X* (version 2)	*X*	✓
SimuLTE [7]	*X* LTE	✓	*X*	*X*	*X*	No ETSI stack	n.d.	*X*	✓
Simu5G [8]	✓ LTE, 5G, NR-V2X Mode 1	✓	*X*	*X*	✓	No ETSI stack	n.d.	*X*	✓
iTETRIS [11]	✓ 802.11p, WiMAX, UMTS, DVB-H	✓	✓	*X*	*X*	CAM, DENM, LDM	n.d.	*X*	✓
WiLabV2Xsim [12]	✓ 802.11p, LTE-V2X Mode 4, NR-V2X Mode 2	✓	*X*	*X*	*X*	No ETSI stack	n.d.	*X*	✓[a]
VENTOS [13]	*X* 802.11p	✓	✓	*X*	✓	No ETSI stack	n.d.	*X*	✓
RVE [14]	*X* 802.11p	✓	✓	*X*	✓	No ETSI stack	n.d	*X*	*X*
ms-van3t [4]	✓ 802.11p, LTE, Release 14 C-V2X Mode 4, NR-V2X Mode 2	✓	✓	✓	✓	CAM, DENM, IVIM	✓ (version 1/2)	✓	✓

[a] Additionally to the MATLAB version, WiLabV2Xsim offers an Octave-based branch in their repository.

writing. Furthermore, we evaluate the real-time ms-van3t limits taking as reference two CPU models, and provide this information to the user through our GitHub repository [4].

A summary of all the mentioned works is presented in Table 1 comparing their features with the ones present in ms-van3t. The next sections will describe our framework, following the key features showcased in the Table. Finally, a preliminary version of this work has been presented in our conference paper [20]. With respect to [20], this paper adds:

1. a full ETSI stack, including the Networking and Transport layers;
2. a comparison between our framework and other V2X frameworks available in the literature, showcasing the additional features and advantages of ms-van3t and providing an extensive description of its features;
3. the possibility of using ms-van3t as a HIL and emulation tool, together with an extensive validation of emulation mode, showing how it can communicate with a commercial standard-compliant implementation. We also provide a study on how many vehicles ms-van3t can emulate in real-time on different CPU models;
4. the support to IVIMs and a new sample application, namely, the *Traffic Manager*, to showcase how ms-van3t can be used to develop and test not only safety related applications, but also use cases aimed at improving the traffic conditions in urban scenarios;
5. the inclusion, within ms-van3t, of a 5G NR-V2X model [19], focused on the so-called *Mode 2*, working without support from the infrastructure;
6. A set of results comparing the various access-layer technologies available within ms-van3t, showing how our framework can be leveraged to gather network-related performance metrics and providing useful insights coming from the comparison of IEEE 802.11p, Release 14 LTE-V2X, LTE, and Release 16 NR-V2X.

3. Features and architecture

ms-van3t is composed of different software modules, interacting between themselves through dedicated interfaces, providing a powerful platform for V2X application testing. As mentioned, it draws on SUMO [21] (v1.9.2 at the time of writing), an open-source urban mobility simulator which is used to dynamically model the position of the simulated/emulated nodes, and ns-3 [22] (v3.33 at the time of writing), a modular discrete-event simulator. In particular, ms-van3t enhances ns-3 with the 3GPP Release 16 5G NR-V2X, the ETSI C-ITS stack as well as additional novel features and modules, which can be integrated as libraries thanks to an easy-to-use installation script. The full architecture of ms-van3t is depicted in Fig. 1.

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). Further, as detailed later, the vehicle mobility can be managed either through SUMO, representing the default simulation mode, or thanks to pre-recorded GNSS traces. Finally, ms-van3t provides a flexible web-based vehicle visualizer, which can display vehicles on a map, and which can be particularly useful when simulating a scenario based on GNSS traces.

When compared to other solutions available in the literature and on the market, ms-van3t can jointly provide several additional features without requiring additional extensions. While other solutions available in the literature and on the market offer a subset of the features described in this section, ms-van3t distinguishes itself as a solution encompassing the entire set of such features as part of its core functionality, without requiring any additional installation or plug-in. Below we describe in detail the capabilities used as comparison reference categories in the header of Table 1 and further illustrated as part of the full ms-van3t architecture in Fig. 1.

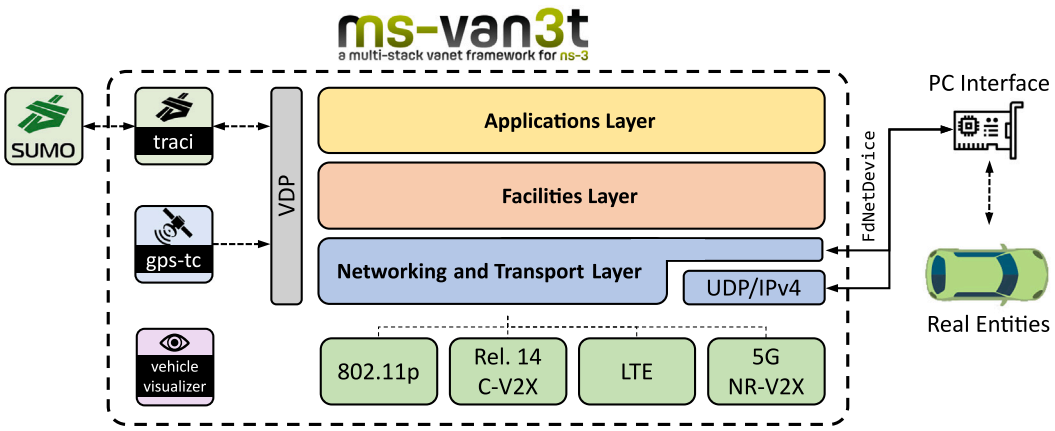


Fig. 1. Full architecture of ms-van3t, comprising simulation and emulation capabilities, and including four different state-of-the-art access layer models.

3.1. Multi-stack

The simulation environment of ms-van3t is designed to provide a high-level abstraction layer, where the vehicular applications can be developed and built 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 users willing to focus only on application-related aspects can be freed from the burden of writing low-level code. The lower layers are indeed almost transparently provided to the end-user, focusing on supporting multiple technologies for the evaluation and deployment of vehicular applications. As of now, ms-van3t enables the simulation of vehicular scenarios using four different access technologies, both well-established and emerging, thanks to state-of-the-art models:

1. *IEEE 802.11p*. When using this model, vehicles are equipped with IEEE 802.11p-compliant On Board Units (OBUs) enabling vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communications: in the first case, OBUs are used to directly exchange messages generated and consumed by vehicles, while in the second case one or more Road Side Units (RSUs) are equipped with IEEE 802.11p interfaces to receive the messages generated by vehicles, and to run centralized applications. In this case, the network entities can broadcast their messages and fully exploit all the features introduced in the ETSI Basic Transport Protocol (BTP) [23] and GeoNetworking protocols [24].
2. *3GPP LTE*. In this case, each simulated vehicle is equipped with a User Equipment (UE), which is connected through the *Uu* interface to an eNB. This access technology is used to model vehicle-to-network (V2N) scenarios, since the centralized application is placed in servers that are connected to the Evolved Packet Core (EPC), acting as remote hosts in the LTE network. Due to the need of complying 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.
3. *3GPP Release 14 LTE-V2X*, i.e., the LTE-based 3GPP vehicular communication protocol. When using this model, proposed in [25], vehicles are equipped with *PC5* interfaces configured in Transmission Mode 4. The vehicles can thus broadcast messages directly to their peers without relying on eNB arbitration. The resulting model represents a V2V out-of-coverage communication scenario, where BTP and GeoNetworking can be fully exploited to deliver V2X messages. This model has been integrated to enable an easy comparison with IEEE 802.11p, when deploying V2V-based applications. Furthermore, the *PC5* interface can also be used to communicate with RSUs equipped with the same technology, realizing a V2I communication.
4. *3GPP Release 16 5G NR-V2X*. In this model, presented in [19] as an extension of the 5G-LENA simulator, vehicles are equipped with Sidelink *PC5* interfaces, configured in Transmission Mode 2. Similarly to the LTE-V2X model, vehicles perform the Sidelink resource allocation in a decentralized fashion, for out-of-coverage scenarios. This particular model, in addition to implementing a sensing-based Semi-Persistent Scheduling (SPS) for resource allocation as in LTE-V2X, also offers a random resource selection implementation, introduced in 3GPP Release 17. Leveraging all the benefits of BTP and GeoNetworking, the addition of this model further extends the capability of the framework to compare access technologies intended for distributed V2V applications.

Finally, it is worth mentioning how 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. As future work, 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 LTE-V2X (or NR-V2X) interfering with each other.

3.2. Large scale simulations

One of the main features of ms-van3t is represented by its simulation environment, enabling the testing of large scale vehicular communication scenarios. As mentioned earlier, it is indeed crucial to evaluate any V2X application in simulation, before the actual deployment, as equipping a large amount of vehicles would imply significant costs. Specifically, ms-van3t supports the simulation of any arbitrary number of vehicles, limited only by the hardware capabilities of the device in which the framework is run.

One issue typically arising when a large number of simulated vehicles is involved is related to data collection. Indeed, it is not trivial to gather the desired metrics for a significant amount of nodes, to be later post-processed to obtain system-wide statistics. To tackle this challenge, ms-van3t provides a special module, called `MetricSupervisor`, enabling an automatic, transparent collection of two important metrics[1]: *(i)* the Packet Reception Ratio (PRR), as foreseen by 3GPP TR 36.885 [26], indicating how many packets are lost due to collisions and harsh propagation conditions, and *(ii)* the one-way latency between the transmission of a packet and its reception by nearby vehicles. Specifically, latency is computed every time a nearby vehicle receives a

[1] These metrics are provided at the end of each simulation as average values and can be easily saved to CSV files, reducing the time needed to process the collected data.

message, and then averaged over all received packets during the whole simulation time. The `MetricSupervisor` can be easily extended to support other metrics as well, such as the Packet Inter-Reception (PIR), which is defined by 3GPP as the time which elapses between two consecutive receptions of two different packets between the same couple of nodes [26] and is especially useful to evaluate the jitter affecting periodic messages. Finally, this module works transparently with respect to the underlying access technology to facilitate the comparison between different protocols.

3.3. Native integration with SUMO

ms-van3t leverages a native integration with the SUMO simulator [21], which provides vehicle mobility models and can emulate a wide range of realistic urban and highway scenarios. Each vehicle traveling in the simulated scenario is then equipped with a full ETSI C-ITS stack, thanks to the integration with ns-3, which provides each node with network connectivity, and on which V2X applications can be developed.

This feature is enabled thanks to the `traci` module, adapted from [27], which implements the so-called *TraCI* interface. The solution implemented in ms-van3t thus offers a bidirectional coupling between ns-3 and SUMO, with functions that can be used either to retrieve information from SUMO (e.g., vehicle position, velocity, and acceleration), or to directly control the vehicle dynamics from ns-3. This enables the realization of complex scenarios, in which connected vehicles can be actively controlled to simulate the presence of partially, or fully, autonomous vehicles.

In addition, the TraCI implementation of ms-van3t fully supports Vulnerable Road Users (VRUs, such as pedestrians), which can be simulated in SUMO and made part of complex scenarios with both connected vehicles and connected and non-connected VRUs. It should be mentioned that, at the time of writing, VRU Awareness Messages (VAMs) are a recent addition to ETSI standards [28], and will be introduced in ms-van3t in the near future.

3.4. Pre-recorded GNSS traces

To the best of our knowledge, a feature which is not found in other available frameworks is the possibility of relying, in addition to SUMO positioning, on pre-recorded GNSS traces. Few works have developed solutions leveraging real GNSS traces using the now outdated ns-2 simulator [29,30], missing support for most state-of-the-art technology models, and not offering SUMO compatibility. Instead, our framework enables working either with SUMO or pre-recorded GNSS traces for the simulated vehicles mobility, without major simulation configuration changes. This feature is enabled by the novel `gps-tc` module, which removes SUMO from the loop, providing classes and functions to leverage offline-collected GNSS traces. This module can come in handy when the application being tested and developed in ms-van3t requires realistic positioning data rather than artificial traces created with SUMO.

As opposed to the usage of SUMO, this feature enables testing the behavior of V2X applications in presence of real GNSS errors. These can be non-negligible especially in urban canyon scenarios [31]. It is also worth underlining that these errors are usually neglected by most literature works leveraging frameworks using SUMO only [5,7–9,11–13]. The user can provide as input any CSV GNSS trace containing at least the following information: *(i)* vehicle ID, *(ii)* timestamp in seconds since any moment in time taken as reference, *(iii)* latitude, *(iv)* longitude, *(v)* speed, *(vi)* heading with respect to the North and, optionally, *(vii)* acceleration. The names of the accepted CSV fields can be customized, reducing the burden of adapting different file formats recorded by different GNSS devices.

The `gps-tc` module, written in C++, which is also the main programming language of ms-van3t, provides a simple helper object, called `GPSTraceClientHelper`, to let the user read CSV files in different formats. Thanks to a method of this object (`createTraceClientsFromCSV`), traces can be easily imported, and each simulated vehicle can be assigned a GPS Trace Client (up to the total number of vehicles available in the CSV file), corresponding to the mobility model based on the trace for that vehicle. When the simulation is started, the `playTrace()` method can be used to start reproducing the traces, making vehicles move either all at the same time, or delaying some of the traces. This is done internally thanks to the scheduling of "position update" events at the right times, based on the timestamps included in the trace.

When a CSV file is being imported, one of the major challenges is represented by coordinate conversions. Indeed, GNSS receivers usually report Position-Velocity-Time (PVT) data with Geodetic coordinates (i.e., Latitude and Longitude), while the ns-3 mobility models expect Cartesian coordinates (i.e., x and y). It is thus necessary to rely on coordinate projections, trying to reduce at the same time the overall projection error on the simulated scenario. A possibility would be to use the Universal Transverse Mercator (UTM) projection. However, this poses a significant challenge when traces are recorded on the boundaries between different UTM zones. Therefore, `gps-tc` adopts an accurate Transverse Mercator (TM) projection [32] considering as central meridian (`lon0`) the average longitude of all the vehicles, averaged over all the timestamps in the trace. This avoids the subdivision into zones, and, at the same time, tries to reduce the projection error (since along `lon0` the distortion is null, and it increases as the simulation moves away from the central meridian).

In order to implement the coordinate conversion in an efficient, lightweight manner, without the need of additional pre-requisite libraries, we imported into ms-van3t the code of a few functions dedicated to TM projections from GeographicLib, a set of open-source C++ libraries for geodesic calculations [33].

It is also worth mentioning how, since not all GNSS receivers can provide acceleration values, `gps-tc` can also compute such values starting from the speed of each vehicle at each time instant. This feature assumes a constant acceleration between consecutive points, and it is thus more accurate with high frequency traces (e.g., more than 10 Hz). Furthermore, other than supporting different CSV formats, `gps-tc` can also support different timestamp formats. These formats include all the standard ones supported by the C++ function `std::get_time()`,[2] plus additional formats specifying milliseconds.

Both TraCI and `gps-tc` act as Vehicle Data Providers (VDP), according to the ETSI standards, for the network stack layers implemented in ms-van3t. 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 [1].

After the implementation of the GNSS feature, we were able to validate its correctness by using two datasets of vehicular traces, and verifying the proper positioning of the ns-3 nodes thanks to both the simulation output and the web-based visualizer presented in Section 3.8.

The first set of mobility traces was recorded with a commercial Android smartphone, thanks to the *Ultra GPS Logger* application, and it is now available as part of the ms-van3t repository as a sample set of traces with low (1 Hz) update rate and 2 vehicles (`BiellaTrace.csv`).

The second set is instead represented by the SAMARCANDA dataset [34], providing the traces of 19 vehicles traveling in an area near Turin, Italy, with a fairly high (10 Hz) update rate. This dataset, in conjunction with the pre-recorded GNSS feature, has been used to test a new protocol for V2X in [34].

[2] https://en.cppreference.com/w/cpp/io/manip/get_time.

3.5. Emulation mode

Not only does ms-van3t provide a simulation environment, but it also enables emulation and communication with external entities, such as real vehicles. This allows the user to perform HIL testing of V2X applications, and to create hybrid scenarios in which emulated vehicles (through SUMO or through pre-recorded GNSS traces, via 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. This paves the way to the testing of a real V2X application behavior when installed on several nodes, without facing the cost of equipping more than one real vehicle. The architecture of ms-van3t, including its emulation mode towards a physical interface, is depicted in Fig. 1. The emulation mode relies 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 protocol stack described in the next section, ms-van3t enables the transmission and reception of CAMs, DENMs, and IVIMs from real vehicles in the simulated scenario and vice versa. Our framework currently supports two modes of operation, when emulating a scenario:

- 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, ready to be sent to a server on the Internet or residing within an RSU; notice that this is one of the communication profiles foreseen for infrastructure services by the ETSI standards, under the name of Communication Parameter Setting (CPS) 003 [3].

In both cases, ms-van3t can receive standard-compliant messages from the same physical interface. To showcase the emulation capabilities of our framework, we also created an ad-hoc sample application, namely, the `v2x-emulator`, with vehicles traveling on a real-world road layout and broadcasting CAMs and DENMs on a target interface, or targeting a remote server. The sample application is available in our repository [4] and can be used as a baseline to develop more complex emulation scenarios.

After presenting the emulation features of ms-van3t, we also showcase a significant use case which leverage our framework to perform validation of a real-world vehicular service. This service is represented by the Server Local Dynamic Map (S-LDM), a highly-efficient and centralized, Multi-access Edge Computing (MEC) LDM, proposed in [35]. The S-LDM is amenable to being deployed on one or more Multi-Access Edge Computing (MEC) platforms to collect messages from vehicles (and, possibly, also from road operators), and create an up-to-date dynamic map of the road, comprising information on all road users within a given area. The received information is pre-processed and stored in a custom database, and a subset of data is provided to other MEC services when needed (for instance, when an automated maneuver has to be managed). As it would be very expensive to perform an initial evaluation of this component with a large number of real vehicles, we rely on ms-van3t in emulation mode to emulate an increasing number of vehicles sending CAMs to a MEC platform where the S-LDM was deployed (the results, omitted here for brevity, can be found in our conference paper [35]). To do so, ms-van3t is configured to send messages in UDP mode towards a local server, packing them into AMQP 1.0, which is the protocol of choice for sending data to the S-LDM. The AMQP messages are then transferred to the MEC platform through cabled Internet connectivity. Thanks to this setup, the behavior of the service can be studied as if a large number of real vehicles were actually traveling on a given stretch of road, sending standard-compliant CAMs.

3.6. ETSI C-ITS stack

With the aim of supporting standard-compliant simulations with vehicles exchanging messages according to the ETSI standards, ms-van3t includes a full-fledged ETSI C-ITS implementation, which is better detailed below. To the best of our knowledge, it is the first framework integrating such a stack into ns-3, thanks to the new `automotive` module. This component contains the whole ETSI C-ITS stack, including the logic of the applications, the encoding and decoding schemes for vehicular messages, and the networking logic foreseen by ETSI for vehicular scenarios.

It is worth mentioning how in Section 5.3 we present a set of interoperability tests, thanks to the ability of ms-van3t to communicate with real V2X entities, which validate the ETSI C-ITS implementation presented in this Section.

The ETSI C-ITS set of standards for vehicular communications [36] specifies a complete architecture and network stack for the exchange of messages between vehicles, and between vehicles and infrastructure. The most used ETSI-compliant V2X messages are CAMs and DENMs, which are fully supported by ms-van3t. However, our framework is not limited to the encoding and decoding of these messages, rather, it implements a full ITS stack, comprehensive of the Networking and Transport layers, and of the so-called Facilities layer, in charge of supporting the transmission and reception of application messages. As far as the lower layers are concerned, one of the most interesting aspects of C-ITS, exploited in ms-van3t, is its compatibility with different access layer technologies, including both IEEE 802.11p and C-V2X. Therefore, the upper layers can be encapsulated into the MAC and physical layers of the state-of-the-art models described earlier. The ETSI C-ITS implementation is briefly described below.

3.6.1. Facilities layer

The Facilities Layer is a middleware composed of multiple services providing, according to the standard, *functions, information or services to ITS applications*. ms-van3t focuses, for the time being, on the Cooperative Awareness (CA) and Decentralized Environmental Notification (DEN) basic services, acting as a middle layer between Application and Transport, and managing the transmission, reception, and processing of CAMs and event-triggered DENMs, following the ETSI standards. ms-van3t also includes an Infrastructure to Vehicle Information (IVI) service, managing the transmission of IVIMs from infrastructure nodes to vehicles. These messages typically carry road signage information and can be used, for instance, to enforce speed limits both in urban and highway scenarios. As mandated by the standard, all these messages are encoded and decoded using ASN.1 (Abstract Syntax Notation One), a platform-agnostic and programming language-agnostic way to describe complex data structures, such as the vehicular messages. Starting from an ASN.1 description file, it is possible to automatically generate encoding and decoding functions, thanks to open-source tools like `asn1c` [37]. The great advantage of this approach is that the generated code is not specific for the simulation framework, and can thus be re-used in standalone implementations. This makes it easier to shift from the simulated world to standalone deployments (e.g., on OBUs), also thanks to ns-3 often adopting functions similar to Linux system calls.

Using the generated code as a base, we developed all the code for the basic services mentioned earlier, making them accessible to the user thanks to easy-to-use functions, such as:

- Functions to start and stop the dissemination of CAMs;
- Functions to fill-in the content of DENMs, set the destination geographical area, and trigger/update/cancel their transmission [2]; these include the possibility of setting a repetition interval for periodically sending DENMs, for a given amount of time (e.g., `setDenmRepetitionInterval()`), as foreseen by the ETSI standards;

- Functions to fill-in the content of IVIMs, and manage their transmission [3];
- Functions providing an easy interface to set and configure optional containers in CAMs, DENMs and IVIMs;
- Function to set customized callbacks that are invoked when CAMs, DENMs or IVIMs are received during a simulation or emulation session.

To further reduce the low-level configuration effort for the user, ms-van3t includes the so-called `BSContainer`. A BSContainer, which is a short name for Basic Services Container, is an additional facility that can be installed on vehicles, wrapping multiple Basic Services together, and hiding most of the complexity of the ETSI C-ITS configuration to the user. Since the main programming language of ms-van3t is C++ (as mentioned earlier and in Section 4), BSContainers are provided as C++ objects that can be assigned to simulated vehicles, and then configured with just a couple of lines of code. They also provide additional functions to configure more low-level details, such as whether to use a `MetricsSupervisor` (described in Section 3.2), or whether the simulation is expected to run in real-time (i.e., if ms-van3t is expected to run in emulation mode). This lets users easily set up the ETSI C-ITS stack with default settings, valid for testing all the most common V2X scenarios, leaving at the same time the option to better customize the configuration of the communication stack. After a BSContainer has been configured, it can be used to access the underlying Basic Service, e.g., to start and stop the dissemination of CAMs through the CA Basic Service.

The implementation of this kind of additional facility comes with two major advantages: *(i)* it reduces the time needed from the installation of ms-van3t to the configuration of new V2X simulations, *(ii)* it makes it easier to run and manage simulations even for users not very experienced with the ETSI standards for V2X.

It should also be mentioned that the ms-van3t CA Basic Service implementation includes a custom smart logic for the storage and dissemination of Path History (PH) points in CAMs, inspired by the strategy employed in commercial devices. Indeed, according to the standard [1], the PH strategy can be application and implementation-dependent, with the possibility of storing and transmitting up to 40 historical points [38]. PH points represent the recent time-tagged positioning and dynamic history of a vehicle, and can be transmitted inside the optional low frequency container. Each point is encoded in terms of a delta with respect to the current reference position.

Specifically, our strategy attempts to store and disseminate only relevant PH points, avoiding, for instance, to store and disseminate very nearby points in CAMs if the vehicle is still or moving slowly.

ms-van3t can store and disseminate up to 23 historical points, taking as reference the most restrictive requirement between ETSI [1] and SAE [39]. It then checks if the vehicle has moved by more than 15 m, or if its heading has changed by more than 10 degrees, with respect to the previous CAM. If this is true, the strategy checks whether 23 points have been already stored for dissemination. If yes, the oldest point is removed, and the current position is added to a PH points list, and will be disseminated as the most recent PH point starting from the next CAM. If not, the most recent position is just added as a new point to the list. In case the vehicle dynamics do not satisfy neither the 15 m, nor the 10 degrees threshold, no new points are added as they would be very similar to already disseminated information, and would carry little useful content. With this strategy, it is possible to store and disseminate at least 345 m of path history (23 × 15 m) for each vehicle.

It is worth highlighting that, despite the inclusion of this default strategy for PH points, the modular structure of ms-van3t enables the implementation and integration of other algorithms directly inside the CA Basic Service.

All the generated messages are passed to the Transport Layer, implementing the Basic Transport Protocol (BTP).

The development of a Local Dynamic Map (LDM) is also planned. The LDM is part of the Facilities Layer foreseen by ETSI, and it includes information on road users and other objects perceived thanks to V2X communication. This data can then be provided to active applications [40].

3.6.2. Basic Transport Protocol

BTP enables the Facilities Layer to access the services provided by the GeoNetworking Protocol, facilitating the exchange of control information between the two layers, adding minimal overhead in the process. It is indeed a connectionless protocol, like UDP. As defined in the standard, the BTP module in ms-van3t relies on ports to multiplex and de-multiplex messages coming from the Facilities Layer to the GeoNetworking entity and vice versa.

3.6.3. GeoNetworking

GeoNetworking (GN) is the protocol developed for the C-ITS Networking Layer designed to cover all the challenges of ad-hoc communications, and it features "geographical addressing and geographical forwarding" of packets. The GN module developed in ms-van3t implements two main forwarding schemes for the dissemination of CAMs, DENMs and IVIMs, respectively Single-Hop Broadcast (SHB) and GeoBroadcast (GBC). The first is used to broadcast CAMs only to the first reachable hop, while the second targets the transmission of DENMs to a specific Geographical Area (GeoArea), specified by the DEN Basic Service. All the network management functionalities needed for the transmission and reception of the different GN packets (e.g., address configuration, ego position vector update and beaconing) rely on the geographical information coming from a Vehicle Data Provider. The VDP can be either linked to the TraCI interface or to GNSS traces via the gps-tc module, as shown in Fig. 1.

3.7. Easy switch between ETSI versions

The standardization activities carried on by ETSI led to the definition of multiple versions of standard-compliant vehicular messages. Typically, new versions are intended to supersede older ones, and provide slight updates to some of the fields included in the respective messages. It is thus always recommended to refer to the latest versions of the standards. However, this may not always be feasible in the automotive field, where the adoption of a new standard may take a significant amount of time, due to additional testing efforts which are needed after each update, and possible interoperability issues which may arise. Therefore, it could be of high interest to simulate and emulate vehicular applications with different message versions, depending on the application requirements and the scenario, and possibly comparing them.

Concerning CAMs and DENMs, two versions are currently defined: CAMs version 1 [41] and 2 [1], and DENMs version 1 [42] and 2 [2]. These versions can be seamlessly switched thanks to a dedicated script included in ms-van3t, enabling simulations with either versions 1 or 2. To the best of our knowledge, this feature is available only within ms-van3t, as opposed to Artery, which relies on version 2 messages only [9].

3.8. Web-based visualizer

Among its capabilities, ms-van3t provides a flexible Graphical User Interface (GUI), in addition to the one available within SUMO. This feature is enabled by the `vehicle-visualizer`, a module implementing a web-based visualizer displaying vehicles on a map, particularly useful for demonstration purposes, especially when SUMO and its GUI are not used in the simulation/emulation loop.

Indeed, when `gps-tc` is used to model the vehicle mobility, the web-based GUI can display the simulated and/or emulated road users on map, based either on OpenStreetMap or on Mapbox (if the user has

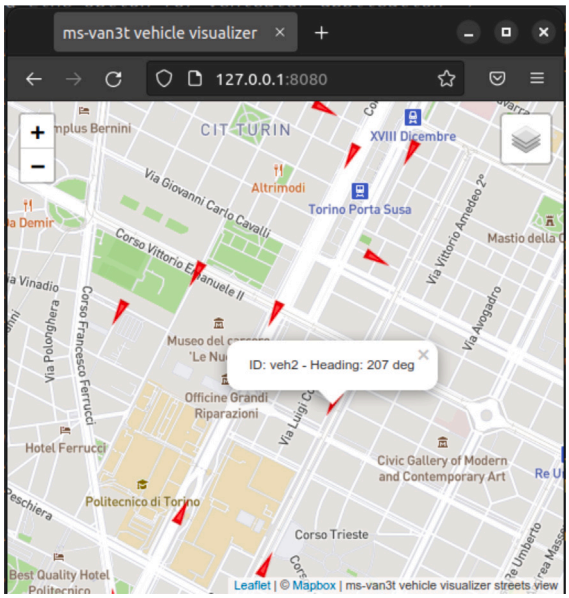


Fig. 2. ms-van3t web-based visualizer with a simulation of vehicles traveling across the city of Turin, Italy.

registered to their services and provides a proper token to ms-van3t). As soon as the simulation starts, the GUI can be accessed at the localhost IP address from any browser. As depicted in Fig. 2, it provides a set of interactive features to display, for instance, the heading and IDs of selected vehicles.

Furthermore, the web-based interface can also be updated, when in emulation mode, with data coming from real vehicles. For instance, if ms-van3t is receiving CAMs from a real vehicle, through a physical interface of the device executing the framework, this road user can be easily displayed on the GUI alongside the other emulated/simulated nodes. This represents a clear advantage over the SUMO GUI, even when SUMO is used to emulate the position of vehicles within ms-van3t. Indeed, external entities communicating with our framework cannot be easily inserted in the SUMO interface, as opposed to the web-based visualizer.

3.9. Open source

ms-van3t has been released with a fully open source license, namely the GNU General Public License (GPL) version 2. This license allows any user to download and redistribute our framework, both for private and commercial use. The full ms-van3t framework, together with an extensive *README* documentation, is available on GitHub [4].

4. Vehicular application development with ms-van3t

The main idea behind ms-van3t is to create a simulation tool to support users seeking to deploy and test applications in vehicular scenarios. For this reason, ms-van3t is designed to limit as much as possible the mandatory input requested by the user: thanks to its straightforward implementation, one can generate arbitrarily complex scenarios with few steps. As ms-van3t is based on ns-3, simulations are coded and configured in C++ files, which are then compiled for the execution of simulations thanks to the `ns3` command line tool.

In general, a user setting out to run simulations with ms-van3t is required to follow four steps as set forth below:

1. *Define the mobility model*. If SUMO is used, the user can define the simulation scenario, and upload all the XML configuration files in a dedicated folder inside ms-van3t. If offline-collected GNSS traces are used, instead, the user is required to upload them into a dedicated folder inside ms-van3t, and, if needed, to format them as required by the framework (i.e., a highly flexible CSV format).
2. *Define the application logic*. The user needs to define the behavior of the network nodes when sending and receiving V2X messages, in dedicated C++ files. To do so, the Facilities layer of ms-van3t gives the possibility of configuring dedicated callbacks that will be invoked when receiving CAMs, DENMs, and IVIMs, as well as functions to trigger the creation of new DENMs and start/stop the dissemination of CAMs. Dedicated functions are also provided to generate new IVIMs to be transmitted from infrastructure to vehicles.
3. *Select and configure the access technology*. The user needs to configure the simulated nodes to adopt the chosen access technology; depending on the desired configuration, it may be required to set up the main network parameters (e.g., frequency, transmission power, subchannel size, etc.). In this way, the V2X messages generated by the higher layers of ms-van3t will be dispatched using the MAC and Physical models of the selected access technology. This step should be done by creating a main simulation C++ file, which contains the overall simulation logic and parameters, and in which applications are assigned to each vehicle or infrastructure node. ms-van3t provides, as detailed in the next Section, several examples meant to be used as a starting point for creating customized simulation files.
4. *Configure additional utilities and modules*. Optionally, the user can configure the usage of additional modules, such as the `MetricSupervisor` to seamlessly gather average latency and PRR metrics. The latter should be configured in the same file that is also used to set up the access technology parameters.

After creating the simulation files, they can be compiled, executed and debugged with `ns3`.

Although fundamental, the aforementioned steps require the user to define non-trivial aspects and, most importantly, to work with different modules of the framework. Thus, to improve the tool usability, we have developed several applications aimed at showcasing the most important functionalities of ms-van3t. These applications, shortly described in the next sections, can be used as a baseline for developers, and exemplify the main features of ms-van3t.

4.1. Area Speed Advisor

The Area Speed Advisor is a simple, yet effective, application that, leveraging the exchange of CAMs and DENMs, can be used to efficiently enforce speed limits in specific areas. It provides for a centralized entity that supervises the simulated scenario and generates DENMs to regulate the vehicle speed. Fig. 3 depicts an example of the SUMO map of an urban area, together with an artwork representing the application in operation in proximity of a school. In ms-van3t, this application comes in two variants: one modeling a typical V2I scenario, and the other modeling a V2N scenario. In the V2I scenario, the access technology we use is IEEE 802.11p.

Following the steps, described at the beginning of this Section, for the creation of the SUMO map and vehicles routes, the *netedit* tool (included with SUMO) has been tasked with the generation of the necessary XML files needed as input for the SUMO configuration.

For what concerns the application logic, for both V2I and V2N scenarios, two applications have been developed; one is designed to run at the centralized entity (referred to as "server" for both scenarios), sending DENMs, and one to run inside each vehicle, receiving and processing the messages. Concerning the server application in the V2I scenario, a GeoArea is defined to be passed down to the GeoNetworking module so that only the vehicles within that GeoArea will process the DENMs. This area corresponds to a low speed zone, in which

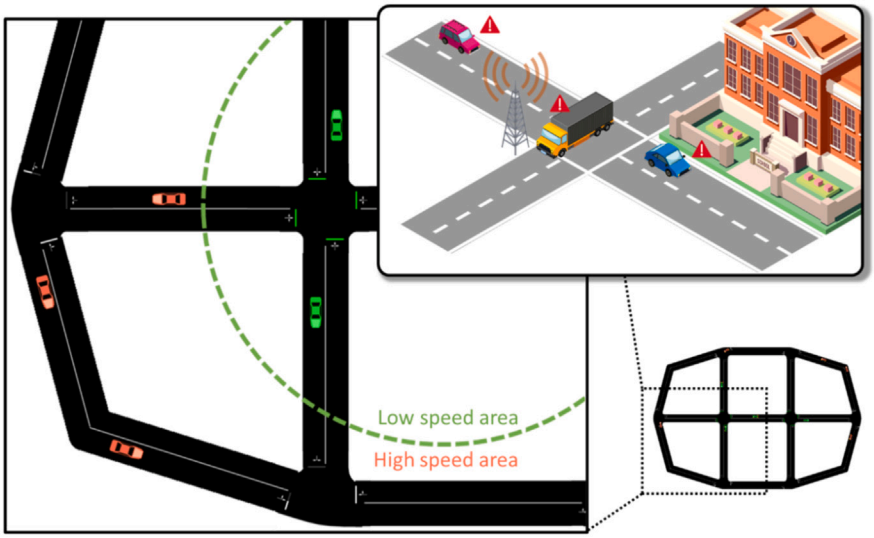


Fig. 3. Example scenario in which the Area Speed Advisor application is deployed in ms-van3t. The SUMO map is composed of a grid topology with 2 intersections. The area with a low-speed limit is highlighted by the green circle.

a speed limit should be enforced. Within the application logic, the CA and DEN Basic Service entities are created for the encoding and decoding of messages, specifying the values to be encoded for the mandatory fields of DENMs (in this case, the *SpeedLimit* field in the *RoadWorksContainerExtended*). Both the CA and DEN Basic Services are instantiated as well for all vehicles in the scenario. When the vehicle-side application receives a DENM, its logic checks the *SpeedLimit* field in DENMs and adjusts the vehicle speed accordingly (thanks to the SUMO TraCI API). Concerning the configuration of the access technology in the V2I scenario, as mentioned earlier, IEEE 802.11p has been selected. Therefore, all nodes (RSU and the OBUs installed on the simulated vehicles) are equipped with IEEE 802.11p *NetDevices* with a configured data rate and transmission power. The RSU is positioned at the center of the map with the server application installed.

As far as vehicle nodes are concerned, SUMO is configured to handle their mobility within the simulation. Specifically, the simulation scenario will install the aforementioned vehicle applications, and configure the CA and DEN Basic Service every time a new vehicle enters the SUMO scenario. It should be mentioned how this operation is carried out by specifying the application and Basic Sevices installation code inside a dedicated callback (called *STARTUP_FCN*), which is automatically invoked by the TraCI API every time a new vehicle enters the scenario.

In the V2N scenario, the selected access technology is LTE, with the application logic running on a server connected to the EPC. In this case, an eNB is placed at the center of the map instead of an RSU, and all vehicle UEs are assigned IP addresses and attached to the eNB. Due to the need of relying on IP addresses when working with on the *Uu* interface of the LTE module, the application logic differs as CAMs and DENMs are now configured to be unicast messages. Similarly to the V2I scenario, the application logic of the simulated vehicles creates the CA and DEN Basic Service entities to start the CAM dissemination towards the server, and, when a DENM is received, it executes the same operations described before, to reduce the vehicle speed if needed. Concerning the application logic at the server, the CA Basic Service is configured to store the IP addresses of incoming CAMs and check the position of the transmitting vehicle (extracted from the messages themselves) to check if it is inside the low speed area. If so, the server application starts scheduling DENMs to be sent (as unicast packets) to that vehicle, to notify it about the lower speed limit. DENMs are periodically sent until a CAM is received with a position outside the area of interest.

4.2. Emergency Vehicle Alert

The Emergency Vehicle Alert (EVA) application improves the ability of emergency vehicles, like ambulances or police cars, to cross an urban area faster, more efficiently and safely while sharing roads with regular vehicles. For the development of the mobility scenario a simple map containing an “interurban road loop” with 2 lanes per direction has been created using *netedit*. Concerning instead the creation of vehicle routes, different speed profiles have been considered for regular vehicles and a special profile has been configured for the Emergency Vehicles (EVs), with a maximum speed of 75 km/h. The latter has been set in the XML file with the vehicle routes. The application logic defined for all vehicles leverages the information carried by CAMs: every time a regular vehicle receives a CAM from a neighboring vehicle, the application reads the sender type field. If this is an emergency vehicle, and if the recipient finds itself to be on the same road segment as the sender (information acquired from the sender’s position in the CAM), it will facilitate the passing maneuver either by reducing its speed, or (in case the regular vehicle is passing other vehicles) by increasing the speed to clear the passing lane as quickly as possible. This application, exemplified in Fig. 4, is used in ms-van3t to demonstrate how to configure and make use of the available V2V communication models, enabled either by 3GPP C-V2X (which includes both LTE-V2X and NR-V2X) or IEEE 802.11p. For the configuration of the application leveraging IEEE 802.11p, the configuration is identical to the one described in Section 4.1 for each of the vehicle OBUs which all run the same application logic. Unlike the previous case, this application is decentralized and no RSU is foreseen. Concerning the configuration of the EVA application with C-V2X, the access technology helper (either the *cv2x_LteHelper* for LTE-V2X or the *NrHelper* for NR-V2X) is used to set up parameters such as antenna models, propagation conditions, path loss models, and frequency settings. As it is necessary for the C-V2X models, an IP is assigned to each node after which Sidelink groups can be created for V2X communication with the definition of Sidelink bearers configurations, including IP addresses and L2 addresses. This step should be coded even if messages, such as CAMs, are then broadcasted and encapsulated into the non-IP-based ETSI C-ITS Networking and Transport layers. The UE sidelink pool parameters are also configured, by specifying the UE-selected resource configuration, subframe configuration, adjacency between Physical Sidelink Control Channel (PSCCH) and Physical Sidelink Shared Channel (PSSCH), sub-channel size, number of subchannels, Resource Block configurations, data transmission power and other additional parameters. Lastly, each

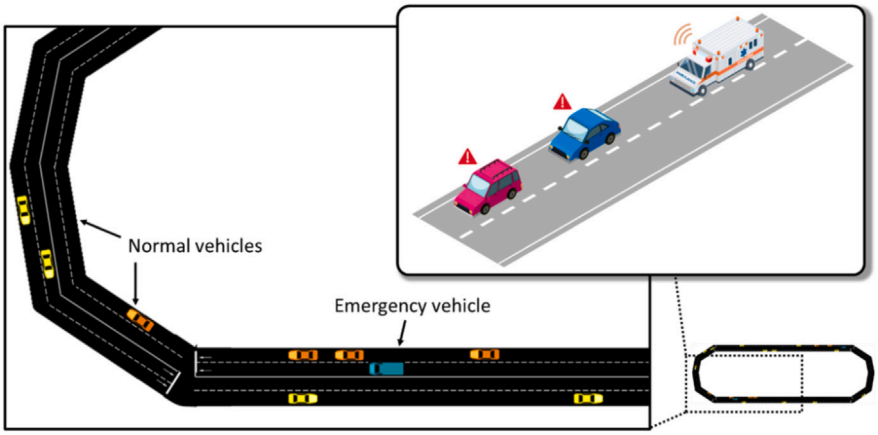


Fig. 4. The scenario in which the Emergency Vehicle Alert application is deployed in ms-van3t. The SUMO map is composed of a ring topology with 2 lanes in each direction. The emergency vehicle is the green vehicle, while all others are regular vehicles. A possible implementation of the Emergency Vehicle Alert application is shown in the overlying image where it is used by an ambulance to signal its presence.

application logic is installed to each simulated vehicle (i.e., to its simulated OBU) every time a new vehicle is spawned in the SUMO scenario.

The EVA application represents a relevant example of how V2X technologies can enable distributed strategies aiming at increasing road safety.

4.3. Traffic Manager

The Traffic Manager application is designed to improve traffic efficiency at road intersections regulated by traffic lights, adapting the length of light phases to the ever-changing traffic flows. For this application, the SUMO map is almost identical to the one used for the Area Speed Advisor application with the only difference being the traffic lights placed in two of the main intersections of the map, as depicted in Fig. 5.

In this scenario, the main application logic is run by a centralized entity that is in charge of controlling the traffic light phases. To perform this task, it collects the information coming from CAMs sent by nearby vehicles and uses it to monitor the traffic load at the intersection. Depending on the traffic load state from the different directions, the application recomputes the phase duration for the next cycle; in case of high unbalance in the monitored traffic, a phase preemption is performed. For all vehicles in this scenario, the application logic only creates an instance of the CA Service to disseminate the CAMs, and no DENMs are foreseen. The centralized entity can be either deployed within an RSU, reachable through IEEE 802.11p, or reside on a MEC server, which can be reached through LTE connectivity. For both access technology configurations, the setup is identical to the one described in Section 4.1.

5. Evaluation and results

As described in the previous sections, ms-van3t includes the models of the most important access-layer technologies for vehicular communications: IEEE 802.11p, LTE, LTE-V2X (Release 14) and NR-V2X (Release 16). In this section, we present some simulation results comparing the available technologies under different aspects, with the aim of highlighting how the proposed framework enables not only the development and testing of V2X applications, but also the evaluation and comparison of the underlying access technologies. We assess the system performance by measuring three key performance metrics: one of them is purely application-related (Section 5.1), while the other two strongly depend upon the adopted access technology (Section 5.2). Further, we also validate the emulation capabilities of our framework, showing how well a relatively large number of vehicles can be emulated in real-time (Section 5.3).

5.1. Application key performance indicators

When developing a vehicular application, in order to measure the application performance, it is important for virtual validation tools to provide the necessary flexibility for gathering the performance metrics of interest. In this section, we provide results for one of the sample applications presented in the previous section, in particular the EVA application described in 4.2. Similarly to the other presented applications, EVA does not rely on a high complexity algorithm; nevertheless, a wide set of application-related metrics can be quickly analyzed when working with ms-van3t to understand potential positive outcomes. We consider two Emergency Vehicles (EVs), each with a maximum speed of 75 km/h, and we gradually increase the number of regular vehicles traveling in the scenario. Then, we analyze the average speed of the EVs when the EVA is enabled, versus the case in which the application is disabled (i.e., vehicles are configured not to react to the CAMs generated by the EVs).

Concerning the simulation parameters, we selected IEEE 802.11p as underlying access technology, working on channel 180 at 5.9 GHz, with a bandwidth of 10 MHz and a physical data rate of 3 Mbit/s (corresponding to a binary phase-shift keying 1/2 modulation). The same settings have been selected for the network key performance indicators presented in Section 5.2, as shown also in Table 2.

It should be highlighted how very similar application-layer results are yielded for C-V2X, and they are thus not reported here to improve the readability of the plots.

The results are plotted in Fig. 6, as the vehicle density varies. Each point is the average over 10 simulations lasting 200 s, with the 95% confidence interval reported as error bars (the same number of simulations has been performed for all the results presented in this section). One can observe that, when the EVA is enabled, the EVs can travel at higher speed; in particular, in jammed scenarios, the speed of EVs under EVA is almost twice the speed without the application in place.

We remark that the evaluation of application performance indicators, proved, on the one hand, that ms-van3t can be an effective tool for the evaluation of innovative V2X applications, based on the exchange of standard-compliant messages. On the other hand, it shows the advantages brought by V2X to future connected and autonomous vehicles.

5.2. Network key performance indicators

After evaluating the application performance, we now focus on the network performance. The main purpose of the results presented herein

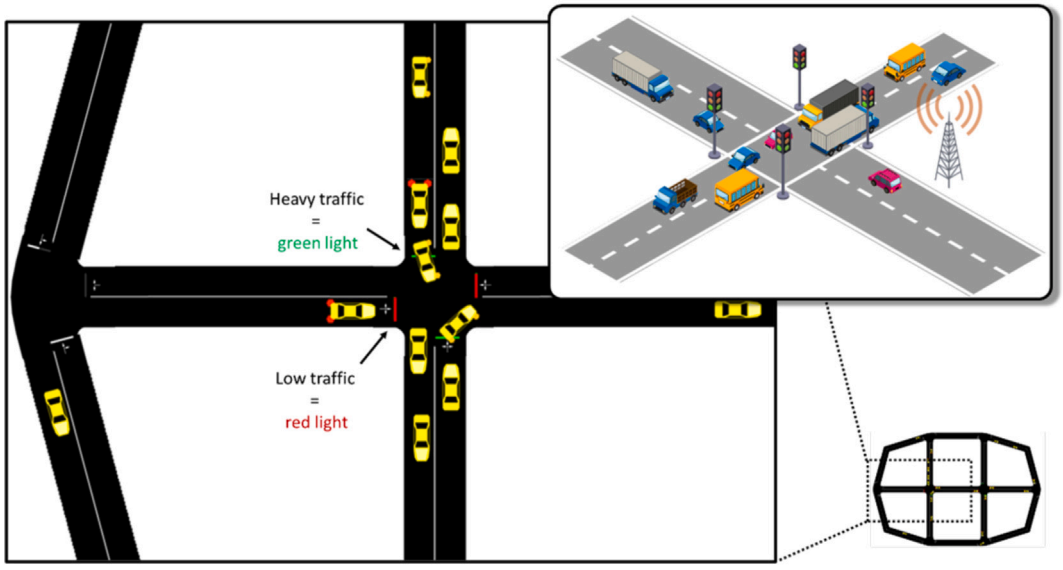


Fig. 5. The scenario in which the Traffic Manager application is deployed in ms-van3t. The SUMO map is composed by a grid topology with 2 intersections. The centralized logic is set to monitor the traffic, and to regulate the traffic lights phases accordingly. The overlying image shows a possible deployment of the application in a regulated intersection.

Table 2
Main simulation parameters for the evaluation of the network KPIs.

Parameter	Value
Number of simulations for each point	10
Duration of each simulation	200 s
Vehicle density	3 veh/km ⟶ 33 veh/km
IEEE 802.11p	
Transmission power	23 dBm, 30 dBm, 33 dBm
Center frequency	5.9 GHz @ 10 MHz
Physical data rate	3 Mbit/s
Propagation Loss Model	WINNER B1 (urban microcell)
LTE-V2X	
Transmission power	23 dBm, 30 dBm, 33 dBm
Center frequency	5.9 GHz @ 10 MHz
MCS	20
Probability of keeping the resources in SPS	0
Resource Reservation Interval	20 ms
Propagation Loss Model	WINNER B1 (urban microcell)
NR-V2X	
Transmission power	23 dBm, 30 dBm, 33 dBm
Center frequency	5.9 GHz @ 10 MHz
MCS	20
Probability of keeping the resources in SPS	0
Resource Reservation Interval	20 ms
Propagation Loss Model	WINNER B1 (urban microcell)
LTE	
Transmission power (eNB)	30 dBm
Transmission power (UE)	10 dBm
Center frequency	2.12 GHz @ 25 MHz (DL), 1.93 GHz @ 25 MHz (UL)
MCS	Adaptive [43]
Propagation Loss Model	Friis

is not to provide an exhaustive comparison of different access technologies but instead to showcase how ms-van3t represents an effective framework for the evaluation of different access technologies, especially in large scale scenarios and during any pre-deployment phase. We therefore present here a comparison among the different technologies available within ms-van3t, taking as reference scenario the one of the EVA application, as depicted in Fig. 4. The evaluated technologies are, respectively, LTE, IEEE 802.11p, Release 14 LTE-V2X Mode 4, and Release 16 NR-V2X Mode 2. To provide comparable results between the technologies using direct device-to-device communications, and LTE, the latter has been configured in such a way that an eNB, placed at the center of the scenario, is connected to a server acting as message relay. Therefore, in the LTE scenario, each V2X message is received by the eNB, which then forwards it through a sequence of unicast packets to all vehicles located within its coverage area.

The evaluation of the different technologies focuses on the one-way latency and PRR performance metrics. We recall that the latter represents a measure of the network reliability, being related to the number of packets lost due to harsh propagation conditions or channel congestion. It should be noted that this metric is commonly referred to as Packet Delivery Ratio (PDR). However, we decided to use the term "PRR" as it is the one used by the 3GPP Technical Reports [26]. Indeed, to compute the PRR, we follow the 3GPP TR 36.885 specifications [44]: given a baseline distance around the sending vehicle (varying between

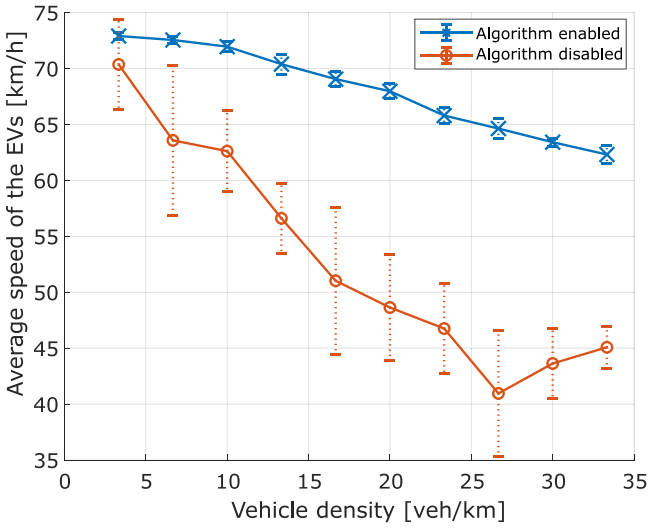


Fig. 6. Average speed of the EV as a function of the vehicles spatial density. The results highlight the effectiveness of the EVA system, showing an increased average speed of the EVs even in case of very high traffic congestion.

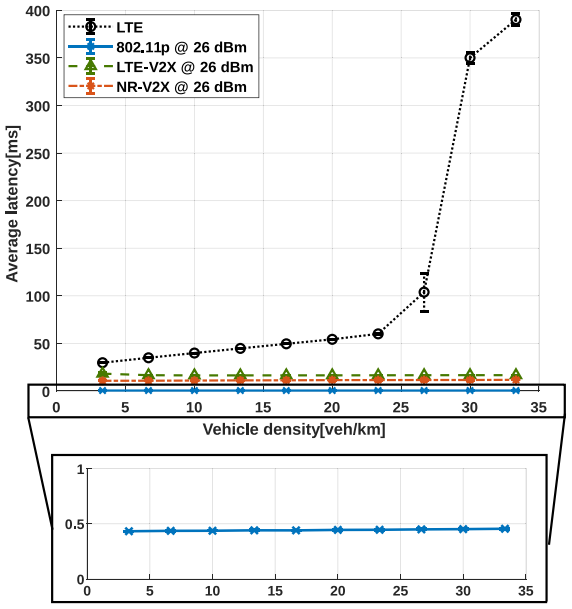


Fig. 7. One-way latency as a function of the vehicle spatial density. IEEE 802.11p, LTE-V2X, and NR-V2X show a stable and low latency. Vehicles communicating with LTE, instead, experience a higher latency, reaching 390 ms under very high traffic congestion.

100 and 200 m), for each message, we calculate the PRR as the ratio between the number of vehicles within the baseline distance that successfully receive the message, X, and the total number of vehicles within the same distance, Y, i.e.,

$$PRR_i = \frac{X_i}{Y_i}, \quad i = 1, \ldots, n_{Tx} \tag{1}$$

Subsequently, we average the above PRR value over all n_{Tx} transmitted messages.

As can be inferred from the above definition, a PRR of 1 means that no packets are lost and the communication is highly stable, while a PRR of 0 means that no communication can be established. ed.

Every metric has been averaged over 10 simulations, each characterized by a different traffic pattern and lasting 200 s. For all simulations, we varied the vehicle density, considering a relatively wide range from 3 veh/km to around 33 veh/km. Additionally, three different power levels have been considered, i.e., 20 dBm, 26 dBm, and 30 dBm, thanks to the capability of our framework to easily tune the OBU or UE transmission power.

Finally, we configure IEEE 802.11p, LTE-V2X, and NR-VRX to work on the Dedicated Short-Range Communications (DSRC) spectrum, at 5.9 GHz, and LTE to work in band 1, at 2.1 GHz. IEEE 802.11p has been configured with a physical data rate of 3 Mbit/s, while LTE-V2X and NR-V2X with a channel bandwidth of 10 MHz, a Resource Reservation Interval (RRI) of 20 ms, a probability of keeping the same resources of 0 and a Modulation and Coding Scheme (MCS) of 20. It should be mentioned that these parameter configurations were among the ones making the selected technologies perform best in the chosen scenario. Table 2 reports a summary of the selected simulation parameters.

The RRI is a crucial parameter for the latency performance of both LTE-V2X and NR-V2X. In the C-V2X Semi-Persistent Scheduling (SPS) mechanism, it represents the periodicity at which a vehicle is expected to reserve the same resources for a number of consecutive transmissions [45]. After a certain number of reservations, these resources are selected again with a certain probability, between 0.2 and 1. The probability of keeping the same resources is thus the inverse of this value, and it can take on values between 0 and 0.8.

The most relevant one-way latency results are presented in Fig. 7, again as a function of the vehicle density. The technology achieving the best performance, with values of latency always around 0.5 ms, is IEEE 802.11p. For what concerns NR-V2X, the obtained latency values are around 10 ms with a slight increase up to 12 ms on very congested scenarios. On the other hand, the values exhibited by LTE-V2X, for almost all vehicle densities, are around 16 ms. LTE, instead, shows substantially worse performance, especially at higher vehicle densities, reaching values above 350 ms. The progressive latency increase in LTE is due to two combined effects: *(i)* the higher latency caused by the need of communicating through the eNB, and *(ii)* the broadcast storm effect due to the relayed traffic, when the density grows high.

Even though we show only the most relevant results for a transmission power of 26 dBm, very similar results were obtained for the other transmission power levels (i.e., 20 dBm and 30 dBm), which are omitted for the sake of brevity.

The average PRR is instead shown in Fig. 8, as a function of the vehicle density for a baseline of 150 m. As can be seen from the plot, following the same trend depicted in Fig. 7, LTE exhibits a significant performance drop for high values of vehicle density. However, for densities up to 23.33 veh/km, LTE instinctively provides the best performance thanks to the higher transmission power of the eNB relaying the messages. Moreover, performance appears relatively stable for all the other access technologies and always providing a PRR above 0.97, with LTE-V2X being slightly more susceptible to higher congestion.

When considering a single vehicle density of 20 veh/km, the PRR that is obtained, as a function of the baseline distance, is depicted in Fig. 9. Here, we also vary the transmission power. The plot suggests that all the cellular-based technologies (i.e., LTE-V2X, NR-V2X, and LTE) can correctly deliver a high number of messages (in excess of 98% for LTE-V2X, and of 99% for LTE and NR-V2X), at a maximum distance of 200 m. When considering a transmission power of 30 dBm, IEEE 802.11p exhibits comparable PRR values to LTE-V2X, slightly under 98%. However, worse results are obtained when considering OBUs transmitting at 26 dBm, with the PRR dropping to almost 80% at a baseline distance of 200 m. It is thus possible to observe how IEEE 802.11p is characterized by a worse PRR for high baseline values (e.g., 200 m), especially for lower transmission power levels. It should be noted how the obtained results are comparable with the ones presented in [46], with IEEE 802.11p providing a slightly lower PRR at given baseline values than LTE-V2X.

It is also possible to observe how IEEE 802.11p is characterized by a worse PRR for high baseline values (e.g., 200 m), especially

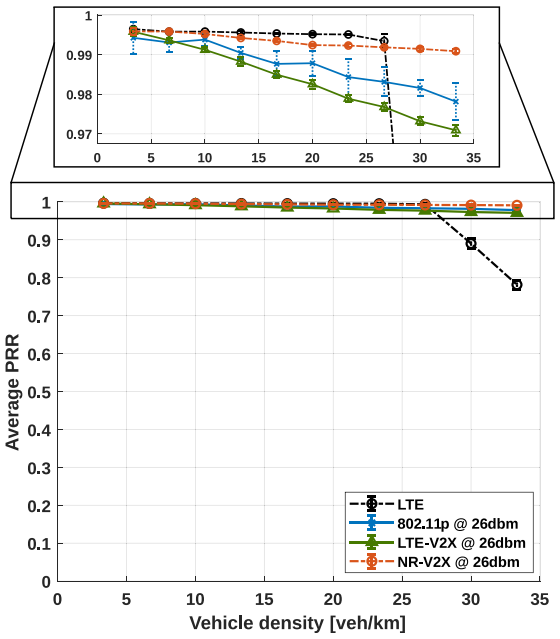


Fig. 8. PRR as a function of the vehicle density, for a baseline distance of 150 m and transmission power level of 26 dBm.

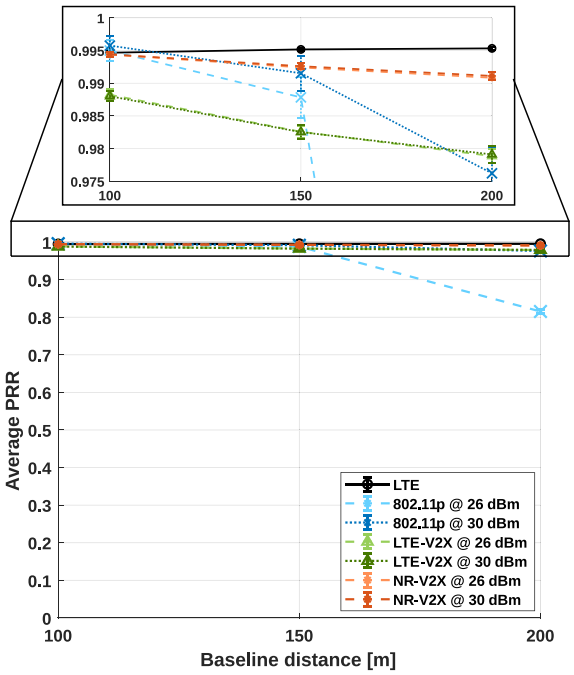


Fig. 9. PRR as a function of the baseline distance from the sender, with at the top the zoom in of the region between 0.97 and 1, where the advantages of using cellular-based technologies are evident. Vehicle density: 20 veh/km.

for transmission power of 26 dBm. This is due to the DSRC-based technology requiring a higher transmission power than LTE-V2X and NR-V2X to reach all vehicles within the baseline. Indeed, it can be observed, from the curves in Fig. 9, that if a sufficient power is used and the baseline is properly set, IEEE 802.11p can deliver more than 99% of packets. It is the case, for instance, of the results for a baseline of 100 m and a transmission power higher than 26 dBm.

It is worth mentioning that the results shown here are strongly dependent on the propagation loss model adopted by the various simulation models: in our case, to enable a fair and realistic comparison between the different technologies, we selected the Friis model for LTE, and the WINNER B1 model (urban microcell, as described in [26]) for all the other technologies (LTE-V2X, NR-V2X, IEEE 802.11p). It should be mentioned that the NR-V2X module comes with a default propagation model based on [47], and that IEEE 802.11p uses as default option a Log-distance model. Furthermore, we kept the default PHY layer settings for the different technologies (e.g., SNR and RSSI sensitivity values), with the exception of the ones concerning the path loss model, which are the same for IEEE 802.11p, LTE-V2X and NR-V2X.

Take-away message. Overall, our results highlight the relevance of such applications as EVA, and point out that IEEE 802.11p is the technology providing the smallest latency. However, from the point of view of PRR, it is outperformed by the cellular-based technologies. Our conclusion is therefore that both LTE-V2X Mode 4 and NR-V2X Mode 2 represent a good tradeoff between latency and PRR. Furthermore, the results prove how NR-V2X performs better than LTE-V2X, being able to guarantee a slightly better PRR, with a lower latency and improved resilience to channel congestion. On the other hand, LTE can provide the highest PRR under low vehicle densities, thanks to the infrastructure-based communication, but it appears to be affected by a noticeable performance drop when the number of vehicles increases. We have also demonstrated how ms-van3t can be an easy-to-use, handy framework for simulating and emulating V2X scenarios. It is to be remarked that, thanks to the `MetricSupervisor` module, it is possible to retrieve (and save to CSV files) performance metrics in a completely transparent way with respect to the selected access-layer technology.

5.3. Validating the emulation feature

To exploit the capability of ms-van3t to communicate with real nodes, several interoperability tests have been performed. Most of them involved the usage of a commercial ETSI C-ITS stack. The selected commercial stack is one of the latest releases of the V2X SDK by Cohda Wireless, typically used in conjunction with their MK5 or MK6 boards. It also comes with a self-contained environment (i.e., a Linux virtual machine containing a vehicle emulator), which can be used to run Cohda V2X applications on a PC before deploying them on real vehicles. Thanks to this environment, we were able to run the Cohda Wireless stack on standard laptops, without the need of deploying all the software on an MK5 board.

Thus, to validate the emulation capabilities of our framework, we configured two laptops in our laboratory: *(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 the switched LAN in our lab. Our experimental setup is depicted in Fig. 10.

Although no wireless channel was established between the two devices, these settings allowed us to fully validate the capability of ms-van3t to communicate with real nodes thanks to standard-compliant messages. It is worth mentioning that replacing the Ethernet connection with a wireless connection represents a seamless operation. As an example, it would be easy to replace the switched LAN with a couple of embedded boards capable of providing IEEE 802.11p connectivity, such as the solution presented in our conference paper [48].

The first set of interoperability tests involved the configuration of a simple scenario with few vehicles in SUMO. We observed that the reception from, as well as the transmission to, the commercial stack, were performed in accordance to ETSI standards. In particular, the virtual vehicle running the Cohda Wireless SDK was able to receive all the emulated messages from the emulated vehicles in ms-van3t, and vice versa. This also served as a validation of the ETSI C-ITS stack

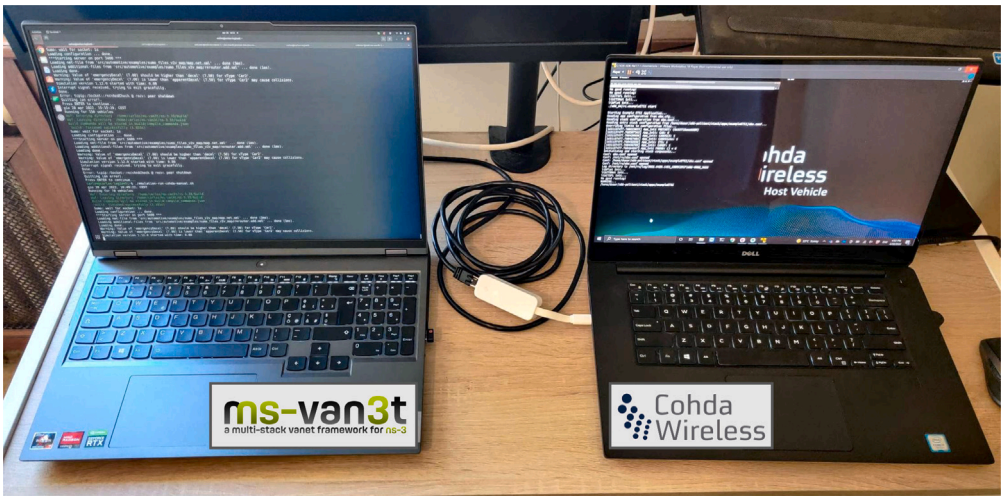


Fig. 10. Laboratory setup for the interoperability tests, performed to validate the emulation feature available in ms-van3t.

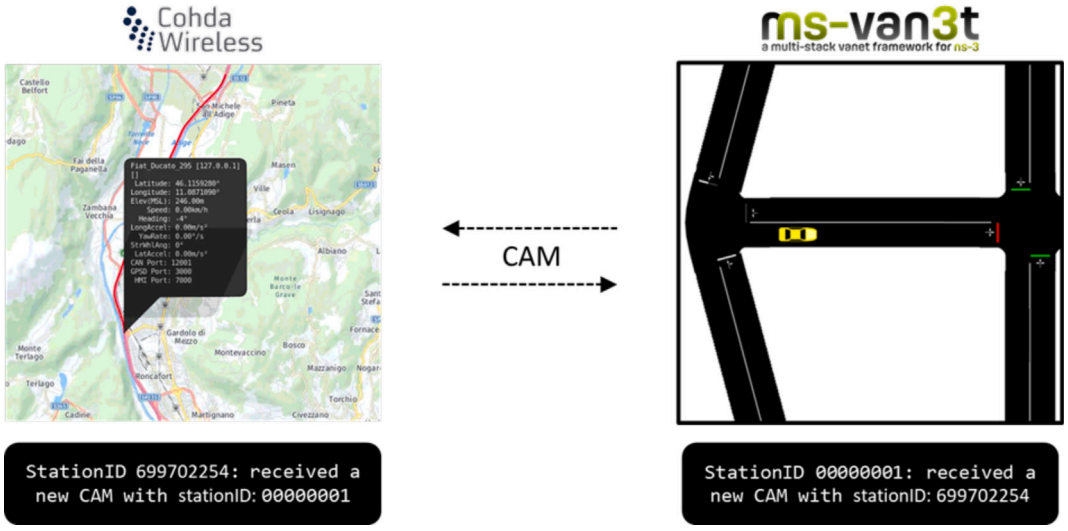


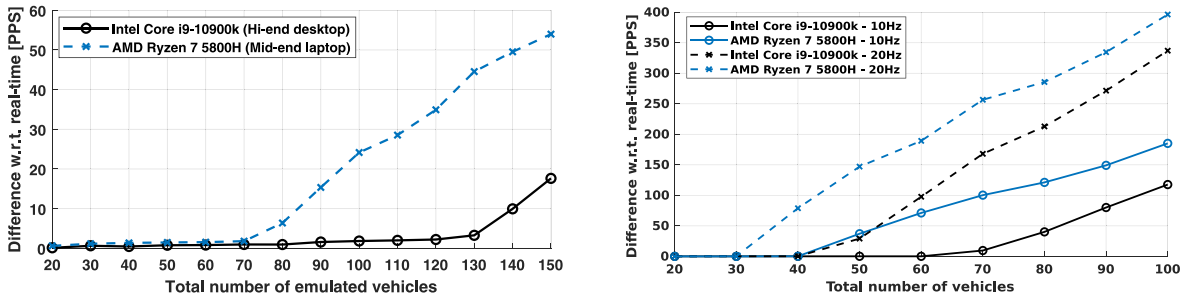
Fig. 11. Communication between ms-van3t (configured to emulate one vehicle with stationID equal to 1) and a commercial ETSI C-ITS stack (configured to emulate a vehicle with stationID 699702254).

implemented as part of ms-van3t, for both its simulation and emulation features.

Fig. 11 depicts, on the left, the GUI of the Cohda emulator, together with its output during a CAM exchange test, while the SUMO GUI and ms-van3t output are shown on the right. It is worth underlining that the Cohda stack was configured to disable the security protocols, as they are not yet implemented in ms-van3t, which, for the time being, can only send non-secured GeoNetworking messages.

Alongside the interoperability tests, we also evaluated the capability of ms-van3t to emulate a large number of vehicles in real-time. We set up a more complex scenario 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 Device Under Test (DUT). Every time the DUT 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 goal of these tests was to analyze the performance of the framework to emulate a high number of vehicles sending messages to a DUT 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. In particular, we leveraged the IEEE 802.11p model described in Section 3, with a transmission power of 23 dBm for all emulated OBUs.

We then created an ad-hoc application 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. Each emulation session has been set to last for 1000 s, with an increasing ms-van3t vehicle density from 20 vehicles up to a total of 150 vehicles. The obtained PPS values, for different numbers of vehicles, have then been compared to a set of *expected values*. These expected values have been obtained by running pure simulations of the system presented earlier, which do not require any real-time constraint. Thus, they represent the ideal PPS values that should be obtained if everything were 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, allowing us to identify the limit after which ms-van3t starts to slow down and cannot support real-time emulation any longer. It is worth noting that real-time capabilities of the emulation are affected by both the number of emulated nodes and the total number of scheduled packets. In order to showcase the effect of both factors in our evaluation, we tested both a traffic generation regulated dynamically



(a) CAM dynamic frequency management [10]

(b) CAM fixed to 10 Hz and 20 Hz frequency.

Fig. 12. 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.

depending on the dynamic condition of vehicles (as foreseen by ETSI), and two fixed packet frequency values, in which each emulated vehicle generates the same amount of traffic. In the latter case, for fixed packet frequencies, the total number of scheduled packets increases linearly with the total number of emulated vehicles.

Fig. 12 depicts the obtained results. On the x-axis, we report the total number of vehicles emulated by ms-van3t, while, on the y-axis, we represent the difference between the achieved PPS on the Cohda stack, receiving packets from ms-van3t, and the expected PPS.

We repeated the test for two different devices, i.e., the laptop described 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 evaluate the limits when providing large computing resources. The same measurements have also been performed both when keeping the standard-compliant dynamic CAM frequency management [1], and when fixing such a frequency to both 10 Hz (the smallest possible periodicity according to ETSI [1]) and 20 Hz (experimental periodicity adopted by a few European projects, such as 5G-CARMEN [35]), to analyze a worst-case scenario.

As expected, the available hardware affects the capability of emulating vehicles in real-time. In particular, a standard AMD laptop CPU can emulate up to 70 vehicles in real time when keeping the CAM dynamic frequency, and up to 40 and 30 vehicles with a fixed frequency of 10 Hz and 20 Hz respectively. These values almost double for a high-power Intel Core i9 CPU, being respectively up to around 120, 60 (10 Hz case), and 40 vehicles (20 Hz case). The obtained results show how even a standard laptop CPU is capable of emulating, with ms-van3t, a relatively large amount of vehicles, without any slowdown and as if the simulated nodes were real vehicles sending messages in real time. Importantly, these results compare well with the ones presented for other solutions available in the literature, focused exclusively on HIL, as opposed to ms-van3t, in which emulation is just one of the available features. In particular, in [15] performance evaluation results of an emulation-only framework are shown for a maximum of 85 vehicles inside the scenario. On the other hand, considering simulation frameworks with emulation capabilities, such as VENTOS [13], there is no performance evaluation found in terms of maximum number of vehicles and only an application with a total of 4 vehicles is available. When comparing the results achieved at fixed frequencies, it becomes evident that the real-time emulation capability is influenced not only by the number of emulated nodes but also by the volume of traffic they generate. As expected, doubling the packet frequency not only reduces the maximum number of vehicles that can be supported for real-time emulation but also leads to a slowdown behavior that becomes twice as pronounced with an increasing number of emulated nodes. 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. The main outcome of these tests is thus twofold: firstly, if proper resources are provided to ms-van3t, even larger numbers of emulated vehicles could be potentially handled in real-time, as new generations of more powerful CPUs are hitting the market; secondly, future work should consider parallelization of the code when in emulation mode, to better exploit multiple cores on today's CPUs.

Finally, it should be mentioned that, even if the most significant tests presented here involve only two physical V2X stations (i.e., the laptop running ms-van3t, and the one with the Cohda Wireless stack to emulate a real vehicle), ms-van3t is not limited to this setup.

Indeed, vehicles simulated by ms-van3t can reach the external world and communicate with real connected vehicles or Road Side Units through one or more physical devices, depending on the user needs and on the availability of network interfaces on the device running our framework.

The only requirement is to create as many *FdNetDevice* objects as the number of desired external devices used for communication with ms-van3t in emulation mode. Therefore, it is possible to configure both *(i)* a one-to-one mapping between simulated vehicles and network interfaces/devices communicating with external entities (for a limited number of vehicles), or *(ii)* a many-to-one mapping, in which all the traffic generated/received by simulated vehicles is routed through a single physical network interface (or a limited set of network interfaces). For instance, if the device running ms-van3t supports the connection of a single IEEE 802.11p device via USB or Ethernet, it is possible to make the emulated vehicles communicate with a real vehicle through a single network interface. This single interface will work as the connection between the simulated and the real world, as shown through the measurement depicted in Fig. 12.

6. Conclusions and future work

We presented a novel, open-source tool, called ms-van3t, for the evaluation and testing of V2X communication and services. The tool, based on ns-3 and supporting the integration of hardware in the loop, exhibits several innovative features, among which the possibility of easily switching between different wireless access technologies, all grouped inside a single repository, enabling a quick and efficient comparison among IEEE 802.11p, Release 14 LTE-V2X Mode 4, 5G NR-V2X, and LTE. Importantly, even if the access technology models are state-of-the-art and come from other open-source projects, they have been adapted and fully integrated in ms-van3t, with the aim of creating a self-contained tool for an easy validation of V2X-based applications.

Further, ms-van3t implements an ad-hoc ETSI C-ITS stack, which enables not only large scale simulations, but also the emulation of vehicle communication capabilities, thanks to its ability to receive and transmit ETSI-compliant messages from/to a physical interface of the device on which the framework is running. In addition to receiving SUMO traces as input, our framework can also be fed with real GNSS traces to simulate vehicle mobility, enabling the possibility of testing the behavior of applications in the presence of real positioning errors.

To showcase the capabilities and flexibility of ms-van3t, we presented three sample applications, as a guideline to develop and test V2X-based use cases chosen by the user, as well as some results in terms

of both application and access technology-related KPIs. Beside showcasing what can be achieved with ms-van3t, the results highlighted the relevance of safety applications powered by V2X communications, and highlighted the performance of the available access technologies. In particular, we have found that IEEE 802.11p can provide the smallest latency for the considered vehicle densities, but it is outperformed, given a transmission power level, by LTE-V2X and NR-V2X when measuring the packet reception ratio at relatively high baseline values (e.g., 200 m).

Future work will focus on expanding the ms-van3t framework, implementing an ETSI-compliant security module, which is not yet available, together with the implementation of a standard-compliant Local Dynamic Map. Furthermore, the integration of a Cooperative Positioning Service, which is undergoing standardization by ETSI, is currently under development. This new service enables vehicles to transmit and receive the so-called Collective Perception Messages, which contain information about detected objects and can provide nearby road users with an additional level of awareness. Finally, future work will integrate additional models for access technologies, including Millicar [49], for the evaluation of mmWave technologies in vehicular scenarios.

CRediT authorship contribution statement

F. Raviglione: Conceptualization, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **C.M. Risma Carletti:** Conceptualization, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **M. Malinverno:** Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. **C. Casetti:** Conceptualization, Supervision, Project administration, Writing – original draft, Writing – review & editing. **C.F. Chiasserini:** Conceptualization, Supervision, Project administration, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Co-author Carla-Fabiana Chiasserini is an Editor-in-Chief for Elsevier Computer Communications. Co-author Francesco Raviglione is part of the Technical Committee for Elsevier Computer Communications.

Data availability

Data will be made available on request, and our framework and code is already publicly available on GitHub.

Acknowledgments

This work has been partially supported by the European Union – Next Generation EU, through the National Center for Sustainable Mobility (Grant No. E13C22000980001). In addition, this work was Funded by the European Union through the project CONNECT under grant agreement no. 101069688. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] ETSI, 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, Standard ETSI EN 302 637-2 V1.4.1, European Telecommunications Standards Institute, 2019.

[2] ETSI, 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, Standard ETSI EN 302 637-3 V1.3.1, European Telecommunications Standards Institute, 2019.

[3] 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, Standard ETSI TS 103 301, European Telecommunications Standards Institute, 2016.

[4] GitHub, ms-van3t - a multi-stack VANET framework for ns-3 [online], 2021, https://github.com/ms-van3t-devs/ms-van3t.

[5] C. Sommer, R. German, F. Dressler, Bidirectionally coupled network and road traffic simulation for improved IVC analysis, IEEE Trans. Mob. Comput. 10 (1) (2011) 3–15.

[6] A. Wegener, M. Piorkowski, M. Raya, H. Hellbrück, S. Fischer, J.-P. Hubaux, TraCI: An interface for coupling road traffic and network simulators, in: Proceedings of the 11th Communications and Networking Simulation Symposium, CNS'08, 2008.

[7] A. Virdis, G. Stea, 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, http://dx.doi.org/10.5220/0005040000590070.

[8] G. Nardini, D. Sabella, G. Stea, P. Thakkar, A. Virdis, Simu5G–an OMNeT++ library for end-to-end performance evaluation of 5G networks, IEEE Access 8 (2020) 181176–181191.

[9] R. Riebl, H.-J. Günther, C. Facchi, L. Wolf, Artery: Extending veins for VANET applications, in: 2015 International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS, 2015, pp. 450–456, http://dx.doi.org/10.1109/MTITS.2015.7223293.

[10] A. Abunei, C.-R. Comşa, 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, http://dx.doi.org/10.1109/ISSCS.2017.8034921.

[11] 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, F. Cartolano, iTETRIS: A modular simulation platform for the large scale evaluation of cooperative ITS applications, Simul. Model. Pract. Theory 34 (2013) 99–125.

[12] V. Todisco, S. Bartoletti, C. Campolo, A. Molinaro, A.O. Berthet, A. Bazzi, Performance analysis of sidelink 5G-V2X mode 2 through an open-source simulator, IEEE Access 9 (2021) 145648–145661.

[13] M. Amoozadeh, B. Ching, C.-N. Chuah, D. Ghosal, H.M. Zhang, VENTOS: Vehicular network open simulator with hardware-in-the-loop support, Procedia Comput. Sci. 151 (2019) 61–68, 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.

[14] G. Shah, R. Valiente Romero, N. Gupta, S.M.O. Gani, B. Toghi, Y. Fallah, S. Gupta, Real-time hardware-in-the-loop emulation framework for DSRC-based connected vehicle applications, 2019.

[15] B. Mafakheri, P. Gonnella, A. Bazzi, B. Masini, M. Caggiano, R. Verdone, Optimizations for hardware-in-the-loop-based V2X validation platforms, 2021, http://dx.doi.org/10.1109/VTC2021-Spring51267.2021.9448667.

[16] C. Obermaier, R. Riebl, 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, http://dx.doi.org/10.1109/ITSC.2018.8569663.

[17] G. Nardini, G. Stea, A. Virdis, Scalable real-time emulation of 5G networks with Simu5G, IEEE Access 9 (2021) 148504–148520.

[18] A. Noferi, G. Nardini, G. Stea, A. Virdis, Rapid prototyping and performance evaluation of ETSI MEC-based applications, Simul. Model. Pract. Theory 123 (2023) 102700.

[19] Z. Ali, S. Lagén, L. Giupponi, R. Rouil, 3GPP NR V2X mode 2: Overview, models and system-level evaluation, IEEE Access 9 (2021) 89554–89579.

[20] M. Malinverno, F. Raviglione, C. Casetti, C.-F. Chiasserini, J. Mangues-Bafalluy, M. Requena-Esteso, A multi-stack simulation framework for vehicular applications testing, in: Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications, DIVANet '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 17–24, http://dx.doi.org/10.1145/3416014.3424603.

[21] P.A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, E. Wiessner, Microscopic traffic simulation using SUMO, in: 2018 21st International Conference on Intelligent Transportation Systems, ITSC, 2018, pp. 2575–2582, http://dx.doi.org/10.1109/ITSC.2018.8569938.

[22] nsnam, ns-3 network simulator, 2021, https://www.nsnam.org/.

[23] 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, Standard ETSI EN 302 636-5-1, European Telecommunications Standards Institute, 2019.

[24] 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, Standard ETSI EN 302 636-4-1, European Telecommunications Standards Institute, 2020.

[25] F. Eckermann, M. Kahlert, C. Wietfeld, Performance analysis of C-V2X mode 4 communication introducing an open-source C-V2X simulator, 2019, pp. 1–5, http://dx.doi.org/10.1109/VTCFall.2019.8891534.
[26] 3GPP, Technical Specification Group Radio Access Network; Study on LTE-based V2X Services; Release 14, Technical Report (TR) 36.885, 3rd Generation Partnership Project (3GPP), 2015, Version 14.0.0.
[27] GitHub, ns3 module for bidirectional coupling with SUMO, 2021, https://github.com/vodafone-chair/ns3-sumo-coupling.
[28] ETSI, ETSI TS 103 300-3 V2.1.1 (2020-11) - Intelligent Transport Systems (ITS); Vulnerable Road Users (VRU) Awareness; Part 3: Specification of VRU Awareness Basic Service; Release 2, Standard ETSI TS 103 300-3 V2.1.1, European Telecommunications Standards Institute, 2020.
[29] S. Khalfallah, B. Ducourthial, Bridging the gap between simulation and experimentation in vehicular networks, in: 2010 IEEE 72nd Vehicular Technology Conference - Fall, 2010, pp. 1–5, http://dx.doi.org/10.1109/VETECF.2010.5594402.
[30] C. Chen, Y. Zhu, An evaluation of vehicular networks with real traces, in: 2012 IEEE 18th International Conference on Parallel and Distributed Systems, 2012, pp. 716–717, http://dx.doi.org/10.1109/ICPADS.2012.110.
[31] S. Ollander, F.A. Schiegg, F.-W. Bode, 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, http://dx.doi.org/10.23919/FUSION45008.2020.9190612.
[32] C.F.F. Karney, Transverse mercator with an accuracy of a few nanometers, J. Geod. 85 (8) (2011) 475–485.
[33] C. Karney, GeographicLib 2.1 documentation [online], 2022, https://geographiclib.sourceforge.io/.
[34] F. Raviglione, S. Zocca, A. Minetto, M. Malinverno, C. Casetti, C. Chiasserini, F. Dovis, From collaborative awareness to collaborative information enhancement in vehicular networks, Veh. Commun. 36 (2022) 100497.
[35] F. Raviglione, C.M. Risma Carletti, C. Casetti, F. Stoffella, G. Yilma, F. Visintainer, S-LDM: Server local dynamic map for vehicular enhanced collective perception, in: 2022 IEEE 95th Vehicular Technology Conference (VTC2022-Spring) (Accepted Paper), 2022.
[36] ETSI, ETSI ITS: automotive intelligent transport systems (ITS), 2021, https://www.etsi.org/technologies/automotive-intelligent-transport.
[37] GitHub, asn1c [online], 2021, https://github.com/vlm/asn1c.
[38] ETSI, ETSI TS 102 894-2 V1.2.1 (2014-09) - Intelligent Transport Systems (ITS); Users and Applications Requirements; Part 2: Applications and Facilities Layer Common Data Dictionary, Standard ETSI TS 102 894-2 V1.2.1, European Telecommunications Standards Institute, 2014.
[39] Society of Automotive Engineers, Surface Vehicle Standard J2735 MAR2016 Dedicated Short Range Communications (DSRC) Message Set Dictionary, Standard SAE J2735, European Telecommunications Standards Institute, 2016.
[40] ETSI, ETSI EN 302 895 V1.1.1 (2014-09) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM), Standard ETSI EN 302 895 V1.1.1, European Telecommunications Standards Institute, 2014.
[41] ETSI, ETSI EN 302 637-2 V1.3.2 (2014-11) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service, Standard ETSI EN 302 637-2 V1.3.2, European Telecommunications Standards Institute, 2014.
[42] ETSI, ETSI EN 302 637-3 V1.2.2 (2014-09) - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service, Standard ETSI EN 302 637-3 V1.2.2, European Telecommunications Standards Institute, 2014.
[43] CTTC, The LENA ns-3 LTE Module Documentation, Documentation, Centre Tecnològic de Telecomunicacions de Catalunya, 2014.
[44] 3GPP TR 36.885, v14.0.0, Technical Specification Group Radio Access Network; Study on LTE-based V2X Services; Release 14, Technical Requirement, 3rd Generation Partnership Project, 2016.
[45] M.H.C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, A. Kousaridas, A tutorial on 5G NR V2X communications, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1972–2026.
[46] T.V. Nguyen, P. Shailesh, B. Sudhir, G. Kapil, L. Jiang, Z. Wu, D. Malladi, J. Li, A comparison of cellular vehicle-to-everything and dedicated short range communication, in: 2017 IEEE Vehicular Networking Conference, VNC, 2017, pp. 101–108, http://dx.doi.org/10.1109/VNC.2017.8275618.
[47] 3GPP, Study on Evaluation Methodology of New Vehicle-to-Everything (V2X) use Cases for LTE and NR. Release 15, Technical Report (TR) 37.885, 3rd Generation Partnership Project (3GPP), 2019, Version 15.3.0.
[48] F. Raviglione, M. Malinverno, C. Casetti, Open source testbed for vehicular communication, in: Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing, Mobihoc '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 405–406, http://dx.doi.org/10.1145/3323679.3326623.
[49] M. Drago, T. Zugno, M. Polese, M. Giordani, M. Zorzi, MilliCar: An ns-3 module for mmwave NR V2X networks, 2020, pp. 9–16, http://dx.doi.org/10.1145/3389400.3389402.

Francesco Raviglione was born in Biella, Italy, in 1994. He received a B.Sc. degree in computer engineering from Politecnico di Torino, followed by an M.Sc. degree in mechatronics engineering, with a focus on automotive and embedded systems. In 2022, he pursued a Ph.D. cum laude in Electronics and Telecommunication Engineering in Politecnico di Torino, presenting a final thesis work titled "Open platforms for connected vehicles". During his Ph.D., he also spent a six-month visiting period at the Roux Institute at Northeastern University, in Portland, Maine, USA. He is currently an Assistant Professor with time contract at the Department of Electronics and Telecommunications in Politecnico di Torino. He is working in the field of developing and evaluating platforms able to provide vehicular connectivity, on open source, customizable, solutions for wireless networking use cases, and on network measurements and performance assessment.

Carlos Mateo Risma Carletti is a Ph.D. student at the Department of Control and Computer Engineering in Politecnico di Torino. He is working on the evaluation of vehicular networking technologies for automated driving and on the development of flexible and open source message encoders and decoders for vehicular messages.He is also dealing with the development and testing of innovative vehicular micro-cloud applications.

Marco Malinverno, who got his PhD from Politecnico di Torino in 2021, works in Italdesign Giugiaro as Connect Car and V2X specialist. He spent 6 months as visiting PhD student at the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC). He is currently leading a project focused on the integration of connected functionalities in electric and hybrid vehicles.

Claudio Casetti is a Full Professor at the Department of Control and Computer Engineering, Politecnico di Torino. He has published more than 200 papers in peer-refereed international journals and conferences on the following topics: vehicular networks, 5G networks, transport and network protocols in wired networks, and IEEE 802.11 WLAN.

Carla Fabiana Chiasserini is a Full Professor at Politecnico di Torino, Italy, and a Research Associate with the Italian National Research Council (CNR). She was a Visiting Researcher at UC San Diego from 1998 to 2003, and a Visiting Professor at Monash University in 2012 and 2016. She is a Fellow of the IEEE.