POLITECNICO DI TORINO

Master Degree course in Automotive Engineering

Master Degree Thesis

Integration of VRUs in Cooperative Perception: Extending CARLA and VaN3Twin

Supervisors
Prof. Claudio Ettore Casetti
Carlos Mateo Risma Carletti
Diego Gasco
Francesco Raviglione

Candidate
Victor Fiorin de Carvalho

Academic Year 2024-2025

Acknowledgements

I would like to thank my supervisors, Prof. Claudio Casetti, Carlos Risma, Diego Gasco, and Francesco Raviglione. Each of you contributed in a unique and essential way to this thesis, and without your guidance, this work would never have taken shape.

I also wish to thank my friends, with whom I had the pleasure of sharing this path, those who have been there from day one, and those who joined for only a brief but joyful moment in time. You are all part of this story and without your support, I would not have made it here.

Finally, I want to express my deepest gratitude to my family. To my parents and sister, thank you for your love and support, your belief in me, and the strength you have given me at every step. It is because of you that, through the rain, I arrived here dry.

Abstract

The integration of Vulnerable Road Users (VRUs) into Cooperative Intelligent Transportation System (C-ITS) is essential to enhance road safety, however, it introduces significant challenges related to network scalability. Evaluating their integration in real-world field testing is still limited given the related costs, complexity, and logistical constraints, therefore, simulation plays an essential role in this area of research. This thesis extends the VaN3Twin-CARLA co-simulation framework to support and investigate passive and active participation of VRUs in cooperative scenarios. The contributions of this work are as follows: (i) The integration of VRUs into the OpenCDA framework, enabling realistic perception of VRUs; (ii) the implementation of ETSI-compliant communication services, allowing VRUs to actively participate by transmitting Vulnerable Road User Awareness Messages (VAMs); (iii) the implementation of a perception-based clustering mechanism for pedestrians, to evaluate its effect on Collective Perception Messages (CPM) size and channel load; (iv) the assessment of network impact of connected VRUs under varying densities and penetration rate. The results demonstrate that clustering significantly reduces the fluctuations in CPM size, being three times lower, on average, in the most dense scenario. Clustering also stabilizes channel load, measured through the Channel Busy Ratio (CBR), particularly in dense scenarios. In addition, the results show that the inclusion of connected VRUs increases the channel load as the penetration rate increases. In particular, the CBR values doubled with three times the number of pedestrians. The channel load remained well below the critical values in all tested conditions, reaching 12.5% in the worst-case scenario. This thesis provides key insights into the performance of C-ITS with VRU participation in realistic simulated environments, supporting the development of safer and more efficient cooperative mobility solutions.

Contents

List of Figures

List of Tables

Acronyms

3GPP	Third Generation Partnership Project.
AI	Artificial Intelligence.
AI4CCAM	Artificial Intelligence for Connected and Cooperative Automated Mobility.
BTP	Basic Transport Protocol.
CAM	Cooperative Awareness Message.
CAV	Connected and Autonomous Vehicle.
CBR	Channel Busy Ratio.
CCAM	Connected and Cooperative Automated Mobility.
C-ITS	Cooperative Intelligent Transportation System.
CDA	Cooperative Driving Automation.
CPM	Collective Perception Message.
C-V2X	Cellular Vehicle-to-Everything.
D2D	Device-to-Device.
dBi	Decibel-isotropic.
dBm	Decibel-milliwatts.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise.
DENM	Decentralized Environmental Notification Message.
DSRC	Dedicated Short-Range Communication.
DUT	Dalian University of Technology.
EDCA	Enhanced Distributed Channel Access.
EMA	Exponential Moving Average.
ETSI	European Telecommunications Standards Institute.
GNSS	Global Navigation Satellite System.
gRPC	Google Remote Procedure Call.

HIL	Hardware-in-the-Loop.
HMI	Human-Machine Interface.
IEEE	Institute of Electrical and Electronics Engineers.
ITS	Intelligent Transportation System.
LDM	Local Dynamic Map.
LiDAR	Light Detection and Ranging.
LoS	Line of Sight.
LTE	Long Term Evolution.
MAC	Medium Access Control.
NR-V2X	New Radio Vehicle-To-Everything.
ns-3	Network Simulator 3.
OBU	On-board Unit.
ODD	Operational Design Domain.
OFMD	Orthogonal Frequency Division Multiplexing.
OMNeT++	Objective Modular Network Testbed in C++.
OpenCDA	Open Cooperative Driving Automation.
PRR	Packet Reception Ratio.
RAT	Radio Access Technology.
RSU	Road-side Unit.
SAE	Society of Automotive Engineers.
SPaT	Signal Phase and Timing.
SUMO	Simulation of Urban Mobility.
TraCI	Traffic Control Interface.
UE	User Equipment.
UPER	Unaligned Packed Encoding Rules.
V2I	Vehicle-to-Infrastructure.
V2N	Vehicle-to-Network.
V2P	Vehicle-to-Pedestrian.
V2V	Vehicle-to-Vehicle.
V2X	Vehicle-to-Everything.
VANET	Vehicular Ad-hoc Network.

VAM	Vulnerable Road User Awareness Message.
Veins	Vehicles in Network Simulation.
VENTOS	Vehicular Network Open Simulator.
VRU	Vulnerable Road Users.
WAVE	Wireless Access in Vehicular Environments.

Chapter 1

Introduction

The automotive industry is currently experiencing an extensive transformation, driven by advances in on-board technology, pushing vehicles towards higher levels of automation and connectivity. A crucial point of this transformation is the development of the Intelligent Transportation System (ITS), which combines traffic management, communication technologies, and automation to make road transport safer, more efficient, and more sustainable. ITS is a broader term and refers to all types of services that are related to transport, including logistics, statistics, maintenance, and safety, among others. Adaptive traffic lights, dynamic route guidance, and electronic toll collection are some examples of ITS services.

As a subset of this framework, Cooperative Intelligent Transportation Systems (C-ITS) have emerged. C-ITS is characterized by cooperative information sharing through Vehicle-to-Everything (V2X) technologies, in which any road participant can be included if equipped with the necessary communication hardware, such as on-board units for vehicles or personal devices carried by pedestrians. The purpose of this communication framework, called V2X, is to improve traffic efficiency and safety by allowing vehicles to share their state, such as position and speed, and receive external information from the infrastructure or other vehicles about hazards, traffic conditions, and other road users. For example, a vehicle could be alerted about a pedestrian crossing a road in a place possibly hidden from the vehicle. This exchange of information extends the field of view of each vehicle, increases their situational awareness, and supports safer and more coordinated driving. In simple terms, C-ITS aims to create an ecosystem in which vehicles, infrastructure, and road users can cooperate with each other.

Building on C-ITS, the concept of Cooperative Driving Automation (CDA) has been introduced by SAE (Society of Automotive Engineers) J3216 [45]. CDA can be understood as an extension of traditional driving automation, introduced by SAE J3016 [46], with an additional cooperative feature enabled by V2X communication. In this concept, vehicles do not act simply relying on their own sensor data but also in coordination with others. Typical use cases include maneuvers for lane merging, where a vehicle communicates its intention to join a lane, and the other vehicles may adapt to this request accordingly, and platooning, where vehicles act in coordination to synchronize

their speed and spacing, improving safety and efficiency.

While CDA encompasses not only cooperation between vehicles and infrastructure, but also with Vulnerable Road Users (VRUs), including pedestrians, cyclists, and motorcyclists, ensuring their safety remains a major challenge. These road users are often more exposed and at higher risk of severe injury in the event of collisions. According to the World Health Organization, pedestrians and cyclists account for more than a quarter of all road traffic deaths worldwide, with pedestrians alone accounting for approximately 22% [52]. Their exposure, unpredictable behavior, and lack of physical protection, combined with potential occlusions due to parked vehicles blocking the line of sight or even other pedestrians, make it challenging for autonomous systems to detect and respond accordingly in critical conditions.

To address these challenges, cooperative perception (CP) has emerged as a possible solution. The principle is that vehicles and infrastructure nodes share their sensor data through a vehicular communication network, extending each other's perception capabilities beyond their line of sight [5, 34]. These communication networks are often referred to as Vehicular Ad-hoc Networks (VANETs) and constitute the foundation of a large number of ITS services, where all the actors involved (vehicles, VRUs, infrastructure nodes) can exchange information directly without relying on a fixed network backbone. Nonetheless, given the common interest in reducing the number of traffic accidents, researchers and developers have prioritized services related to the transmission of critical and safety information [24].

Although cooperative perception extends situational awareness, it also introduces scalability challenges. In dense traffic scenarios with multiple vehicles and VRUs exchanging data simultaneously, the communication channel can become overloaded, which may compromise the delivery of safety-critical information.

To reduce this channel load, techniques such as clustering have been proposed in the literature. In clustering, multiple pedestrians with similar positions and movements are grouped into a single entity before broadcasting, reducing the number and size of transmitted messages.

The realization of such a highly interconnected ecosystem with vehicles, infrastructure, and other road users is a growing trend, with a few pilot projects that demonstrate its feasibility, which will be presented in Section 2. However, widespread implementation in the real world is still a challenge, mostly due to safety, cost, and logistical constraints. Deploying hundreds of connected vehicles and pedestrians with experimental communication hardware is not practical, especially when studying safety-critical scenarios. Consequently, simulation plays an essential role in the development and testing of V2X technologies. Simulation provides a controlled and reproducible environment while supporting large-scale experiments without safety risks [2].

Although many simulation frameworks offer support for autonomous driving and V2X research, each of them has its own limitations. The simulation of VRUs, especially pedestrians, in a cooperative perception or V2X-based scenario is still quite limited.

For instance, SUMO (Simulation of Urban Mobility) offers a microscopic traffic simulation that accurately models traffic flows and pedestrian mobility, and it can be integrated with network simulators through dedicated frameworks such as VaN3Twin [41] or Veins [47]. However, its built-in VRU behavior remains simplistic, and the lack of perception and sensor modeling limits its use in cooperative perception cases. On the other hand, CARLA (Car Learning to Act) [7], which is an open source simulator designed for autonomous driving research, provides a high-fidelity environment, with a realistic mobility simulation scenario, together with its accurate sensor models, including cameras, LiDAR, and radars, among others.

To further leverage the potential of CARLA, frameworks such as OpenCDA have emerged [54]. This framework extends CARLA by providing a modular architecture that includes perception, localization, planning, control, and V2X communication modules, facilitating the development and evaluation of Cooperative Driving Automation (CDA) systems. Nonetheless, this framework still does not provide a realistic communication model and does not fully integrate pedestrians.

These limitations highlight the need for simulation frameworks with VRU capabilities, as not only passive but mainly active participants in the C-ITS. Thus, extending VANET simulations to include VRUs becomes essential for a better understanding of their integration into the C-ITS. This thesis therefore focuses on addressing this gap by extending the capabilities of the VaN3Twin-CARLA framework to accurately simulate the integration of VRUs into C-ITS environments, supporting future deployments of cooperative mobility solutions. In particular, the main contributions of the thesis are as follows:

i. **Integration of VRUs into OpenCDA:** incorporate VRUs into the OpenCDA framework by further developing existing modules, allowing the spawn of VRUs into the simulation, the perception module to identify and differentiate VRUs, and all the logic necessary to handle object matching and fusion;

ii. **Standard-compliant communication capabilities:** equip VRUs with communication capabilities compliant with ETSI standards. A VRU data provider module was developed, allowing these actors to retrieve their state from the server and insert them into VAM messages, leveraging the existing VRU Basic Service facility to connect to OpenCDA;

iii. **Evaluation of VRU clustering in CPMs:** investigate the impact of pedestrian clustering on CPM messages in terms of message size and channel load, by implementing a clustering algorithm based on current detections and a fusion logic to better track a cluster, even if occlusions are present;

iv. **Assessment of connected VRUs in C-ITS scenarios:** assess the overall effect of integrating connected VRUs into the C-ITS context by equipping VRUs with different penetration rate, and measuring network performance indicators such as Channel Busy Ratio (CBR), latency, and Packet Reception Ratio (PRR), under different transmitting data rates;

The thesis is structured as follows: **Chapter 2** reviews the state of the art related mainly to the integration of VRU into the cooperative ITS framework, divided into three parts. First, the main communication technologies and standards for C-ITS are presented. Second, initiatives and real-life projects that somehow address VRU integration. And third, existing co-simulation frameworks related to mobility and V2X research are outlined, along with key contributions from the literature. **Chapter 3** introduces the simulation framework adopted in this thesis, describing the OpenCDA and VaN3Twin modules together with the modifications made within these modules and the newly created ones. **Chapter 4** explains the general setup of the simulation experiments, including the parameters and variables considered, for both the mobility and communication sides. Then, the specific scenarios for both experiments carried out in this paper are outlined together with the numerical results obtained. **Chapter 5** discusses these results in detail, highlighting their implications, along with the related literature. **Chapter 6** concludes this thesis by summarizing the main contributions, limitations, and directions for future research.

Chapter 2

Related Works

2.1 Connected Vehicle Technologies and Communication Standards

Connected and autonomous vehicles (CAVs) rely on Vehicle-to-Everything (V2X) communication technologies to improve situational awareness and improve safety, efficiency, and cooperation on the road. V2X encompasses the communication between the vehicle and all the other actors present in this ecosystem: vehicle-to-vehicle (V2V) communication enables cars to share their state such as position, heading, and velocity, and also information about perceived objects by their on-board sensors; vehicle-to-network (V2N) communication connects vehicles to the cellular mobile infrastructure, providing access to traffic information and cloud-based services such as dynamic route optimization; vehicle-to-infrastructure (V2I) communication connects vehicles to traffic lights and road site units, allowing a wide range of services such as signal phase and timing (SPaT), which allow vehicles to adjust their speed based on the state of the traffic lights, and speed limit warnings, which inform vehicles about the maximum speed on specific road segments; and vehicle-to-pedestrian (V2P) communication enables interaction with VRUs through smartphones or wearable devices, allowing vehicles to be alerted about the presence of vulnerable users. These technologies enable vehicles to share data with other road users, allowing them to coordinate maneuvers and respond to dynamic traffic conditions in real time. Moreover, combining these technologies extends the Operational Design Domain (ODD) of automated vehicles, which means extending the conditions under which an autonomous vehicle can operate safely.

Several telecommunication standards support the V2X functionality, such as IEEE 802.11p/bd, 3GPP Cellular V2X (C-V2X), and 3GPP NR-V2X. IEEE 802.11p, specifically developed for the vehicular environment, is the most widely used Dedicated Short-Range Communication (DSRC) standard for V2X. It is based on standard 5 GHz WiFi and operates in the 5.85 - 5.925 GHz band reserved for Intelligent Transportation Systems. The physical layer integrates seven 10 MHz channels and a 5 MHz guard band, to avoid interference with other technologies. IEEE 802.11p employs Orthogonal Frequency Division Multiplexing (OFDM) as the modulation technique, which supports data rates

from 3 Mbit/s up to 27 Mbit/s. The Medium Access Control (MAC) layer is based on Enhanced Distributed Channel Access (EDCA) from IEEE 802.11e, which provides four access categories with different contention window sizes and inter-frame spacing, allowing data prioritization for time-critical safety messages. The low latency, maturity, interoperability, and robustness of the IEEE 802.11p standard make it a reliable choice for real-time vehicular applications. Its evolution, IEEE 802.11bd, currently under development, aims to improve throughput, reduce end-to-end latency, and increase transmission range, while having backward compatibility with its predecessor 802.11p [35].

Introduced by the Third Generation Partnership Project (3GPP), C-V2X is an alternative V2X technology based on 4G LTE and 5G infrastructure. It includes two communication modes, the PC5 interface or sidelink, and the Uu interface or uplink/-downlink. The PC5 interface is independent of cellular networks and enables direct low-latency device-to-device (D2D) communication between vehicles, road side units (RSUs), or pedestrians. In LTE-based C-V2X, sidelink transmissions are mainly based on a broadcast type, where the message is sent to all receivers in the transmission range. The sidelink mode is suited for V2V, V2I and V2P communication, in which low-latency is critical for safety applications. The Uu interface, in contrast, refers to the interface between the mobile device or User Equipment (UE), and the network base station. By leveraging the conventional cellular mobile infrastructure, it supports V2N communication, enabling vehicles to receive real-time information about the road, traffic conditions, and other types of services that do not require strict low latency and can benefit from having wide area coverage. The evolution of this technology, New Radio V2X (NR-V2X), aims at overcoming the limitations of LTE-based C-V2X while keeping the same two interfaces. In particular, the PC5 interface is extended to support four types of communication. In addition to the broadcast type, the NR-V2X sidelink supports unicast, groupcast, and multicast. Unicast means a one-to-one communication, where a message is sent from one sender to a single receiver. The groupcast type refers to a transmission where a message is sent to a specific group, for example, when a vehicle wants to merge in a platoon, only the participants of the platoon will receive the request. Similarly to groupcast where groups are predefined, multicast transmission is a one-to-many communication, but with dynamic groups managed by the network, for example, vehicles approaching an intersection will receive a specific message, and those moving away will not. NR-V2X also improves how radio resources are managed and allows a more flexible signal configuration, for instance, it can adjust the distribution of the radio spectrum, depending on whether the application needs a very slow delay or a high data rate, addressing the requirements for latency, reliability, and throughput for advanced V2X use cases [35].

Given these communication standards, to ensure interoperability between the systems and the effectiveness of communication, a standardization of the exchanged messages is necessary. To this end, the European Telecommunications Standards Institute (ETSI) has defined a set of message specifications and standards that regulate how information is structured and transmitted among participants and the C-ITS infrastructure. Defined by the standard ETSI EN 302 637-2 [14], Cooperative Awareness Messages

(CAM) are messages that are periodically disseminated, at a rate between 1 Hz and 10 Hz, depending on vehicle dynamics, to continuously provide their own position, speed, and direction, supporting basic road safety applications. In contrast, Collective Perception Messages (CPMs), specified in ETSI TS 103 324 [18], are used to broadcast information about perceived objects, such as vehicles, pedestrians, and animals, that were detected by the onboard sensors. CPMs are generated quasi-periodically, at a frequency of 1 Hz if the vehicle is not detecting anything, including in the message only basic information about the vehicle and the on-board sensors mounted on it, and are generated with a higher frequency, up to 10 Hz, if the vehicle is detecting another object, and include the observed status of objects, such as time, position, attitude, and other kinematic attributes. These messages are the backbone for cooperative perception, allowing connected and autonomous vehicles to share information and have better awareness of their surroundings, essential for autonomous driving and safety applications.

Further ensuring interoperability, ETSI specifies that all standardized messages must be encoded and decoded using Abstract Syntax Notation One (ASN.1) [25]. ASN.1 is a standardized interface description language used to describe how data are structured in these exchanged messages, ensuring that the same message can be correctly interpreted by different systems. In practice, ASN.1 defines how the content of the message should be structured and defines a few encoding rules which determine how this content is translated into bits for transmission. These encoding rules differ in their processing speed and message compactness. For ITS messages (CAMs [14], CPMs [18], and others) , ETSI adopts the Unaligned Packed Encoding Rules (UPER) [26], designed to minimize message size.

2.2 VRU Integration

Vulnerable Road Users, including pedestrians, bicycles, and motorcycles, account for a significant portion of global traffic fatalities. According to the European Commission's 2023 road safety report [19], within urban areas, vulnerable road users (pedestrians, cyclists, and two-wheeler powered users) represent almost 70 percent of road traffic deaths in the EU, and pedestrians alone account for one third of total emergencies. Therefore, the protection of VRUs has become a major focus within Cooperative Intelligent Transportation System (C-ITS) research.

Several projects have explored the integration of VRUs into C-ITS to enhance road safety. The iMOVE research center [48], among their projects on C-ITS and mobility, has one near completion, centered on integrating motorcycles into existing C-ITS frameworks, with the use of smart helmets and smart glasses, LEDs in the mirror and handlebars for warning signals, and haptic wristbands. The HEIDE project [20], with another approach, investigates a cooperative Human-Machine Interface (HMI), for both drivers and VRUs, aiming to improve situational awareness by integrating internal and external HMI solutions and synchronizing action recommendations for drivers and other road users. Complementary to this, the AI4CCAM [3] project investigates the use of Artificial Intelligence (AI) for Connected and Cooperative Automated Mobility (CCAM). The

main goal of the project is to develop an open environment integrating Trustworthy AI models that can anticipate the behavior of VRUs in mixed urban scenarios. In parallel, the SOTERIA project [36,37] aims to accelerate the attainment of the European Union's "Vision-Zero" goal by developing a comprehensive framework of data-driven tools and services to improve VRU safety. This includes accident and micro-mobility demand prediction models, risk detection, safe route recommendations, and a speed advisory system, among others. The effectiveness of these tools is being validated throughout Europe, which is called "living labs", a real-world testing ground where VRUs, stakeholders, researchers, and public actors cooperate to test and refine the solutions. Initiatives such as DECICE [29] are currently working on an edge-cloud AI-based management framework, with adaptive optimization of applications in a federated infrastructure. A key focus of this project is the use case for the so-called intelligent intersections, where the framework manages the different devices (Road Side Units, cameras, and other sensors) and their resources, and creates a digital twin, that is a virtual replica of the entire system, including the intersection and its devices, with real-time updates. This digital twin, combined with AI, is used for behavior prediction and resource prioritization.

Larger-scale urban C-ITS deployments have also begun to include VRUs in their services. Within the C-ROADs platform, for instance, which has the objective of harmonically deploying C-ITS related services throughout Europe, many pilot projects in cities such as Dresden, Hamburg, Graz, and others have been exploring the integration of VRUs, using Road Side Units (RSUs) for bicycle or crosswalk pedestrian detections, for example, according to the last available C-ROADs annual deployment report from 2023 [6]. The SAFE STRIP project [22], conducted across Italy, Spain and Greece, proposed a low-cost C-ITS solution that uses on-road strips embedded with micro / nano sensors and with communication technologies to provide real-time information about the environment, road conditions, and other road users. Field trials demonstrated its ability to improve safety for VRUs by alerting drivers about pedestrians preparing to enter the crosswalk. With a more robust implementation, the Aveiro Tech City Living Lab [44], in Portugal, is a city-scale testbed with 44 access points spread across the city, equipped with a variety of sensors, including Radar, LiDAR, and video cameras, and with a multi-protocol network integrating ITS-G5, C-V2X, WiFi, and 5G. The developed use cases in this ecosystem demonstrate support for VRU safety through integrated sensing (including the use of smartphone applications) and edge computing.

From a standardization perspective, ETSI experts have made significant progress in the standardization of VRU communication within the C-ITS architecture. ETSI TS 103 300-2 [16] and ETSI TS 103 300-3 [17] specify the VRU Awareness Basic Service and the characteristics of the VRU Awareness Message (VAM), including message generation rules, format specifications, and relevant use cases [15]. In particular, the trigger conditions for VRUs to transmit VAMs according to the specifications [17] are the following:

i. The Euclidean distance between the current position and the position lastly included in a VAM exceeds 4 meters.

ii. The difference between the current ground speed and the absolute speed of the VRU lastly included in a VAM exceeds 0.5 m/s.

iii. The difference between the orientation of the current velocity vector and the estimated orientation of the vector lastly included in a VAM exceeds 4 degrees.

iv. The time elapsed since the last VAM transmission exceeds 5000 ms.

v. The difference on the estimated interception probability since the last VAM exceeds 10%

vi. The VRU has decided to join a cluster

vii. The VRU has determined that there is another vehicle or VRU within a minimum lateral, longitudinal and vertical distance. The lateral and longitudinal distances correspond to the distances that the VRU could travel in 5000 ms.

By broadcasting their position, velocity, and heading via VAMs, VRUs can actively participate in cooperative traffic scenarios, allowing vehicles and infrastructure to anticipate potential collisions.

Despite these advances, several challenges continue to limit the large-scale real-world adoption of VRU communication. Firstly, VRUs exhibit far less predictable mobility behavior than vehicles. For instance, pedestrians typically move at low speeds but may cross the street independently of road signs, while cyclists move much faster, but still exhibit unpredictable behaviors. Secondly, the real-world localization accuracy is still far from ideal for safety-related VRU services. The Global Navigation Satellite System (GNSS) of smartphones, which is often considered a solution device for C-ITS, cannot provide an accurate position, starting at 3 to 5 meters of position error, up to tens of meters in dense urban environments, which critically affects the reliability of collision warnings [56]. Additionally, the lack of dedicated hardware in smartphones for direct communication, such as 802.11p or PC5 sidelink from C-V2X, poses a significant technical issue for their use in VRU safety applications. This limitation makes these devices unsuitable for low-latency safety applications, as they rely on the cellular network instead.

Finally, real-world trials that included VRUs as an active participant are scarce. Most pilot deployments focus on V2V or V2I services, treating VRUs as passive elements detected by sensors. Furthermore, these trials are often performed under controlled conditions, which may not capture the complexities of real heterogeneous traffic with its unpredictability. As a result, complementary to physical testing, simulation-based research has emerged, allowing repeatable large-scale testing under varying traffic and communication conditions, without the financial, safety, and logistical constraints of field deployments. The next section reviews existing VRU co-simulation frameworks that integrate mobility and communication layers for modeling cooperative safety scenarios.

Table 2.1: Summary of European projects addressing VRU C-ITS integration, communication, and safety

Project	**Focus**	**Contribution to VRU Communication**
iMOVE [48]	Motorcycles in C-ITS	Integration of powered two-wheelers into cooperative systems by means of wearable devices for warning signals.
HEIDI [20]	Human–Machine Interfaces	Development of HMI concepts for both drivers and VRUs for intention recognition. No active communication for VRUs.
AI4CCAM [3]	AI for Cooperative Mobility	Use of AI for perception and decision-making in C-ITS. Addresses VRU safety in mixed traffic through predictive models.
SOTERIA [36]	VRU Protection and Safety	Combines C-ITS communication, risk assessment, and prediction models to enhance VRU safety via real-time alerts, recommendations, and cooperative warnings. Passive VRUs.
C-ROADS [6]	C-ITS European deployment	Includes VRU-oriented test cases in large-scale C-ITS pilots across multiple member states. Mostly passive VRUs.
SAFE STRIP [22]	In-road sensing	Development of micro-sensor strips to detect VRUs and provide cooperative safety services.
DECICE [29]	Intelligent Intersection safety	Cooperative perception framework with pedestrian detection at intersections. Passive VRUs.
Aveiro Tech City [44]	Living lab deployment	Real-world pilots where smartphones act as cooperative beacons, transmitting VRU position via cellular networks.

2.3 VRU Co-simulation Frameworks

The evaluation of cooperative systems in simulation, in which VRUs are also participants, requires the integration of multiple domains: accurate mobility models for pedestrians and cyclists, a standard-compliant V2X communication stack (e.g. IEEE 802.11p, LTE-V2X or NR-V2X), and high-fidelity perception and sensing models, while maintaining accurate synchronization. No single simulation environment addresses all these

requirements. As a result, research has increasingly relied on co-simulation frameworks, connecting multiple simulators, and leveraging their individual strengths.

In the context of vehicular networks, Veins [47] is one of the most popular open source simulation frameworks. The Veins framework couples OMNeT++, a discrete event network simulator, with SUMO microscopic traffic simulator via TraCI (Traffic Control Interface), supporting the simulation of dynamic nodes equipped with IEEE 802.11p, LTE, or C-V2X. SUMO does support pedestrians and cyclists; however, most Veins-based studies treat them as passive elements, rather than connected C-ITS stations. Building on Veins, Artery [42] extends the framework by incorporating an implementation of the ETSI ITS-G5 protocol stack, including GeoNetworking, the Basic Transport Protocol (BTP), and support for standardized services such as CAM and DENM (Decentralized Environmental Notification Message), compliant with European ITS specifications.

Within this framework, a recent contribution for active VRUs is presented in [23]. The authors introduced a pedestrian model into the Artery framework, integrating the VRU Awareness Service and ADAS application relying on VAM safety messages. This work evaluates metrics such as Channel Busy Ratio (CBR) and detection performance under varying penetration rates. The analysis of the impact of VAM messages on network load is partially aligned with the work carried out in this thesis.

Another simulator relying on SUMO and OMNeT++ is called VENTOS (Vehicular Network Open Simulator) [4]. Different from Veins, this one implements the IEEE WAVE protocol stack, instead of the European standard ITS-G5. Moreover, VENTOS incorporates hardware-in-the-loop (HIL) capabilities, giving the possibility to physically test real hardware like On-board Units (OBU) and Road-side Units (RSUs) into the simulation environment. In HIL testing, the simulator provides a virtual environment while interacting with the real hardware under testing, for example, to evaluate the behavior of a real OBU under any scenario, without the constraints of field testing.

A growing area of research is focused on network congestion, usually in high-density VRU scenarios, where having multiple vehicles and pedestrians exchanging messages can overload the communication channel. This channel load, typically measured through the Channel Busy Ratio (CBR), can be significantly increased due to the high frequency of transmission and the size of the messages, leading to higher latency and packet loss. The CBR is determined by measuring the total busy time of the channel, in other words, the total time in which a transmitter finds the channel occupied divided by the duration of observation. A higher latency means that a transmitted message will take longer to arrive, and in the worst case, the message may not arrive at all, compromising the delivery of safety-critical information. To mitigate these scalability issues, clustering techniques have been proposed in which multiple VRUs with similar attributes (position, speed, and direction) are grouped into a single entity before broadcasting. A general introduction to clustering is presented in Section 2.3.1. Clustering can be performed actively by the pedestrians themselves, known as active VRU clustering [33, 51], where a pedestrian will try to form a cluster with other pedestrians with similar kinematic attributes, reducing the number of VAM transmitted. Alternatively, it can be performed

by vehicles or infrastructure, known as passive VRU clustering [53], in which vehicles generate a cluster to be included in CPMs, rather than the individual road user, reducing the size of the transmitting message.

For instance, Valle et al. [51] present a non-negotiable implicit VAM Clustering method, where cluster operations are performed without negotiation (no additional message exchange between pedestrians). This method, which is different from ETSI standards, halved the number of generated VAMs while maintaining VRU awareness levels. Similarly, Lobo et al. [33] present the impact of clustering methods on metrics such as CBR, position error, and latency. Their work relies on Artery, which provides a C-ITS network simulation environment, based on OMNet++ in combination with INET for wireless networking, and SUMO for traffic mobility. In addition, Vanetza is used to implement the ETSI C-ITS protocols within the simulated framework. Vanetza is an open-source software implementation of the full ETSI C-ITS protocol stack, used not only for network simulations but mainly to deploy the standard-compliant C-ITS stack into real OBUs. With VRU clustering, a significant reduction in CBR values and overall latency was observed, due to more available resources for the communication nodes.

Another study with a focus on network congestion and VRU clustering is presented in [53]. The authors first analyzed different potential shapes for VRU clustering in terms of accuracy and efficiency of cluster description and, secondly, the impact of including clusters on CPM messages rather than individual pedestrians. Although not performed on any of the aforementioned simulators, but rather using the Dalian University of Technology (DUT) dataset, which contains trajectories of real VRUs collected with a drone, the findings remain relevant. The results show that employing clustering can lead to up to two-thirds in the reduction of the CPM data rate.

Leveraging ns-3 (Network Simulator 3) and SUMO, the Van3Twin [41] framework implements a full ETSI C-ITS, supporting multiple access technologies, such as IEEE 802.11p, C-V2X, and LTE-V2X. Unlike OMNeT++, which is a general-purpose event simulation platform, based on a graphical interface, which requires additional frameworks such as INET or Veins to model communication protocols, ns-3 is specifically designed for network simulation. It is implemented in C++, where it provides the building blocks of devices (nodes), communication links (channel) and protocol stack, meaning that the simulator provides ready-to-use models of these instances, allowing them to be easily configured directly through code. Van3Twin supports large-scale simulations, integrating a measurement module for metrics collection, such as one-way latency and Packet Reception Ratio (PRR). Unlike other solutions, Van3Twin supports the usage of pre-recorded GNSS traces as an alternative to SUMO, giving the possibility to test the behavior of applications under real GNSS errors.

An extension of the Van3Twin framework is presented in [43]. In this work, the authors replaced the SUMO microscopic traffic simulator with the high-fidelity CARLA simulator, which is an open-source simulator designed to support the development, training, and validation of autonomous driving systems. This simulator allows for a realistic and customizable environment, including support for urban and rural scenarios, weather

and lighting variations, traffic generation, pedestrian behavior, and a diverse set of sensors such as cameras, LiDARs, radars, and others. It also supports integration with external modules or other simulation platforms, such as machine learning frameworks or network simulators, as presented in [43]. This control over the entire framework is possible through its powerful Application Programming Interface (API), an interface that allows the user to flexibly program all the aspects of the simulation and control all static and dynamic actors. Having high-fidelity physics and sensor simulation allowed CARLA to become one of the reference platforms in autonomous driving research.

The framework presented by Carletti et al. [43] further incorporates OpenCDA [54], an open source framework for developing and testing Cooperative Driving Automation (CDA) applications. OpenCDA offers a modular architecture that includes computer vision modules for perception, self-driving modules for localization, planning, and actuation, as well as V2X communication modules. However, the V2X stack in OpenCDA is not fully standard-compliant and works under simplified assumptions, such as perfect communication conditions, therefore reinforcing the need of integration with other simulation tools. This integration enables simulation of realistic traffic environments and a realistic perception model from CARLA, together with an ETSI-compliant V2X communication stack, from Van3Twin. This simulation framework proposed in [43] is shown in Figure 2.1, where the main modules are present, separating the existing ones (grey) from each simulator and the newly created ones (blue) to allow such integration. These modules will be explained in detail in Section 3, since they are the basis for this thesis.

It is worth mentioning that, at the time of writing, the VaN3Twin framework was further extended and renamed VaN3Twin [40] (previously ms-van3t). This latest extension leverages the Sionna Ray Tracer (RT), allowing for a high-fidelity representation of wireless propagation effects, such as Line-of-Sight (LoS) blockage, multi-RAT (Radio Access Technology) interference, Doppler effect, and others. However, this extension was not implemented in this work, which remains based on the previous VaN3Twin version, with wireless propagation relying on probabilistic models rather than ray tracing.

Despite the progress that these frameworks have brought, some limitations remain in terms of VRU integration. Common gaps include the scarcity of active VRU modeling as C-ITS stations, as VRUs are often treated as passive elements, and the lack of accurate communication models integrated in a realistic environment scenario. In ms-van3t, for instance, integration with SUMO pedestrians was present, either as connected or non-connected actors. However, the simulation framework lacked realistic sensor perception. Furthermore, most of the existing work has focused primarily on vehicle-only scenarios, leaving open questions regarding network state and cooperative sensing in heterogeneous traffic environments. The VaN3Twin-CARLA integration provides a promising foundation to address these limitations by merging an accurate communication model with realistic mobility and sensor perception. This is the key aspect added by CARLA, which enables high-fidelity sensor perception. The next section, following clustering explanation, presents the co-simulation framework developed in this thesis, extending VaN3Twin-CARLA with VRU communication services and novel clustering

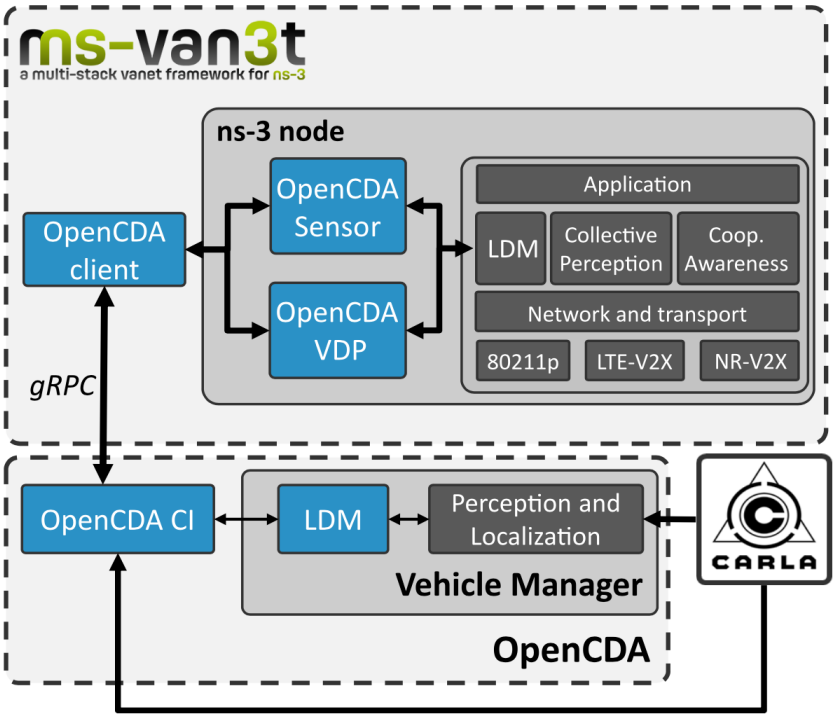


Figure 2.1: Architecture of ms-van3t-CARLA illustrating the interaction between all the implemented modules (blue) with the existing ones (grey). Figure reproduced from [43]

mechanisms, evaluating both communication efficiency and cooperative perception performance in mixed traffic scenarios.

2.3.1 Clustering

This section is dedicated to an explanation of the idea of clustering and the algorithm used in this work. Clustering, as defined by the literature, is an unsupervised machine learning algorithm that organizes any kind of data into groups, based on their similarities or patterns. An unsupervised algorithm means that raw data is used as input, without labels or prior classification, and the algorithm itself will try to find hidden patterns in the data presented. Clustering can be applied to almost any field to analyze data, and each type of data requires a different clustering algorithm, as shown in [39]. For instance, it can be used in marketing to identify common purchasing behavior from customers; in image processing, grouping pixels according to their intensity, helping in the identification of different areas in a Magnetic Resonance Imaging (MRI) for example; in biology, identifying groups of molecules based on their structure; and in spatial applications, grouping objects in dense areas based on their proximity.

Table 2.2: Comparison of selected co-simulation frameworks for C-ITS with VRU considerations

Framework	**Core Components**	**VRU Support**	**Notable Contributions / Use Cases**
Veins [47]	SUMO + OMNeT++	Passive VRU mobility in SUMO; no native active VAMs	Foundation for V2X. Base for Artery
Artery [42]	Veins + ETSI ITS-G5 stack	Active VAM broadcast possible via extensions	[23] VAM-based VRU protection. CBR evaluation.
VENTOS [4]	SUMO + OMNeT++ + WAVE protocol stack	Passive VRUs; potential for active logic	HIL for OBUs/RSUs. Adaptable to VRU communication.
VaN3Twin [41]	SUMO + ns-3 + ETSI C-ITS stack	No default VRU module; full network fidelity	PRR/latency studies under multiple RATs
VaN3Twin-CARLA [43]	VaN3Twin + CARLA + OpenCDA	VRUs possible via CARLA/SUMO. Active VRU requires custom development.	Base for present thesis: active and passive VRU integration, perception-based clustering, CBR analysis.
VaN3Twin [40]	VaN3Twin + Sionna Ray Tracing	VRUs possible via CARLA/SUMO. Active VRU requires custom development	High-fidelity wireless propagation model

There are several clustering algorithms, each with their own strengths and limitations. One of the most popular is the K-means clustering algorithm, in which the algorithm partitions the data into k clusters by iteratively minimizing the distances of the data points and the center of the cluster. It is a very fast algorithm, but requires the prior definition of k (the number of clusters the algorithm will output). Another type of clustering algorithm is the hierarchical one, in which it creates a multilevel structure of clusters (tree-like structure) by merging smaller clusters into larger ones or dividing larger clusters into smaller ones. Although the definition of the number of clusters is not required as in the k-means algorithm, the hierarchical approach can be computationally expensive for large datasets.

Among the many algorithms present in the literature [39], Density-Based Spatial Clustering Application with Noise (DBSCAN) stands out due to its robustness and flexibility. This method does not require a prior specification of the number of clusters, and can cluster data in any shape while efficiently managing outliers, identifying them as *noise* (data points that are not part of any cluster). The output of this algorithm is a list of labels corresponding to the input data, where each point is assigned to a cluster or defined as an outlier. An example of this algorithm is shown in Figure 2.2, where it shows the different clusters formed, with different colors, and the data outliers, in black. The DBSCAN algorithm requires the definition of two parameters:

- ϵ **(Epsilon)**: the maximum distance between two points to be considered part of the same cluster.
- **minPts**: the minimum number of points required to form a cluster.

These parameters are crucial and directly influence the results and should be tuned according to the type of dataset. The minPts parameter defines how many points should be close together to be considered a dense region. It generally scales with the size of the dataset, so in a large dataset, it may be desirable to increase this parameter. The parameter ϵ is more sensitive to variation. When chosen too small, the algorithm may not find any cluster and all data may be considered an outlier; if chosen too large, the algorithm may cluster all the data into a single cluster.

The working principle of DBSCAN is the concept of core samples or core points. A core point is a point in the dataset such that there exists a minimum number of points (**minPts**) within a certain distance (ϵ), often called neighbors of the core point. A cluster is formed by recursively taking a core point, and finding its neighbors that are also core points, and finding their neighbors which are also core points, and so on. This is done until the neighboring points do not meet the criteria to be a core point, therefore, defined as border point. A point that is not a core point, and is at least **Epsilon** in distance from the closest core point is considered an outlier. Figure 2.3 illustrates the idea of core points, border points, and outliers.

In the context of this work, the DBSCAN algorithm was chosen, particularly to be used in two circumstances: first, as a support for the perception algorithm presented in 3.1.2, in which the algorithm is applied to the LiDAR points within the detected bounding box, isolating the points of interest (the points that actually "hit" the object); second, it is used as a basis to create a cluster of detected pedestrians, as described in 3.2.3, in which the algorithm is applied in a dataset containing the list of the positions of all detected pedestrians.

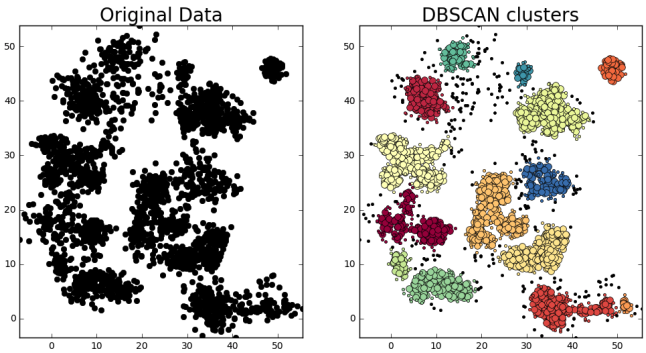


Figure 2.2: Example of DBSCAN algorithm. Reproduced from [9]

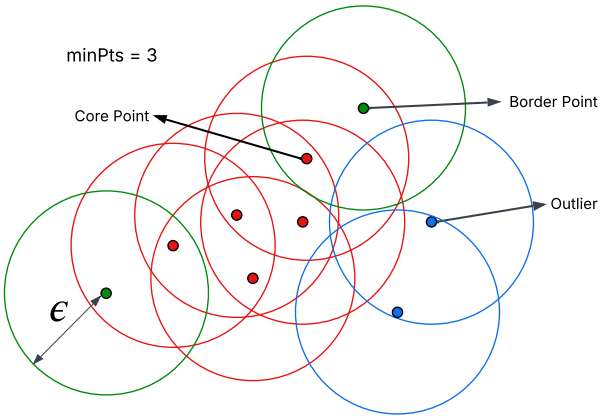


Figure 2.3: Illustration of DBSCAN algorithm with minPts = 3. Red points are core points; Green are border points; and blue are outliers.

Chapter 3

Co-Simulation Framework

This thesis is built upon the open source framework presented in [43]. This combines mobility, physics, and sensor-generated data from CARLA with accurate communication models from VaN3Twin. By using Google Protobuf and gRPC Remote Procedure Call for the communication bridge, and with the implementation of certain modules such as the OpenCDA Client and CARLA Adapter Interface (which is the correspondence of the OpenCDA CI module in Figure 2.1), it became possible to query perceived information from ns-3, for each of the simulated nodes. A simulated ns-3 node represents any vehicle, VRU or RSU equipped with communication capabilities, therefore, capable of sending and receiving messages within the simulation. This framework enables cooperative perception evaluation through synchronized traffic and network simulation. Figure 3.1 shows the implemented architecture for this simulation, together with a simple explanation of what each module is responsible for. Although OpenCDA and VaN3Twin contain many modules, the figure presents only those that constitute the core of the simulation.

The following sections provide a brief description of the role of each module, along with the modifications implemented to integrate Vulnerable Road Users into this framework. In particular, the following modules will be described:

- Scenario Manager (3.1.1): scenario and traffic creation.
- Perception Manager (3.1.2): detection and sensor fusion.
- Local Dynamic Map (3.1.3): local representation of the traffic environment.
- CARLA Adapter Interface (3.1.4): gRPC server to bridge OpenCDA and VaN3Twin.
- OpenCDA Client (3.2.1): gRPC client, manages the co-simulation interface and the synchronization of both simulators.
- OpenCDA VRUdp (3.2.2): Data Provider for active VRUs (those that transmit VAMs).

- OpenCDA Sensor (3.2.3): Synchronization of LDMs and initial clustering algorithm.
- VaN3Twin script setup (3.2.4): configuration for VaN3Twin initialization, ns-3 node creation, and setup (Appendix 6).
- Metric Supervisor (3.2.5): collection of communication-related performance metrics.
- Facilities Layer (3.2.6): Functionalities and services to support ITS applications. Within this layer, services for generating CPMs and VAMs are described (CP basic service and VRU Awareness basic service, respectively), together with the clustering algorithm, developed within the LDM facility.

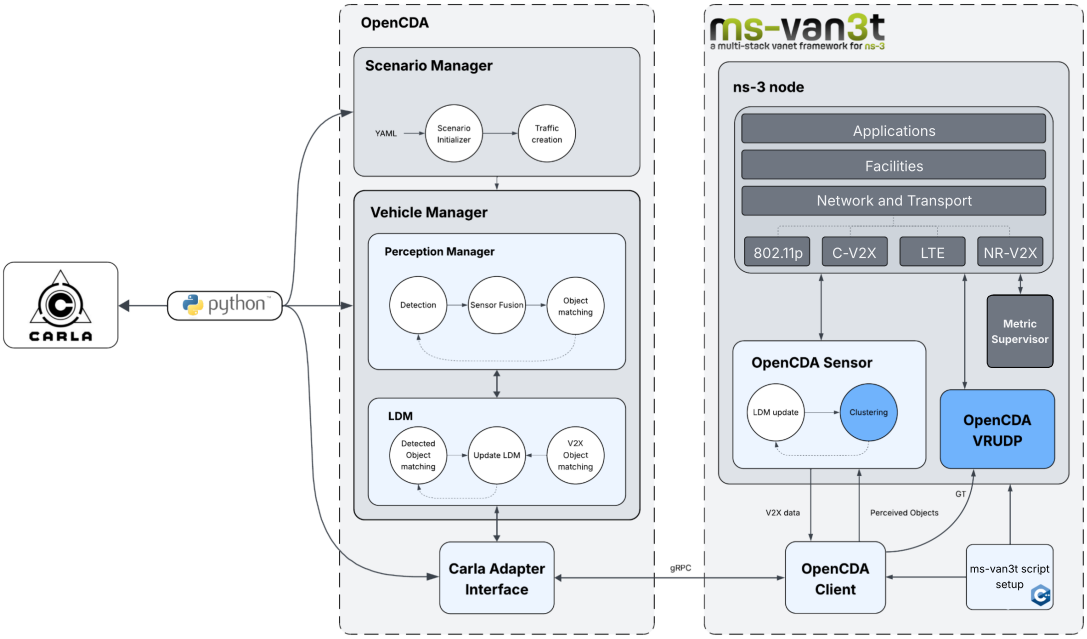


Figure 3.1: Co-simulation architecture VaN3Twin-CARLA illustrating the interaction between different modules. Light blue modules were adapted for this work. Dark blue modules were implemented as new work. Dark gray represents the communication stack and the metric supervisor module

3.1 OpenCDA modules

3.1.1 Scenario Manager

The Scenario Manager module is responsible for managing the construction of the CARLA simulation. It configures all the simulation's characteristics at initialization.

Specifically, it creates the scenario in the desired town, sets the weather conditions, and spawns all connected and autonomous vehicles, background vehicles, and pedestrians. This set of parameters defines the characteristics of the mobility simulation that are specified in the YAML files, which are configuration files that use a key-value structure, where the key represents a parameter and the value defines its setting, allowing the simulation to be easily configured without modifying the source code. These parameters include the spawn points, which are predefined positions in the simulation map in which vehicles and pedestrians are placed, and the destination for background vehicles and CAVs, together with their base behavior. They also include spawn points and areas for pedestrians, as well as the perception system configuration for each CAV. In particular, this involves defining the number and position of sensors, such as RGB cameras, used to capture visual information as a conventional camera does, and a LiDAR sensor, which uses laser light to detect and measure distances to objects in sight, creating a 3D representation of the surroundings.

For pedestrian spawning, this module introduces two methods: list-based spawning and radius-based spawning. Using a list, it is necessary to know all the spawn positions for each pedestrian in advance, which can be quite labor intensive as it requires retrieving them from the server and defining them manually. On the other hand, spawning by radius allows pedestrians to be positioned randomly in the simulation, within a specified area, given the number of desired pedestrians, the radius, and the center point. This second approach allows the creation of multiple areas with pedestrians with varying densities. Additionally, the radius-based method allows the definition of the same destination for all the pedestrians, described by a circle; otherwise, the simulator will set a random destination for each one.

3.1.2 Perception Manager

In this module, the detection of vehicles and VRUs is carried out, together with the fusion of information from the different sensors, and the implementation of object matching algorithms. The latter refers to the process of pairing the detected objects with the corresponding object ground truth from simulation. Figure 3.2 shows what happens inside this module, from camera images to having a final detected object to be stored. Initially, all RGB images are passed through the YOLOv5 algorithm [28], a well-known computer vision machine learning model for object detection. The output of the detection algorithm contains the position of the 2D bounding box, the detection label, and its confidence level, which represents the probability that the detected object actually belongs to the predicted class, simply saying, the probability that a detected pedestrian is actually a pedestrian, for example. For each detected object, a DBSCAN cluster algorithm (explained in Section 2.3.1) is applied to the LIDAR points located within its bounding box, isolating only the points of interest. Essentially, it finds the main dense cluster of points that most likely constitutes the detected object. If there are enough points of the LIDAR hitting the object, the 3D bounding box for that object is calculated, otherwise the perceived object is discarded. Once the bounding box is calculated, an obstacle instance for this object is created and inserted into the object list, the data

structure created to save all perceived vehicles, pedestrians and other VRUs.

The next step is to match these detected objects with the ground-truth list of all the actors in the simulation. This matching process is treated as a linear assignment problem. The cost matrix is first computed by initializing a matrix N x M with reasonably high values, where N is the number of ground-truth objects, and M is the number of detected objects, in essence, the length of the object list. The matrix is populated considering only the Euclidean distances between the objects of both lists as the cost of assigning a detected object with a ground-truth object. Then, the linear assignment algorithm finds the best matches for all the perceived objects, and the IDs from the ground-truth actors are assigned to the detected objects that were matched. Moreover, the speed and heading angle are retrieved from the server to have more precise data, since the heading angle of the detection algorithm is not reliable and fluctuates constantly, and the speed calculation algorithm has not yet been implemented. Finally, possible duplicates are removed.

It is worth noting that the removal of duplicates is done only for vehicles, since the linear assignment algorithm with the cost matrix is done for pedestrians, and this process already removes possible duplicates. These duplicates come from the fact that the RGB cameras will superimpose a small area, and since each image is passed through the YOLOv5 algorithm one at a time, an object that is found within this superimposed area will be counted twice. Additionally, objects that have a confidence level, from the YOLOv5 detection, smaller than a certain threshold are also removed. These thresholds, selected from experiments, are set to be 0.7 for vehicles and a much lower value of 0.4 is decided for pedestrians, given the intrinsic limitations of a detection algorithm in a simulated environment, such as having a not-so-realistic environmental model, for instance. This much lower value was chosen to keep track of pedestrians as much as possible, reducing false negatives, situations where clearly detected pedestrians were being excluded due to low confidence values. Figure 3.3 shows the distribution of confidence levels for all VRUs that were matched with their corresponding ground truth, throughout many simulation runs, varying the number and position of spawned pedestrians. With the selected threshold, more than 90 percent of true detected pedestrians remain for further processing, allowing for a better continuous monitoring of the perceived actors.

Tracking a pedestrian's trajectory is essential for their inclusion in CPMs, as it ensures reliable information is being shared over the network, rather than being transmitted as soon as the object is detected. To this end, a detected pedestrian will be sent to VaN3Twin for a possible inclusion in a CPM only if this pedestrian is detected for at least ten frames. Although the perception algorithm for pedestrians guarantees that a detected pedestrian is actually a pedestrian, the ETSI standards for CPM inclusion [18] introduce the idea of *object perception quality*, which means the quantification of the likelihood that the object actually exists, in other words, the object has been detected and is continuously detected by the sensors. This quantification of the detection quality was not implemented; however, it was decided that ten frames, which corresponds to half a second of the simulation, is enough for it to be considered a stable detection. The detected objects have to be sent to VaN3Twin to provide information from the

CARLA simulation. They are filtered inside the CARLA Adapter Interface to ensure that just updated data will be sent and stored in the on-board databases containing the perceived objects (called Local Dynamic Maps, LDMs). The management of LDMs and the synchronization process between CARLA and VaN3Twin will be explained in the next section.

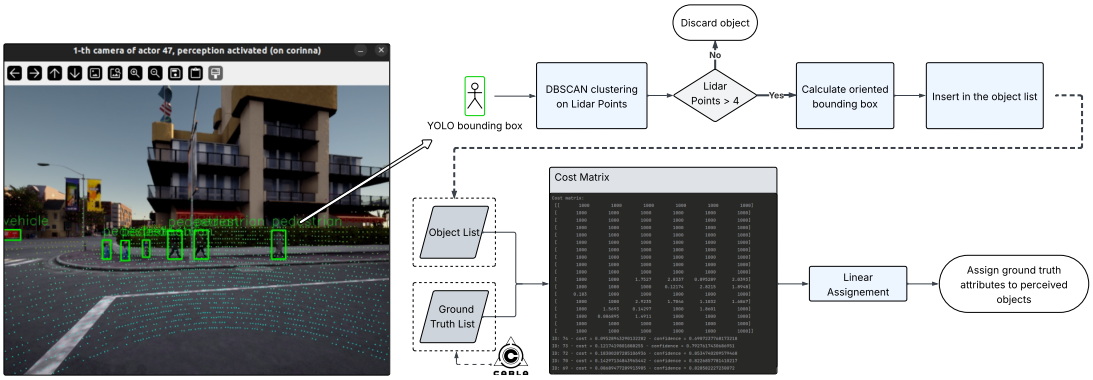


Figure 3.2: Perception Manager module overall process. Camera and Lidar detection and fusion, and further matching with ground truth objects.

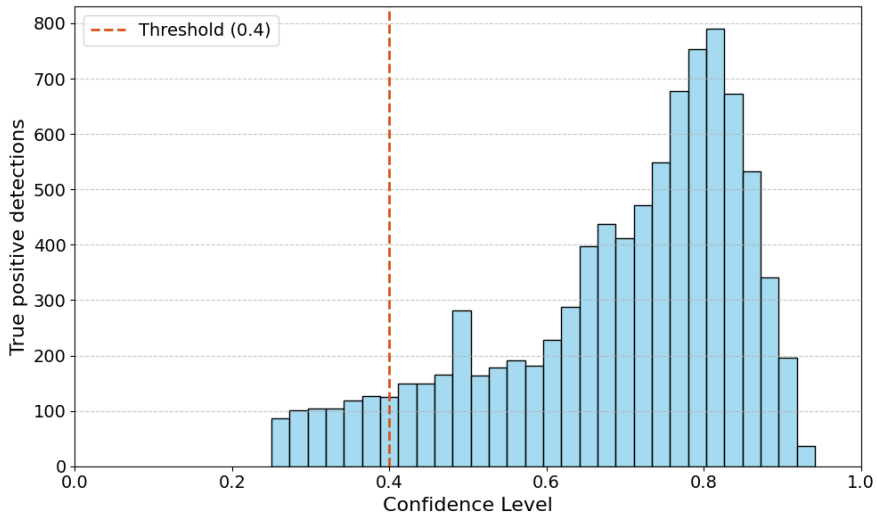


Figure 3.3: Distribution of confidence levels for over 9000 true positive detections. Vertical dashed line represent confidence threshold used for evaluation. The distribution has a median around 0.73. The majority of detections, 92 percent, exceeds the threshold value of 0.4

3.1.3 Local Dynamic Map

The Local Dynamic Map (LDM) module, as envisioned by ETSI [13], is a data structure that maintains a local representation of the traffic environment. It has been implemented with the objective of storing and keeping track of dynamic information about objects detected by either on-board sensors or receive from V2X messages, at each simulation step. In the OpenCDA framework, the LDM is structured within the Vehicle Manager module, as shown in Figure 3.1, therefore, each connected vehicle has its own LDM module and, of course, its own perception manager module. All objects identified and matched during the perception stage will be inserted into the LDM, either as a new entry or as an update to an existing track.

This module deals not only with locally perceived objects but also with other actors in the simulation, received from connected vehicles through CAMs and CPMs or from connected pedestrians through VAMs. The core of the process within this facility is a matching algorithm that treats the problem as a linear assignment task, by calculating the Intersection over Union and the distance between new entries and stored ones. When an actor is first inserted into the LDM, a Kalman filter [27, 31] is initialized for that particular actor. The Kalman Filter is responsible for predicting the next state of that actor, given the current stored information and the time elapsed since the last measurement. For pedestrians, in particular connected ones, position and speed are estimated using a Kalman Filter with a constant velocity model. Given the frequency and standard rules defining when a connected pedestrian will transmit its state, the constant velocity assumption remains valid between transmissions. Since a VRU transmits only upon a change in behavior or status, ensuring reliable and adequate state estimation, and a minimum transmission interval for packets is also defined, both mechanisms are useful given that we assume and employ a constant speed model.

In addition to the functions already implemented to match locally perceived objects with objects perceived from another CAV, through CPMs, or with connected vehicles, through CAMs, the new `VAMfusion` function was implemented to combine the information transmitted via VAM messages with data collected from on-board sensors for VRUs. It optimally pairs connected pedestrians with its own track. In case it is not the first time that the pedestrian transmits a VAM, it appends the information retrieved from the update step of the Kalman Filter, using the position information contained in the VAM message as the input for this Kalman Filter step. Whereas, in case it is the first time we receive data from this VRU, the algorithm performs a matching as a new local detection, using Intersection over Union and distances as matching parameters. If there is a match, and the connected pedestrian is also perceived locally by the on-board sensors, the details included in the VAM and the corresponding match stored in the LDM are merged. This merged info is then used as the input for the calculation of the Kalman Filter update step. If no matching is found, a new entry is created, together with the initialization step of the Kalman Filter for that actor. Figure 3.4 shows the idea behind the `VAMfusion` algorithm. It shows two main diagrams; the upper one shows the steps taken to handle Perceived Objects (PO), while the lower one shows the logic for the newly implemented function. The two dashed lines represent cases in which,

for instance, the pedestrian is sending VAMs and only later, the same pedestrian starts being perceived by the sensors, or vice versa, leading to the part where these data are merged.

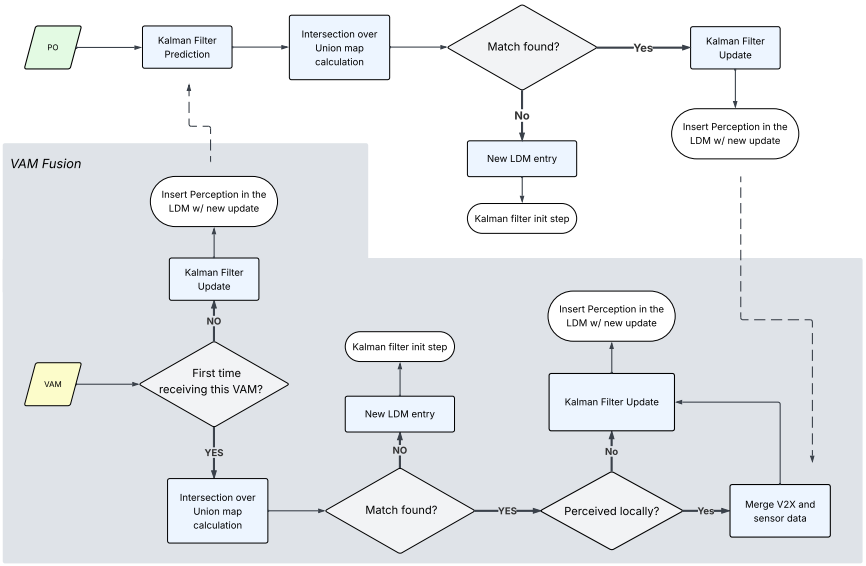


Figure 3.4: `VAMfusion` function flowchart with the gray background. The upper part illustrates briefly the steps for a new perceived object. Dashed lines represent where the information is merged in the case of PO and VAM being the same actor

3.1.4 CARLA Adapter Interface

As already detailed in [43], this module creates a gRPC (Google Remote Procedure Call) server, which is a framework that connects two or more applications or services, having the server side (where the gRPC is initialized) and the client side (the other application or service in which it will connect to the server side). In our case, the gRPC works as a bridge between OpenCDA and VaN3Twin, enabling bidirectional communication between the simulators, allowing them to exchange data in real time. It works mainly with Protocol buffers (Protobuf), which is a mechanism for storing and sending structure data in a simple way, independent of the programming language of the applications involved. This protocol works with a blueprint of the data to be exchanged (`.proto` file). This blueprint is structured with services, which define the functions in which the client will use to retrieve data, and messages, which define the type of data that the function will return.

This interface was mainly designed to provide LDM information for CAVs, for a possible later inclusion on a CPM, and basic mobility information for background vehicles, for them to send CAMs with their own information. This is achieved by encoding the information from the LDM into the same structure defined in the associated configuration

file (`.proto` file) for Protobuf messages.

The CARLA Adapter Interface works by allowing the other simulator to call the CARLA methods and to control the simulation through remote calls. To guarantee this functionality and synchronicity, CARLA is configured to run in synchronous mode, which means that the simulation advances only when triggered. This module is responsible for catching the VaN3Twin trigger for a step forward in the simulation (called tick), and executing the update on the CARLA side. Each tick represents a fixed time duration that matches the duration of one VaN3Twin mobility step.

In addition to providing local information for V2X applications, either perceived objects to be sent on CPMs, or ground truth information to be used for the generation of CAMs and VAMs, this interface is responsible for inserting V2X objects into OpenCDA LDM. Based on the type of object received from VaN3Twin, such as CPM objects, CAMs, or VAMs, this interface will select the appropriate fusion algorithm, such as the implemented `VAMfusion`, to ensure accurate matching and LDM synchronization.

Lastly, this interface enables the control of CAVs by an application or facility within VaN3Twin. By requesting speed, acceleration, and possible waypoints, it translates into CARLA-compliant commands, including steering, throttle, and brake. Background vehicles, on the other hand, are managed by CARLA's built-in traffic manager. It is important to note that, in this current simulation setup, CAVs also rely on the traffic manager itself, rather than using the OpenCDA modules for control and actuation, since there was no interest in this work in addressing vehicle behavior in different conditions.

3.2 VaN3Twin modules

3.2.1 OpenCDA Client

The OpenCDA Client module is responsible for managing the co-simulation interface between the mobility simulator (i.e., CARLA) and the network simulator (contained in VaN3Twin) through the use of a gRPC adapter. It implements all necessary gRPC interfaces required to interact with CARLA Adapter Interface. It coordinates the execution of the OpenCDA simulation by establishing a connection with the CARLA Adapter Interface and managing the synchronization of the two sides at each simulation step [43].

At simulation start-up, a pool of idle ns-3 nodes is created based on the number of CAVs and Connected VRUs defined in the scenario configuration, given the limitation of ns-3 in dynamically creating nodes during runtime. The OpenCDA Client module is responsible for assigning each CARLA actor to a corresponding ns-3 node. This mapping ensures a one-to-one correspondence between actors in the mobility scenario with nodes in the network domain, allowing nodes to retrieve their mobility information at any point using their unique CARLA actor identifier

To maintain synchronization and coherence between mobility and communication, the OpenCDA Client updates the position of each ns-3 node at every simulation step to match the real-time positioning of their associated CARLA actor. This continuous synchronization ensures that the communication behavior reflects the actual movement

and state of their counterparts simulated in CARLA.

To reflect real-world scenarios, only a subset of CARLA actors are assigned ns-3 nodes, based on the penetration rate parameter, defined in the main VaN3Twin setup script (described later in Section 3.2.4). The OpenCDA Client applies this rate during the initialization phase, enabling only a configurable percentage of vehicles and pedestrians to participate in the communication scenario. This allows for flexible experimentation with varying levels of connectivity, facilitating the analysis under various deployment conditions.

3.2.2 OpenCDA VRUdp

The Vehicle Data Provider (VDP) in VaN3Twin acts as an interface that provides the Facilities and all layers of the C-ITS stack with vehicle dynamics and status data used to generate standard-compliant messages. Similarly, the Vulnerable Road User Data Provider (VRUdp) module performs the same role, but for pedestrians and other VRUs, offering the mobility data required for their integration into the C-ITS communication framework.

The OpenCDA VRUdp module implemented in this work maintains the same concept, interfacing with CARLA through the OpenCDA Client. It allows facilities, such as VRUBasicService (explained later in Section 3.2.6), to access real-time mobility information of VRUs within the CARLA simulation to be inserted into each message (VAM) generated by that road user. Each connected VRU retrieves data, such as speed, position, and heading, by querying the OpenCDA Client using its unique ID assigned at simulation start-up.

3.2.3 OpenCDA Sensor

The OpenCDA Sensor module is mainly responsible for the synchronization of the LDMs. This synchronization of information allows the OpenCDA LDM to manage the data fusion while ensuring rapid access to the updated data by the VaN3Twin LDM, relying on fewer remote calls during the simulation. The reason for having fewer remote calls is that, even though both LDMs are synchronized at every simulation step, the information of all actors in the LDM is transferred in one call. Once stored in the VaN3Twin LDM, the data to be inserted into messages such as CPMs can be accessed locally by ns-3, without the need to query or request the OpenCDA LDM for every perceived object. However, it should be noted that connected actors, such as pedestrians capable of communicating, still require individual remote calls to retrieve their own dynamic state that will be later included in VAM messages.

The alignment of both LDMs is performed first by inserting all V2X-derived data into the OpenCDA LDM, including connected vehicles via CAMs, connected pedestrians via VAMs, and perceived objects through CPMs. During this step, actor by actor, OpenCDA LDM performs data matching and fusion, as described in Section 3.1.3. Once all entities have been processed, the OpenCDA Sensor retrieves the updated LDM, containing data

from both perception and communication, and synchronizes it with VaN3Twin LDM by removing outdated perceived objects.

Other than its main function, this module was extended as part of this work to include a clustering algorithm. Following the synchronization and fusion of data, there is a specific function to perform this clustering operation, which is called `createCluster`. The cluster creation algorithm works by first applying a simple Density-Based Spatial Clustering Application with Noise (DBSCAN) algorithm, based on [8], to all actor positions stored in the LDM, including connected pedestrians. The working principle of DBSCAN is explained in Section 2.3.1.

The initial clustering step consider only spatial proximity to identify potential groups. For each resulting cluster, the circular mean heading is calculated. Given a predefined threshold, the heading of each pedestrian is compared against the mean heading of the group. Pedestrians whose orientation deviates by more than this predefined value are excluded. This process is repeated until no other pedestrian is excluded. If the final cluster has at least the minimum number of pedestrians, the cluster goes through a matching assignment to be inserted into the LDM.

OpenCDA Sensor also provides the capability of real-time visualization of all the actors stored in the LDM, along with their ground-truth information. The visualization interface clearly distinguishes between objects perceived by onboard sensing and those received via cooperative messages from connected actors. An example of the graphical user interface (GUI) output provided by this module is shown in Figure 3.5. Additionally, the rendered frames are logged and stored individually for each CAV present in the simulation, supporting post-simulation analysis and validation.

3.2.4 VaN3Twin Script Setup

This module is responsible for initializing and configuring the simulation environment in VaN3Twin. It handles all the setup required before the simulation starts, including reading configuration files, parsing command-line arguments, establishing connectivity with CARLA and OpenCDA servers, configuring wireless communication parameters, and setting up all the ns-3 nodes. Runtime parameters, such as the total simulation duration, the penetration rate for both vehicles and VRUs, and the output file names for communication metrics, are also defined here. In Appendix A, it is presented a code snipped from the VaN3Twin script, with key parts of the configuration.

To ensure consistency between physical and communication layers of the co-simulation, the number of actors spawned in the simulation is retrieved from the same YAML file used for scenario creation. Based on this information, a pool of nodes is created, applying the specified penetration rate for communication capabilities for both vehicles and VRUs. These nodes are then equipped with wireless devices configured with the specified parameters, such as data rate and transmission power.

Following node creation, the cooperative perception application is installed on each one. This application is responsible for managing and configuring the Facilities layer on every ns-3 node. It attaches the node to its mobility source (OpenCDA/CARLA in this case), instantiates the LDM and Data Provider module (VDP for CAVs, VRUdp for

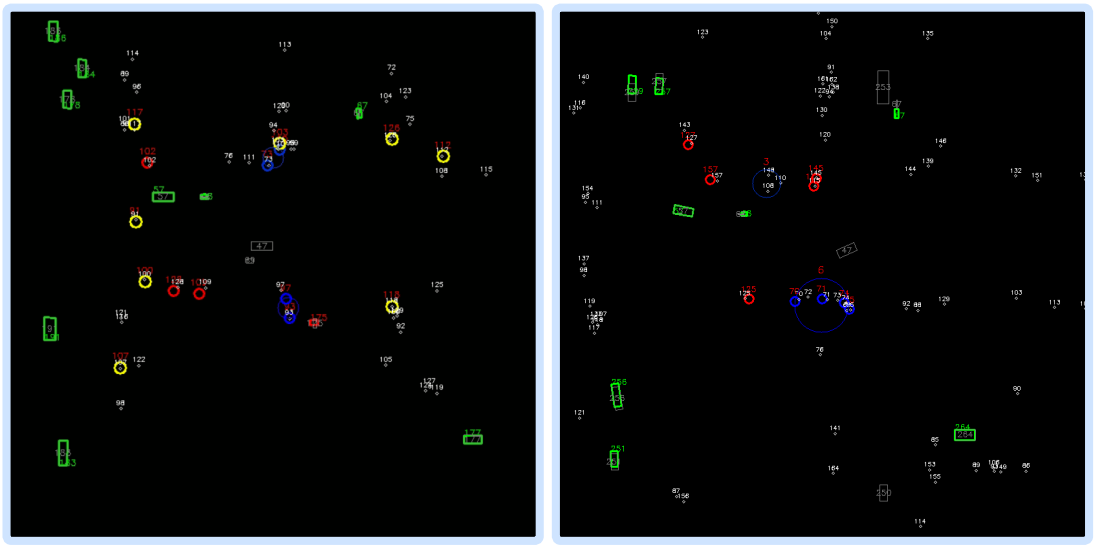


Figure 3.5: Real-time visualization of the LDM content for the same CAV in two different scenario configurations,as rendered by the OpenCDA Sensor module. It shows the ego CAV (centered gray box), connected vehicles transmitting CAMs (green boxes), ground-truth pedestrian position (gray circles), pedestrians perceived either through onboard sensors or received via CPMs (red circles), connected pedestrians broadcasting VAMs (yellow circles) and clustered pedestrians detection (blue circles)

VRUs), and initializes the corresponding sensor interface (OpenCDA Sensor 3.2.3). It then sets up the transport and networking layers (BTP and GeoNetworking), configured to use the selected access technology (i.e. IEEE 802.11p). In addition, it initializes the ETSI Facilities services, which are responsible for generating and processing CAM, CPM, and VAM messages, in accordance with ETSI standards [14, 17, 18]. The application sets up communication sockets, assigns unique identifiers, and defines callback functions for message reception to update the LDM. In practice, the application configures the cooperative stack on each node so that it can generate, transmit, and process cooperative messages during simulation.

In parallel, the OpenCDA Client is instantiated within ns-3. As explained in Section 3.2.1, this module establishes the co-simulation interface between perception and communication layers, and ensures a one-to-one mapping between each ns-3 node with a corresponding CARLA actor. This mapping enables the application, and any other module, to access mobility information at any point using the unique ID of the corresponding actor.

Finally, this module also initializes the `metricSupervisor`, which is responsible for managing the collection and evaluation of communication metrics throughout the simulation. These metrics include CBR and latency, as introduced in Section 2.3, and Packet Reception Ratio (PRR). The PRR expresses the amount of packets or messages that were successfully received and is defined by the ratio of the total received packets to the

total transmitted ones. These metrics are then logged into a CSV file for offline analysis. The configuration parameters, including the window size and the alpha value for the CBR calculation, are provided at initiation. The window size defines the measurement or observation time interval for the CBR calculation, so a new CBR is calculated at the end of every *measurement window*. The alpha parameter represents a smoothing factor for the exponential moving average (EMA) calculation of the CBR; in other words, it is the weight of the previous CBR value when calculating the new one. A lower alpha value means the EMA is more responsive to recent changes because it gives more importance to newly calculated values. Meanwhile, a higher alpha gives more weight to older data, smoothing out short-term fluctuations.

3.2.5 Metric Supervisor

The *MetricSupervisor* module in VaN3Twin is responsible for the collection of communication-related performance metrics during simulation. In this work, the module was adopted without modifications and was primarily leveraged to evaluate the Channel Busy Ratio (CBR), a key indication of channel utilization in vehicular networks.

The simulation focuses on understanding the impact of channel load on cooperative perception mechanisms rather than individual packet delivery performance. Although this study does not implement a Decentralized Congestion Control (DCC) mechanism, CBR remains highly relevant, as it has been widely adopted as a basis for adaptive transmission control in vehicular networks [32]. DCC is a mechanism that allows each transmitter to independently adjust its transmission parameters, such as frequency and transmission power, and the data rate. The transmitting device adjusts these parameters based on the measured CBR values, preventing the channel from becoming overloaded.

The Metric Supervisor is configured through the main VaN3Twin script, described in Section 3.2.4, where the parameters for the calculation of communication metrics are defined, along with channel technology (i.e. 802.11p) and linkage with OpenCDA Client.

3.2.6 Facilities Layer

The Facilities Layer is a component of the ETSI protocol stack as part of the ITS station reference architecture, defined in [10]. This layer provides a set of functionalities and services (referred to as facilities) to support ITS applications for road safety, traffic efficiency, and others [12]. It can be seen in Figure 3.1 that the Facilities layer sits between the Applications layer, which contains the actual ITS application (e.g. collision warning, cooperative perception) and the Network and Transport layer, which handles the delivery of data over the V2X communication channels.

In practice, the Facilities layer provides support for the encoding/decoding of standardized ETSI messages, for data management and collection such as the LDM, and many other services related to communication. With this, applications can use facilities as standardized building blocks, focusing on higher-level logic without the need to implement the underlying communication functions.

Among the facilities implemented in the VaN3Twin framework, three are particularly relevant to this work: the LDM, the Collective Perception (CP) Basic Service, and the VRU Basic Service.

Collective Perception Basic Service

The CP Basic Service is responsible for the generation, encoding, and dissemination of CPM messages, in accordance with the standard ETSI TS 103 324 [18]. It processes information stored in the VaN3Twin LDM and prepares the data structure required for CPM transmission using the ASN.1 UPER encoding [25, 26], ensuring interoperability with standardized message formats.

Originally, this facility supported only the inclusion of vehicle detection in CPMs. As part of this work, this module was extended to support the inclusion of pedestrians and other types of VRUs, such as bicycles and motorcycles, as well as pedestrian groups. According to the ETSI standard, pedestrian groups fall into the same category as individual pedestrians and are therefore subject to the same inclusion rules for CPM generation, which are the following [18]:

i. *"The object has first been detected by the perception system after the last CPM generation event."*

ii. *"If the object list contains at least one object of Type-A which has not been included in a CPM for a time equal or larger than 500 ms, all objects of Type-A should be included in the currently generated CPM."*

Due to the LDM implementation, in which single objects and clusters are stored as separate entities, additional logic was implemented to properly handle their distinction during the CPM generation phase. For each detected pedestrian, it is determined whether that object is part of a cluster. If so, the cluster is evaluated for insertion into the CPM, rather than the individual pedestrian. This ensures that no redundant information is included in the transmission and contributes to a more compact CPM structure.

Furthermore, the CP service is configured to log the relevant information for each CPM into a CSV file. This includes the timestamp of the message, the number of perceived objects, the number of included VRUs and their corresponding ID, the number of clusters and their respective cardinalities, and the total encoded message size. These logs support the offline analysis of CPM load and content variation carried out in this work.

LDM

The LDM implemented in the VaN3Twin framework is responsible for storing and maintaining a dynamic representation of the surrounding environment. Unlike the LDM implemented in OpenCDA, which integrates perception and V2X data, the VaN3Twin

LDM acts as a storage layer, carrying information already processed and updated within the OpenCDA framework.

As part of this work, the VaN3Twin LDM has been extended with a cluster fusion algorithm suited for pedestrian clusters. Since locally perceived clusters are redetected at every simulation step, this fusion logic ensures a consistent temporal representation of clusters by determining whether a newly detected cluster corresponds to an existing one already stored in the cluster map. This enables the system to maintain a consistent cluster representation over time, contributing to a more reliable shared perception.

The cluster fusion works in a few steps. At first, the current position of each cluster stored in the LDM is estimated using a simple motion model, based on its last known velocity and the time elapsed since it was detected or updated.

Next, a matching procedure is performed to associate newly detected clusters with stored ones. For each candidate pair, a set of matching parameters is computed. These include the Euclidean distance between centroids, actor ID overlap, Intersection over Union (IoU) of cluster areas, and the difference in circular mean heading. Clusters are considered a match when their circular mean heading difference is less than 60 degrees, and at least one more parameter that fulfills the matching condition is met, being the centroid distance less than five meters, or if the areas intersect, or if there is an overlap on the list of IDs.

These conditions are designed to cover cases such as partial overlap, a slight shift in cluster positioning, or shared members between clusters, ensuring that multiple detections of the same group are correctly merged.

Once a match is confirmed, both clusters merge into a single entity. The resulting cluster is reshaped with a new minimum enclosing circle that encompasses all the actors from both clusters. When the matching process is over for all the new cluster entries, the cluster map stored in the LDM goes through the same matching process, but against itself, to search for duplicates.

The main idea of the fusion algorithm is to support a continuous tracking of a cluster, minimizing loss of information. It does so by flexibly allowing the algorithm to detect different segments of the same cluster and then merge them into a unified representation. Figure 3.6 illustrates the concept behind this fusion mechanism. Moreover, the logical steps of the process are described in Algorithms 1 and 2.

VRU Awareness Basic Service

The VRU Basic Service is responsible for generating and disseminating VAM messages, according to ETSI standards [17]. In this work, the existing implementation, originally designed to interface with SUMO through the TraCI API for mobility data, was adapted instead to interface with CARLA. The new implementation relies on the OpenCDA VRUDP module 3.2.2 to retrieve real-time mobility data for VRUs directly from CARLA, through the OpenCDA Client.

The service runs a periodic check to determine whether a VAM should be generated for each VRU. The periodicity of this check is set to 100 milliseconds, the minimum value allowed by the standards. Following the ETSI rules, these checks include detecting a

Algorithm 1 Cluster Fusion Algorithm - Main Procedure

Input: $\mathcal{C}_{\text{new}}$: Set of newly detected clusters
Input: $\mathcal{C}_{\text{stored}}$: Set of stored clusters (e.g., from LDM)
Output: $\mathcal{C}_{\text{stored}}$: Updated set of clusters

procedure FUSECLUSTERS($\mathcal{C}_{\text{new}}, \mathcal{C}_{\text{stored}}$)
  for all cluster $c_{\text{s}} \in \mathcal{C}_{\text{stored}}$ **do** ▷ *Step 1: Predict new positions of stored clusters*
    $\Delta t \leftarrow$ CurrentTime $- c_{\text{s}}$.last_predicted_timestamp
    Predict and update c_{s}.center using c_{s}.speed, c_{s}.heading, and Δt
    c_{s}.last_predicted_timestamp $\leftarrow$ CurrentTime
  end for

  $\mathcal{M}_{\text{new\_matched}} \leftarrow$ FindAndMerge($\mathcal{C}_{\text{new}}, \mathcal{C}_{\text{stored}}$, 'match') ▷ *Step 2: Match new clusters with stored clusters and merge them*

  for all cluster $c_{\text{new}} \in \mathcal{C}_{\text{new}}$ **do**▷ *Step 3: Add new clusters that were not matched*
    if $ID(c_{\text{new}}) \notin \mathcal{M}_{\text{new\_matched}}$ **then**
      Generate a new unique ID for c_{new}
      Add c_{new} to $\mathcal{C}_{\text{stored}}$
    end if
  end for

  $\mathcal{C}_{\text{to\_remove}} \leftarrow$ FindAndMerge($\mathcal{C}_{\text{stored}}, \mathcal{C}_{\text{stored}}$, 'deduplicate') ▷ *Step 4: Find and remove duplicate clusters from the stored set*
  Remove all clusters with IDs in $\mathcal{C}_{\text{to\_remove}}$ from $\mathcal{C}_{\text{stored}}$

  for all cluster $c \in \mathcal{C}_{\text{stored}}$ **do** ▷ *Step 5: Refine remaining clusters*
    repeat
      outlier_removed_this_iteration $\leftarrow$ false
      Find member m_o in c with max heading deviation from c.heading
      if deviation of $m_o >$ HEADING_DEVIATION_THRESH **then**
        Remove m_o from c
        Recalculate c.heading from remaining members
        outlier_removed_this_iteration $\leftarrow$ true
      end if
    until outlier_removed_this_iteration is false
    if cardinality of $c <$ MIN_PED_PER_CLUSTER **then**
      Mark c for removal
    end if
  end for
  Remove all marked clusters from $\mathcal{C}_{\text{stored}}$
end procedure

Algorithm 2 Cluster Fusion Algorithm - Find and Merge Helper

procedure FINDANDMERGE($\mathcal{C}_{\text{source}}, \mathcal{C}_{\text{target}}$, mode)
  $\mathcal{M}_{\text{matched\_source\_IDs}} \leftarrow \emptyset$
  for all cluster $c_s \in \mathcal{C}_{\text{source}}$ **do**
    found_match $\leftarrow$ `false`
    best_match_target_id $\leftarrow$ `null`
    for all cluster $c_t \in \mathcal{C}_{\text{target}}$ **do**
      if mode = `'deduplicate'` **and** $ID(c_s) \geq ID(c_t)$ **then**
        continue ▷ Avoid self-comparison and double-checking
      end if
      Compute *id_overlap*, *iou*, *dist*, *head_diff* between c_s and c_t
      if *head_diff* $\leq$ MAX_HEADING_DIFF **and** *id_overlap* $>$ MIN_ID_OVERLAP_RATIO **then**
        found_match $\leftarrow$ `true`
        best_match_target_id $\leftarrow ID(c_t)$
        break
      else if *head_diff* $\leq$ MAX_HEADING_DIFF **and** *iou* $\geq$ MIN_IOU **then**
        found_match $\leftarrow$ `true`
        best_match_target_id $\leftarrow ID(c_t)$
        break
      else if *head_diff* $\leq$ MAX_HEADING_DIFF **and** *dist* $\leq$ MAX_DISTANCE_THRESHOLD **then**
        found_match $\leftarrow$ `true`
        best_match_target_id $\leftarrow ID(c_t)$
        break
      end if
    end for
    if *found_match* **then**
      ▷ A match was found; fuse the source cluster into the target cluster
      Let c_{match} be the cluster in $\mathcal{C}_{\text{target}}$ with ID *best_match_target_id*
      Update c_{match}'s enclosing circle based on both clusters
      Update c_{match}'s heading (e.g., weighted circular mean)
      Merge member IDs: c_{match}.IDs $\leftarrow c_{\text{match}}$.IDs $\cup\, c_s$.IDs
      Update c_{match}'s cardinality and timestamps
      Add $ID(c_s)$ to $\mathcal{M}_{\text{matched\_source\_IDs}}$
    end if
  end for
  return $\mathcal{M}_{\text{matched\_source\_IDs}}$
end procedure

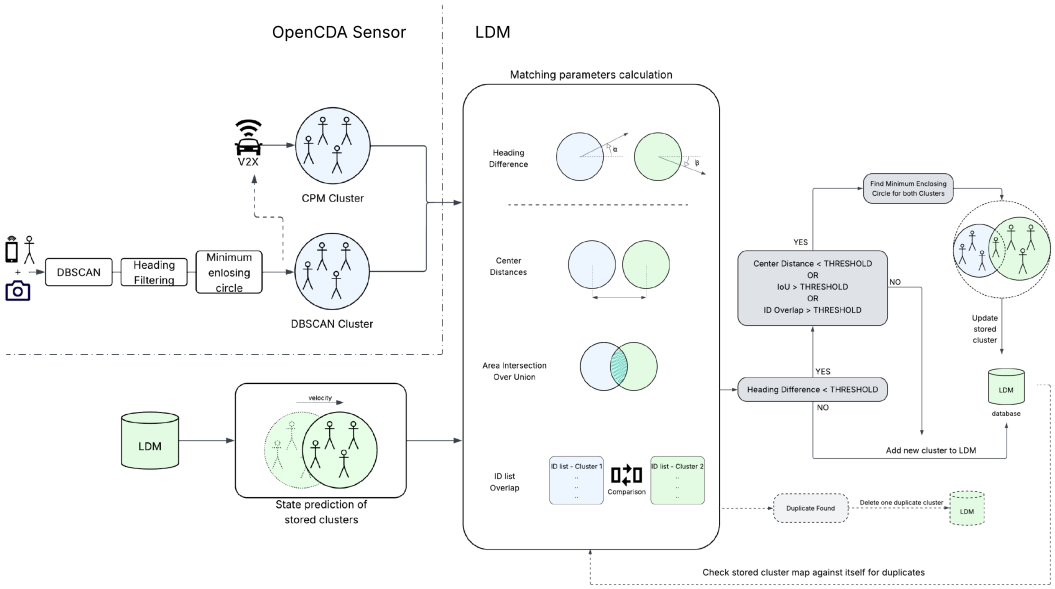


Figure 3.6: Diagram illustrating the pedestrian cluster fusion algorithm implemented in the VaN3Twin LDM. At each simulation step, newly detected clusters (blue circles) from the OpenCDA Sensor are matched against stored and predicted clusters (green circles). Centroid distance, actor ID overlap, Intersection over Union, and heading similarity are the criteria for a match. When there is a match, a new minimum enclosing circle is computed for the merged cluster. The updated cluster map goes through the same matching process for duplicate detection.

significant change in position, velocity, or heading, as well as the time elapsed from the last message sent. When a condition is satisfied, the service prepares the data structure with all mandatory and, if available, optional VAM fields and encodes it using ASN.1 [25], which is compliant with standard message formats.

This module is also configured to log detailed information about each transmitted VAM into a CSV file. This includes the timestamp of the generated message, the triggering condition, and the mobility data included in the awareness message, such as velocity, heading, and position.

Chapter 4

Experimental Setup and Simulation Scenarios

4.1 Objectives of the simulation

The primary objective of the simulation experiments conducted in this work is to evaluate the extended cooperative perception framework, with a focus on Vulnerable Road Users, particularly pedestrians. Two main goals can be underlined.

First, the objective is to assess the impact of representing pedestrian groups as clusters, where a cluster is a single perceived object, versus reporting individual pedestrians when generating a CPM. This comparison focuses on the CPM size under both approaches and the influence it may have on CBR values.

Second, the effect of introducing connected VRUs into the network is investigated by assigning ns-3 nodes to pedestrians, with a variable penetration rate (i.e., some pedestrians can send VAMs, while others remain passive) and pedestrian density. The analysis of the CBR values is done under each configuration.

Since the two experiments share most of the simulation parameters, this chapter is organized as follows:

- The first part 4.2 outlines the common simulation setup applied to all analyses in this work;
- The second part 4.3 describes the experiment evaluating the effect of including clusters in CPMs;
- The third part 4.4 is the experimental setup for the investigation of the influence of connected VRUs in the network.

4.2 Simulation Environment

The experimental setup is built on top of the co-simulation framework detailed in Chapter 3, combining CARLA for traffic and sensor simulation with VaN3Twin for V2X

communication.

The mobility side of the simulation is set up according to OpenCDA's documentation [49]. A default YAML file is provided, which can be used across different simulation scenarios. In this file, the synchronization mode, the simulation time step, the weather conditions, vehicle parameters, the lidar sensor configuration, and the traffic manager parameters are specified. Since the simulation involves sensor-based perception and runs in parallel with another simulation tool, VaN3Twin, there is a need for synchronization between the tools. Therefore, the synchronization mode is set to true (parameter *sync_mode* in the default YAML file). This means that both CARLA and VaN3Twin need to complete all their computations (perception, traffic management, and communication processes) before advancing to the next simulation step. This is to avoid that the mobility and network simulations advance in different rates, leading to the exchange of inconsistent data (data from different simulation instants). The fixed time step is set at 0.05 seconds, while other parameters such as weather conditions, vehicle dynamics, and traffic manager settings remain at their default values, as defined in the OpenCDA default YAML file [30].

The LiDAR configuration was modified to emulate a more realistic sensor, and for that, the OS1 LiDAR sensor [38] from Ouster was chosen. The OS1 LiDAR is a mid-range high-resolution sensor currently used by some companies in the field of autonomous shuttles and trucking. Taking into account the possible configurations of the real sensor, the cloud point rate for the simulation was set to 655.360 points per second. This value would correspond to a horizontal resolution of 1024 pixels, considering that the rotation frequency is 20 Hz, aligned with the frequency of the simulation. The horizontal resolution corresponds to the number of points in a 360-degree horizontal sweep, and in this case, 1024 lies in the middle resolution range of the configurable sensor. The vertical field of view was rounded down from 42.4 degrees of the real sensor to a total of 40 degrees, -30 degrees as the lower boundary value, and +10 degrees as the upper one. The number of channels is set to 32, being the lowest possible configuration of the OS1 LiDAR, and a conservative range of 80 meters was chosen.

All simulations were carried out under clear, sunny weather, with no precipitation or clouds. The parameters common to all simulation runs, not related to mobility, are summarized in Table 4.1.

Scenario creation and actor definition are defined in a second YAML file, called after the simulation script file. This file specifies a set of parameters related to the simulation environment itself. They include the spawn and destination positions of all CAVs, along with the definition of the number and placement of cameras, the number of background vehicles, and the spawn position of pedestrians, individual or clustered, and the map/town to be used in the simulation. In CARLA, a map (or town) is a predefined 3D environment where simulation takes place. Each map provides different urban features, including road layouts, intersections, buildings, and other environmental elements. Each CAV is equipped with one 360-degrees LiDAR sensor and four RGB cameras, each with a default 90-degree field of view, mounted in a way to provide full coverage. For each experiment, two CAVs are placed on the same road but facing

Parameters	Values
CARLA	
CARLA version	0.9.12
Synchronization mode	True
Fixed delta time step	0.05 seconds
Weather (sun angle)	15 degrees
LIDAR	
Channels	32
Range	80
Cloud point rate	655.360 points/second
Frequency	20 Hz
Upper field of view	10 degrees
Lower field of view	-30 degrees
RGB CAMERAS	
Quantity	4
Positions [x, y, z, yaw]	[2.5, 0, 1.0, 0] [0.0, 0.3, 1.8, 100] [0.0, -0.3, 1.8, -100] [-2.0, 0.0, 1.5, 180]
Other attributes	Default

Table 4.1: Important common parameters for the simulation defined in the YAML files

opposite directions, and the number of background vehicles varies from 0 to 20. The vehicle penetration rate is fixed at 0.7 for all simulations, reflecting a high adoption scenario. Although connected vehicle deployment is still in an early stage, it is reasonable to assume such penetration in the near future. Moreover, recent simulation studies [55] support that optimal safety performance (in terms of collision avoidance) is achieved at 70% vehicle penetration rate. The number of VRUs varies from 10 to 150, with penetration rates of the VRUs between 0 and 0.8.

The OpenCDA framework also provides a behavior agent module for local and global path planning, and a control manager module, which translates the desired target value into compliant inputs such as steering, throttle, and brake. However, the overall traffic flow in the simulation is still managed by CARLA's Traffic Manager (TM). The TM is the module responsible for coordinating and controlling the movements of all vehicles operating in *autopilot* mode, ensuring realistic urban traffic conditions. For that, both background vehicles and CAVs are set to autopilot, which assigns them to the TM. In addition to that, specifying the same communication port when enabling autopilot ensures that all vehicles are linked to a single TM instance, which manages their behavior throughout the simulation. This module supports high-level user customization through parameters that can allow or encourage specific behaviors. However, in this study, these parameters were left as their default values, as the focus is not on analyzing vehicle

behavior under varying traffic conditions. Table 4.2 specifies the number of background vehicles, CAVs, and pedestrians used in the experiments.

Parameters	**Values**
SCENARIOS	
Town	Town10HD
Background vehicles	20
Vehicle Penetration Rate	0.7
CAVs	2
Pedestrian number	10 to 150
VRU Penetration Rate	0.0 to 0.8

Table 4.2: Overall mobility configuration for all the experiments

On the VaN3Twin side, all aspects of the communication configuration for the simulation are configured in a dedicated ns-3 script written in C++, detailed in Section 3.2.4. This script defines the parameters of the OpenCDA and CARLA interface, the access layer technology, the transmission power, the data rate, and the penetration rates for vehicles and VRUs. It also configures the metric supervisor module for the collection of V2X metrics, such as the packet received ratio (PRR), latency, and channel busy ratio (CBR), as described in Section 3.2.4. Additionally, this script configures the installation of the cooperative perception application on each dynamic ns-3 node, as explained in Section 3.2.4. Through the *cooperativePerceptionHelper*, a helper interface that simplifies the configuration process, attributes such as the OpenCDA Client instance, real-time execution flag, output logging options, communication model and metric supervisor linkage are set. Once these configurations are set, a method that establishes the connection with CARLA is called. This is a central link between the network simulator and the traffic simulator, as it sets up the gRPC bridge with CARLA and triggers the start of the simulation, using a dedicated callback function for node setup. A callback function is a function that is executed automatically after a specific event, in this case, when a node is created at the start of the simulation. More specifically, at the start of the simulation, all existing actors in the CARLA environment are retrieved, and, for each of these actors, the callback function is triggered. The callback assigns each node to a corresponding actor ID and sets its station type as either a pedestrian or a passenger car. Once configured, the application is installed on the node.

Regarding access technology, the simulation uses IEEE 802.11p [1], given its established role as a standard for DSRC within ITS. The PHY layer is configured with a fixed transmission power of 30 dBm, in light of the fact that this should take into account a simulated OBU 802.11p chipset and a simulated antenna. The chosen transmitting power is a reasonable value considering that a real OBU may have a chipset capable of transmitting at 24 dBm, and on top of that, an automotive antenna may have a gain of about 6 dBi, which focuses and concentrates the 24 dBm signal generated by the OBU chipset. This results in an Effective Isotropic Radiated Power (EIRP) of about 30 dBm, which corresponds to the total power transmitted from the antenna. For the

data rate, a fixed value of 3 Mbit/s was chosen, which is one of the ones suggested by the standard [11]. This is considered the most reliable option for vehicular communication. Even though IEEE 802.11p allows for data rates up to 27 Mbit/s, higher rates require a more complex modulation scheme. Although these modulation schemes increase data throughput, they are more susceptible to signal fading and interference, which are common challenges in a dynamic environment. On the other hand, a data rate of 3 Mbit/s utilizes a simpler and more robust modulation, such as Binary Phase-Shift Keying (BPSK), which ensures a higher probability of message success reception, crucial for safety-related communication [50]. For the CBR calculation, the measurement window is set to 200 milliseconds. So, the CBR is calculated every 200 ms with the Active and Busy time related to this time window. A lower value for the measurement window might capture only transient bursts, and a too-long window might smooth out important fluctuations. The exponential moving average of the CBR is computed with an alpha value of 0.1, which gives more importance to recent measurements and makes the metric more responsive to current conditions.

For both experiments, relevant metrics such as CPM message size and Channel busy ratio (CBR) are recorded through the metric supervisor module and other facilities on VaN3Twin. These results are further aggregated and post-processed using Python scripts to compare performance with different simulation configurations.

Parameters	**Values**
VaN3Twin	
Access technology	802.11p
Transmission power	30 dBm
Center Frequency	5.9 GHz @ 10 MHz
Propagation Loss Model	Log distance (Default)
Physical Data Rate	3 Mbit/s
CBR measurement window value	200 milliseconds
CBR alpha	0.1

Table 4.3: VaN3Twin configuration parameters

4.3 Pedestrian cluster impact on CPM

This experiment aims at evaluating the impact of representing multiple objects (e.g. pedestrians) as a single entity when transmitting CPMs, in terms of message size and channel load. The cluster algorithm is explained in Section 3. For this analysis, the spawned pedestrian were not equipped with communication capabilities, therefore, they are not active in the simulation.

For the scenario setup, three main areas were defined for the spawning of pedestrians. The first area is determined to be right next to the Ego vehicle (CAV1) and the other two are located further away from the Ego vehicle, on its path, one on each side of the road. In this way, there are two moments in the simulation in which the vehicle will

be near a group of pedestrians, as soon as the simulation starts and close to the end of it. Figure 4.1 exhibits the bird eye view of the ego vehicle, its trajectory, and the areas designated to spawn pedestrians.

The simulation duration is set to 15 seconds. Three levels of pedestrian density were tested, 10, 20, and 30, for each spawned region. The radius of each region is set to a fixed value of 15 meters and in all conditions, the pedestrians are set to have the same destination point (those within the same area). Table 4.4 summarizes the setup for the scenarios. For each pedestrian density, two configurations were tested: i)Without clustering, including individual detections in the CPM; ii) With clustering, allowing the algorithm to form a cluster and include it in the CPM if conditions are met. For each different scenario, three independent simulation runs were performed in order to reduce randomness and increase the reliability of the outcomes.

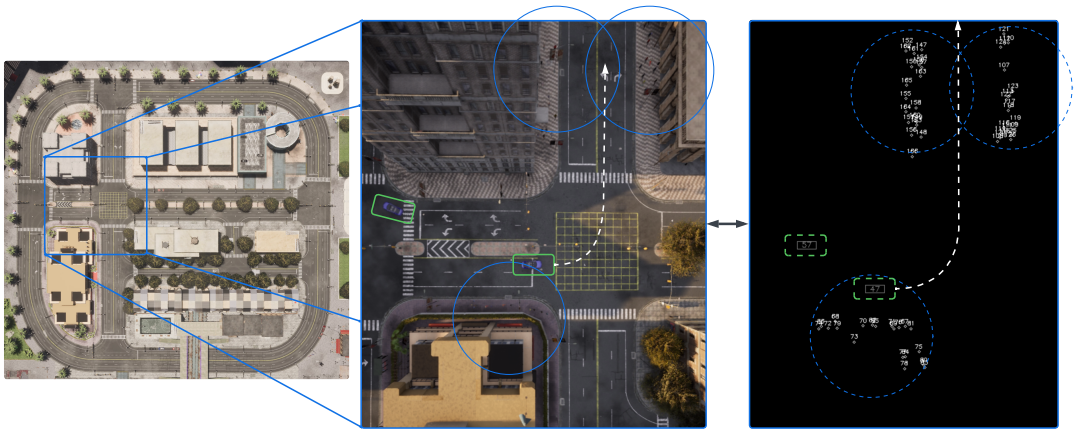


Figure 4.1: Bird-eye view of the map used in the simulation. Left figure is a static picture of Town10HD. Figure in the middle is the spectator view during simulation. Figure on the right is the OpenCDA Sensor GUI output with ground truths. The three blue circles refers to the spawn location of pedestrians. The two green rectangles refers to the spawned CAVs. White dashed line is the trajectory of the Ego Vehicle.

		Common to all scenarios	
	Pedestrians per area	**Number of spawn areas**	**Radius of each area (m)**
Scenario A	10		
Scenario B	20	3	15
Scenario C	30		

Table 4.4: Pedestrians setup for the different scenarios.

CPMs are generated quasi-periodically, following the ETSI specifications for the inclusion of VRUs and other Type-A objects in the CPM, introduced in Section 3.2.6. If

no object is detected or if the detected ones do not meet the criteria to be included in a message, a CPM is generated at a frequency of 1 Hz, as a "presence signal", containing only basic information from the vehicle and information about the sensor system.

To access the effect of clustering, a few metrics were considered, the CPM size over the entire simulation, and CBR values across all nodes. It is worth mentioning that the size of CPMs presented here, in bytes, refers purely to the compressed payload, not the entire network packet.

The results are summarized in Table 4.5, which reports the average CPM size, and average and maximum CBR for each condition. As expected, a higher density of pedestrians increases the size of the CPM, while clustering reduces the number of entities to be transmitted, leading to smaller CPM.

Scenario		**Clustering**	**No Clustering**
A	CPM size - average	126.15	158.62
	CBR - average	0.006	0.0062
	CBR - max	0.0139	0.0141
B	CPM size - average	118.10	270.72
	CBR - average	0.0062	0.0077
	CBR - max	0.0164	0.0204
C	CPM size - average	112.23	308.82
	CBR - average	0.0071	0.0081
	CBR - max	0.019	0.0216
Overall	CPM size - average	118.15	250.48
	CBR - average	0.0064	0.0073
	CBR - max	0.019	0.0216

Table 4.5: CPM size average, CBR average and maximum, for all the scenarios.

Figure 4.3 shows the evolution of the CPM size over time for all scenarios. Without clustering, the CPM size increases rapidly and fluctuates more. With clustering, the message size remains both smaller and stable. The distribution of CPM size, including data from all scenarios and all simulation runs for this experiment, is shown in Figure 4.4. The plot shows that clustering reduces both the spread and the average message size. The impact of clustering on CBR is presented in Figure 4.2, which shows a small but measurable difference in CBR values. Since there are only two nodes transmitting in all scenarios, the Ego vehicle and a secondary CAV, the CBR values remain really low overall. Nonetheless, for all scenarios where clustering is active, the box plot size remains relatively constant, indicating that the CBR values are consistent and stable, regardless of pedestrian density. This result is aligned with the CPM size plots, where a higher spread of the CPM size values leads to an increase in the variability of CBR.

Overall, these results provide clear evidence of the gains obtained through clustering, particularly in more dense environments where the reduction of CPM size is more relevant. These results are further discussed in Section 5.

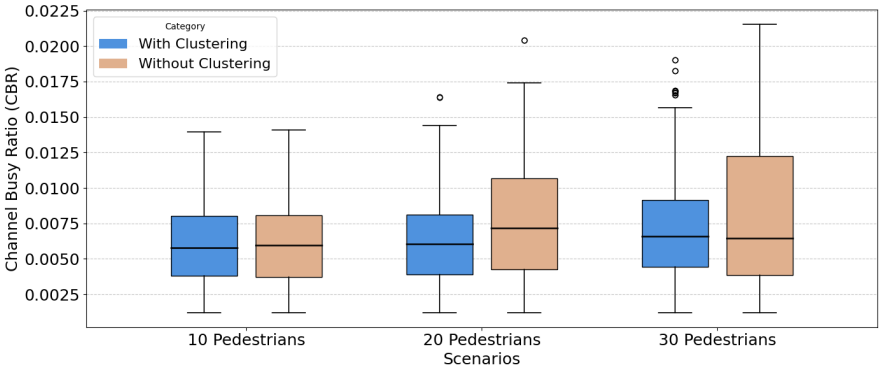


Figure 4.2: CBR values with and without clustering algorithm active, for all three scenarios

4.4 Connected VRUs impact on CBR

This set of experiments investigates the influence of connected VRUs on channel load by measuring the Channel Busy Ratio under different VRU penetration rates and pedestrian density. In a preliminary configuration, the data rate was varied across 3, 6, and 12 Mbit/s with 150 pedestrians randomly spawned within a 70-meter radius, centered around the first CAV. This radius was selected to ensure that throughout the simulation, the CAV remains within the same area as the pedestrians, maximizing the exchange of V2X messages. The background vehicles configuration is the same as the main analysis (Table 4.6), with 20 spawned vehicles and a penetration rate of 0.7. The penetration rate of VRUs ranges from 0 to 0.8, with an increment step of 0.1. The results of this preliminary configuration are illustrated in Figure 4.6

The main analysis instead employs a fixed data rate of 3 Mbit/s, which is the most robust and reliable rate for safety-critical communications. With this data rate, the pedestrian density is varied across 50, 100, and 150 pedestrians, spawned randomly in the same 70-meter radius area. For each density of pedestrians, the penetration rate of VRUs ranges from 0 to 0.8, with an increment step of 0.2. In all configurations, 20 background vehicles were spawned, with a fixed penetration rate of 0.7. Table 4.6 summarizes the configuration parameters for this analysis.

Pedestrians are controlled by CARLA's Traffic Manager, and each of them has a randomly assigned destination.

The CBR is calculated for all connected nodes in the simulation, using a exponential moving average with the parameters defined in Section 4.2. The values presented in this analysis correspond to the average CBR over the entire simulation span.

Certain VAM triggering conditions were beyond the scope of this study. This implementation did not consider (i) variations in the interception probability with another vehicle or VRU, (ii) decisions related to joining or forming a cluster, and (iii) situations where another vehicle or VRU is located within a determined safe distance, both laterally and longitudinally. All connected VRUs are equipped with the same communication

		Common to all scenarios			
	Number of Pedestrians	**VRU penetration rate**	**Radius of spawn area (m)**	**Number of Background vehicles**	**Vehicle penetration rate**
Scenario D	50	0.0 to 0.8 (step 0.2)	70	20	0.7
Scenario E	100				
Scenario F	150				

Table 4.6: Pedestrians setup for the different scenarios.

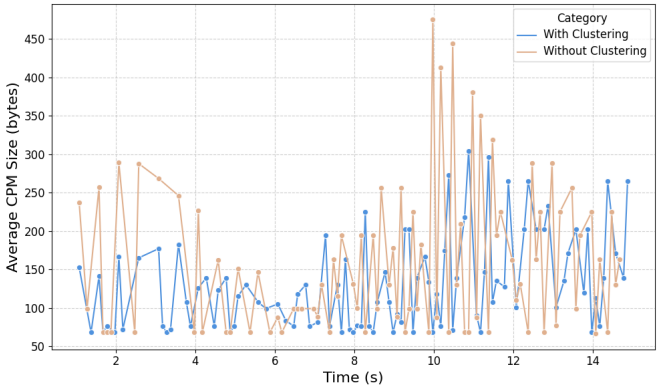
capabilities.

In addition to VRU transmissions, CAVs broadcast CAMs and CPMs, while background connected vehicles transmit only CAMs. Both of these awareness messages are transmitted with a frequency between 1 Hz and 10 Hz, according to the conditions presented by ETSI [14, 18].

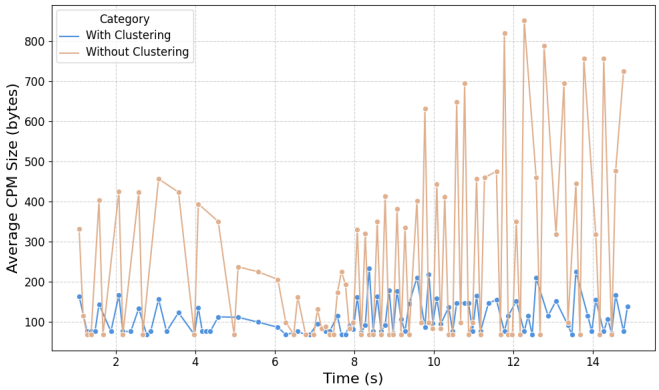
Table 4.7 presents the resulting CBR values for this analysis, with the conditions described in Table 4.6, and Figure 4.5 illustrates the numerical results.

VRU	**50 pedestrians**		**100 pedestrians**		**150 pedestrians**	
Rate	mean	max	mean	max	mean	max
0.0	0.0186	0.0392	0.0184	0.0393	0.02	0.0395
0.2	0.0249	0.0437	0.0283	0.0578	0.0329	0.0657
0.4	0.0239	0.0406	0.031	0.0633	0.05	0.0854
0.6	0.0264	0.0433	0.0451	0.0773	0.0633	0.1221
0.8	0.038	0.0619	0.0589	0.1056	0.0779	0.1255

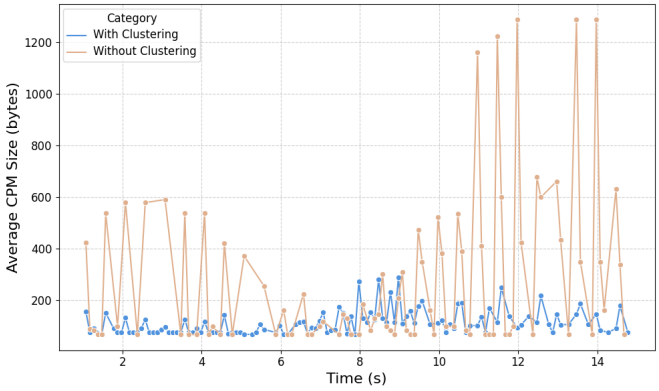
Table 4.7: Summarized CBR values. Mean, and maximum values for CBR vs VRU penetration rate, for each density configuration



(a)



(b)



(c)

Figure 4.3: CPM size over simulation time for all three scenarios, with and without clustering algorithm active. (a) Scenario A - 10 pedestrian; (b) Scenario B - 20 pedestrians; (c) Scenario C - 30 pedestrians

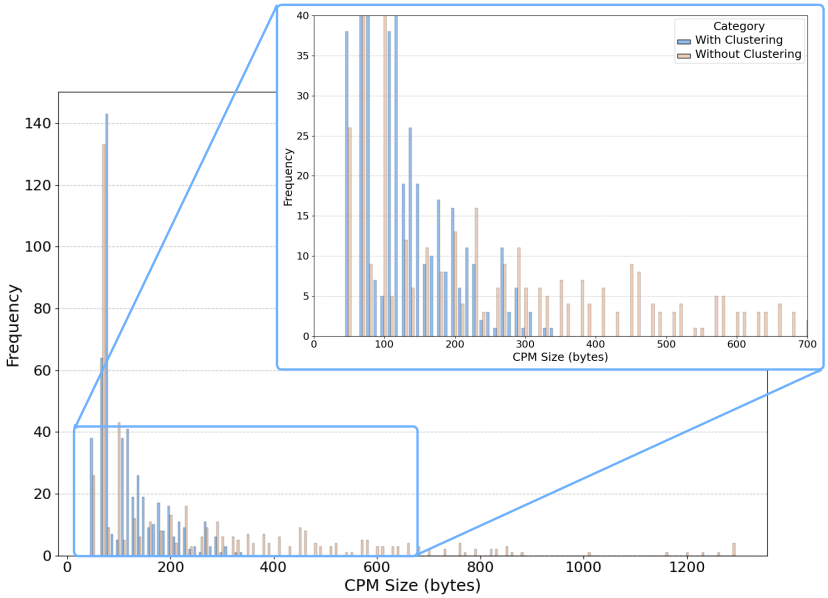


Figure 4.4: Frequency versus CPM size for both conditions. Data from all three scenarios.

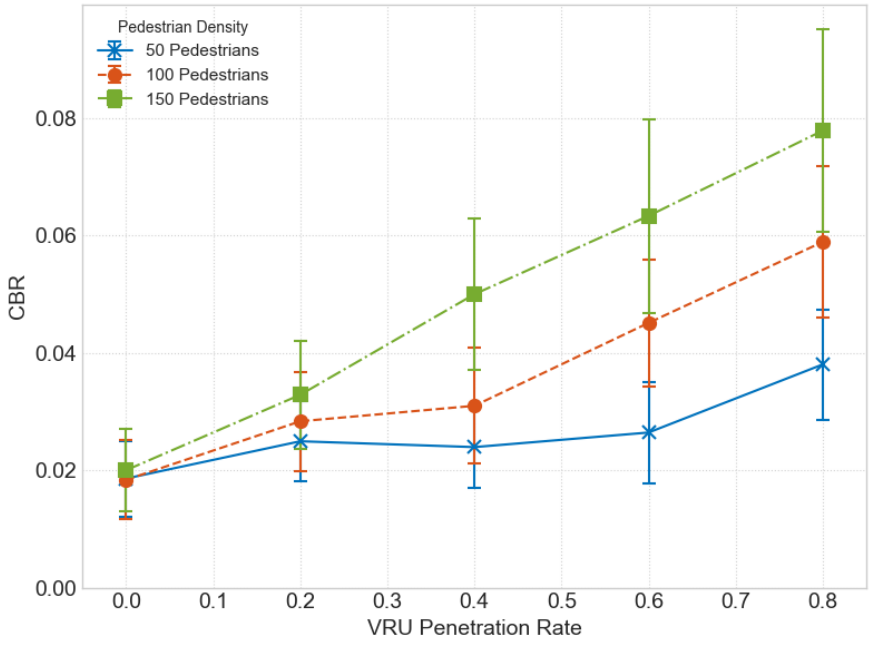
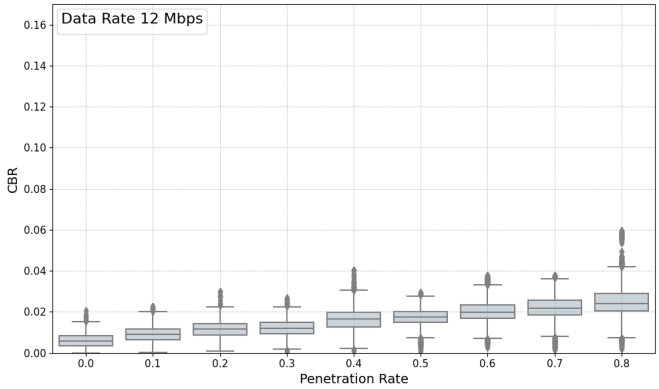
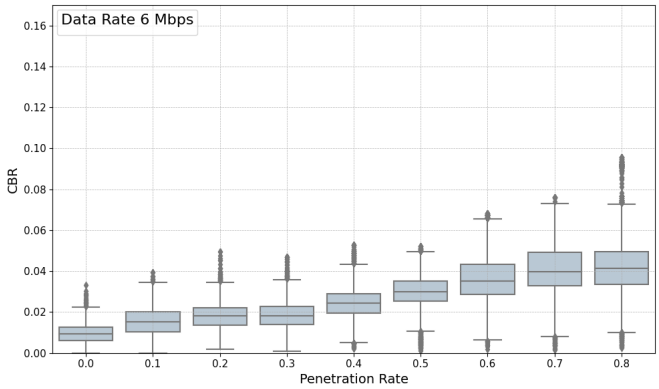


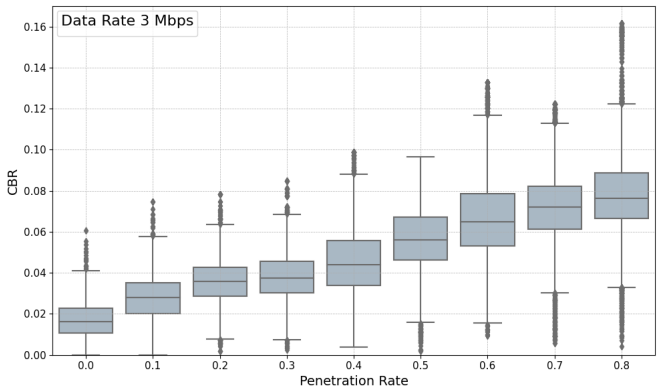
Figure 4.5: CBR versus VRU penetration rate for different pedestrian densities



(a)



(b)



(c)

Figure 4.6: CBR vs VRU penetration rate plots for different Data Rates: (a) 12 Mbps (b) 6 Mbps (c) 3 Mbps

Chapter 5

Experimental Results

The discussion of the results is structured into two parts, (i) addressing the impact of clustering on CPM messages and (ii) the effects of having connected VRUs on channel load.

5.1 Clustering and its Effect on CPMs

This experiment evaluates the impact of representing multiple objects (pedestrians) as a single entity when transmitting CPMs. The simulation results confirm that clustering reduces CPM size significantly, particularly in a higher density scenario. By compressing the representation from multiple entities to just one or a few, not only reduces the CPM size but also reduces the variability across simulation time. These findings align with recent studies that investigated clustering as a method to address scalability challenges for cooperative perception. Pedestrians were not equipped with communication capabilities, therefore they did not actively transmit in the simulation.

Three spawn areas were defined (Figure 4.1), one close to the Ego CAV and two further along its trajectory. Densities of 10, 20, and 30 pedestrians per area were tested (Table 4.4). Each simulation was run for 15 seconds, with three repetitions per scenario to reduce randomness. For each density, two conditions were compared: (i) no clustering, transmitting individual detections; and (ii) clustering, transmitting circular-shaped cluster when conditions are met. CPMs are generated following ETSI specifications, described in Section 3.2.6.

Figures 4.3 and 4.4 confirm that clustering reduces both the average and variability of CPM size. Table 4.5 summarizes the results. Clustering consistently reduces CPM size across scenarios:

- Scenario A: from 158.62 to 126.15 bytes (-20.5%)
- Scenario B: from 270.72 to 118.10 bytes (-56.4%)
- Scenario C: from 308.82 to 112.23 bytes (-63.7%)

- Overall: from 250.48 to 118.15 bytes (-52.8%)

Xhoxhi et al. [53] demonstrated that representing VRUs through geometric clustering will reduce, on average, one third of the transmitted data, in bytes per second, while preserving situational awareness. It further analyzes different enclosing shapes for clustering, concluding that a polygon shape is the most accurate in describing the cluster itself, however, is the less efficient in terms of data required for the shape description. Moreover, circular shape clustering is shown to be the least accurate, but the best in transmission efficiency, meaning that it requires less bytes to be included in a CPM. Therefore, clusters represented in this work can be considered the most efficient to be included in a CPM, since only circular shape was considered.

Lobo et al. [33] and Valle et al. [51] recently analyzed the use of clustering techniques as a way to optimize resource allocation, however, considering VAM-based clustering, not perception-based as it is in this work. The first showed that clustering allowed a significant reduction of CBR values from DCC limits, and lowered the position error. While the second introduces a non-negotiable clustering mechanism for VAM transmission, showcasing the reduction of total generated messages and the improved VRU awareness. Although different mechanisms were used in these studies, they further reinforce the benefits of clustering, considering it as a tool to handle high density VRU scenarios.

Without clustering, CPM size grows rapidly, reflecting many challenges highlighted by Hussein et al. [24] for large-scale deployment of VANET, such as network resource scarcity and communication overhead. By stabilizing message size, clustering contributes to more predictable channel access in contention-based MAC protocols such as IEEE 802.11p, reducing the probability of collisions caused by bursts of large packets. This stability is critical for an effective DCC, since its goal is to balance network load, and having large packet bursts can increase packet collision and retransmission.

For the CBR values, the results show that clustering had only a marginal effect, with overall average values (averaged across all scenarios) of 0.0064 vs 0.0073 between cluster and no-cluster scenarios, respectively. This small variation is given by the reduction in the average CPM size, since the channel load is mainly driven by the number of transmitting nodes, which was limited to two CAVs. Nevertheless, clustering reduces the variability in the CBR values across different pedestrian densities, mirroring the reduced variability in CPM size. Therefore, clustering helps stabilize both the CPM size and the channel load, which can be useful in denser scenarios.

Nonetheless, clustering trades off detail for efficiency in transmission, which may reduce usefulness in certain safety applications where a more precise knowledge of the individual pedestrians is required.

5.2 Impact of connected VRUs on channel load

The second experiment evaluates how connected VRUs impact channel load, that is, CBR, as a function of VRU penetration rate and pedestrian density, with a fixed data

transmission rate of 3 Mbit/s unless otherwise stated. Table 4.7 shows that for a given penetration rate, increasing pedestrian density increases the mean and peak CBR proportionally. Approximately, tripling pedestrian density roughly doubles CBR. For example, at 80% VRU penetration, the mean CBR increases from 3.8% (50 pedestrians) to 7.8% (150 pedestrians), and the maximum CBR from 6.2% to 12.5% (2x in both cases). With no connected VRUs, mean CBR remains near 2% across different pedestrian densities since the only active nodes transmitting are the two CAVs.

To complement this analysis, a separate configuration varied the data rate across 3, 6, and 12 Mbit/s (150 pedestrians within a 70 m radius), to analyze the effect on channel occupancy. As expected, increasing data rate significantly reduces the CBR, since higher rates shorten transmission time. As illustrated in Figure 4.5, moving from 3 Mbit/s to 12 Mbit/s reduces CBR roughly by three to four times. Although higher data rates lower CBR, they are less robust (due to more complex modulation), whereas 3 Mbit/s is commonly used for safety-related communication due to its reliability [50]

Overall, higher penetration rates naturally increase channel load, given the increased number of transmitting nodes, and a higher physical data rate reduces it, reflecting shorter transmission times. Nevertheless, even in the worst case, at 80% penetration rate with 150 pedestrians and a data rate of 3 Mbit/s, CBR values remained well below the critical values and stay in the *Relaxed* state, as defined by [21].

Chapter 6

Conclusions

This work investigates the integration of Vulnerable Road Users (VRUs) into cooperative perception within a co-simulation environment that combines CARLA, through OpenCDA, and VaN3Twin. The developed framework extends the original setup by enabling the perception, representation, and communication of VRUs in the simulation loop. Specifically, it (i) integrates pedestrian detection into the OpenCDA framework and connects this perception output with the facilities layer in ns-3 for the the transmission of Cooperative Perception Messages (CPMs); (ii) implements a new VRU data provider, integrated with the corresponding ETSI Facilities layer service (VRU Basic Service), to enable standard compliant VRU communication, through VAMs; (iii) evaluates the impact of pedestrian clustering for CPM inclusion. These contributions allow for a realistic and synchronized evaluation of perception-driven communication scenarios involving both connected and non-connected VRUs.

The experiments conducted throughout this study assess two main aspects: the impact of clustering on CPM generation and the influence of connected VRUs on channel load. The first experiment focuses on clustering as a mechanism to represent multiple perceived objects as a single group within CPMs, rather than transmitting them individually. The results show that this approach significantly reduces CPM size, by more than 50% on average, and stabilizes message variability over time. Although the Channel Busy Ratio (CBR) remains low, given the limited number of transmitting vehicles, a reduction in CBR variability is also observed. These results confirm that clustering improves the scalability of cooperative perception by limiting the increase in transmitted data as the number of perceived objects increases.

The second experiment analyzes the effect of connected VRUs on network load, considering variations in both pedestrian density and penetration rate. The results indicate that the mean and maximum CBR values increase proportionally with the number of connected VRUs, but remain within the safe operational limits defined by ETSI Decentralized Congestion Control (DCC). Even in the most demanding scenario, 150 pedestrians at 80% penetration and 3 Mbit/s data rate, the mean CBR remains below 8%, corresponding to the *Relaxed* DCC state. An additional configuration exploring higher data rates confirms the expected inverse relationship between transmission rate

and CBR, with values at 12 Mbit/s being approximately three to four times lower than at 3 Mbit/s. Nonetheless, 3 Mbit/s remains the most robust and reliable rate for safety-critical V2X communications.

In summary, the developed framework successfully integrates VRUs into cooperative perception co-simulation, providing a complete environment to evaluate mobility, perception and communication processes in a realistic setup. The results demonstrate that clustering is an effective strategy to reduce CPM size and stabilize network load, while connected VRUs scale channel usage predictably and remain within acceptable limits. Future work may extend these experiments to include larger and more diverse traffic conditions, other communication technologies such as C-V2X and NR-V2X, dynamic DCC control, ray tracing for channel propagation loss, and a full implementation of VAM triggers, according to ETSI. Incorporating real-world data or field validation could also enhance the realism of the simulation and support further analysis of safety-critical use cases involving pedestrians and autonomous vehicles.

Appendix A

VaN3Twin script setup code snippet

Here is presented a code snippet of the script setup for the VaN3Twin framework. It shows some defined parameters, the steps used to create and setup a node, together with initialization of the metric supervisor module and the setup of the OpenCDA Client module. It also shows the callback functions, used to install the applications on the nodes (*setupNewWifiNode*) and to stop the applications at the end of the simulation (*shutdownWifiNode*), which will be called by the OpenCDA client. The code snippet provided does not correspond to the full script required to setup the simulation.

```
/*** 0.a Mobility Options ***/
std::string opencda_folder = "Opencda/";
std::string opencda_config ="ms_van3t_example_ml"; //file to be run on OpenCDA side
bool opencda_ml = true;
/*** Some runtime parameters ***/
bool realtime = false;
// Output files
std::string csv_name_cumulative = "Metrics_PRR";
std::string csv_name_CBR = "Metrics_CBR";

int txPower=30;
double penetrationRate = 0.7;
double VRUpenetrationRate = 0.0;
float datarate = 3; //Mbit/s
bool send_cam = true;
double m_baseline_prr = 150.0;
bool m_metric_sup = true;
double simTime = 15; //Simulation time

/*** 0.c Read from OpenCDA config file the number of vehicles and pedestrians ***/
```

```
std::string config_yaml =
    OpenCDA_HOME+"/opencda/scenario_testing/config_yaml/" +
    opencda_config + ".yaml";

/*** 1. Create containers for OBUs ***/
NodeContainer obuNodes;
obuNodes.Create(numberOfNodes);

/*** 2. Create and setup channel   ***/
YansWifiPhyHelper wifiPhy;
wifiPhy.Set ("TxPowerStart", DoubleValue (txPower));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txPower));
YansWifiChannelHelper wifiChannel =
    YansWifiChannelHelper::Default ();
Ptr<YansWifiChannel> channel = wifiChannel.Create ();
wifiPhy.SetChannel (channel);

/*** 3. Create and setup MAC ***/
wifiPhy.SetPcapDataLinkType (YansWifiPhyHelper::DLT_IEEE802_11);
NqosWaveMacHelper wifi80211pMac = NqosWaveMacHelper::Default ();
Wifi80211pHelper wifi80211p = Wifi80211pHelper::Default ();
std::cout << "Datarate: " << datarate_config << std::endl;
wifi80211p.SetRemoteStationManager
    ("ns3::ConstantRateWifiManager",
                                    "DataMode", StringValue
                                       (datarate_config),
                                    "ControlMode", StringValue
                                       (datarate_config),
                                    "NonUnicastMode",StringValue
                                       (datarate_config));
NetDeviceContainer netDevices = wifi80211p.Install (wifiPhy,
    wifi80211pMac, obuNodes);


/*** 4. Give packet socket powers to nodes ***/
PacketSocketHelper packetSocket;
packetSocket.Install (obuNodes);

/*** 5. Setup Mobility and position node pool ***/
MobilityHelper mobility;
mobility.Install (obuNodes);

/*** 6. Setup OpenCDA client ***/
Ptr<OpenCDAClient> opencda_client = CreateObject<OpenCDAClient>();
opencda_client->SetAttribute("UpdateInterval", DoubleValue(0.05));
opencda_client->SetAttribute("PenetrationRate",
    DoubleValue(penetrationRate));
// ... additional attribute settings ...
opencda_client->SetAttribute("CARLAGUI", BooleanValue(carla_gui));
```

```
61 opencda_client->SetAttribute("ApplyML", BooleanValue(opencda_ml));
62
63 /*** Initializes Metric Supervisor ***/
64 metSup = CreateObject<MetricSupervisor>(m_baseline_prr);
65 metSup->setOpenCDAClient (opencda_client);
66 metSup->setChannelTechnology("80211p");
67 metSup->setCBRWindowValue(200);
68 metSup->setCBRAlphaValue(0.1);
69 metSup->setSimulationTimeValue(simTime);
70 metSup->setNodeContainer(obuNodes);
71 metSup->startCheckCBR();
72
73 /*** 7. Setup interface and application for dynamic nodes ***/
74 cooperativePerceptionHelper cooperativePerceptionHelper;
75 cooperativePerceptionHelper.SetAttribute("OpenCDAClient",
       PointerValue(opencda_client));
76 // ... other helper attributes ...
77 cooperativePerceptionHelper.SetAttribute("MetricSupervisor",
       PointerValue(metSup));
78
79 /* callback function for node creation */
80 STARTUP_FCN setupNewWifiNode = [&] (std::string actorID) ->
       Ptr<Node>
81 {
82   if (nodeCounter >= obuNodes.GetN()) {
83       NS_FATAL_ERROR("Node Pool empty!: " << nodeCounter << "
           nodes created.");
84   }
85     Ptr<Node> includedNode;
86     std::string number_str;
87
88     includedNode = obuNodes.Get(nodeCounter);
89   ++nodeCounter; // increment counter for next node
90
91     if (actorID.find("ped") != std::string::npos){
92         cooperativePerceptionHelper.SetAttribute("itsType",
              StringValue("StationType_pedestrian"));
93     }
94     else {
95         cooperativePerceptionHelper.SetAttribute("itsType",
              StringValue("StationType_passengerCar"));
96     }
97   /* Install Application */
98   ApplicationContainer AppSample =
       cooperativePerceptionHelper.Install (includedNode);
99   AppSample.Start (Seconds (0.0));
100  AppSample.Stop (simulationTime - Simulator::Now () - Seconds
       (0.1));
101
```

```
102   return includedNode;
103 };
104
105 /* Callback function for node shutdown */
106 SHUTDOWN_FCN shutdownWifiNode = [] (Ptr<Node> exNode,std::string
      actorID)
107 {
108   /* stop all applications */
109   Ptr<cooperativePerception> appSample_
        =exNode->GetApplication(0)->GetObject<cooperativePerception>
        ();
110   if(appSample_)
111     appSample_->StopApplicationNow();
112 };
113
114 /* start OpenCDA client with given function pointers */
115 opencda_client->startCarlaAdapter(setupNewWifiNode,
      shutdownWifiNode);
```

Bibliography

[1] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, 2012.

[2] Majeed Ahmad. Hil simulation is the new normal in v2x testing, 2018.

[3] AI4CCAM Consortium. AI4CCAM: Trustworthy AI for Connected and Cooperative Automated Mobility. https://www.ai4ccam.eu/, 2022–2025. Project website.

[4] Mani Amoozadeh, Bryan Ching, Chen-Nee Chuah, and Dipak Ghosal. Ventos: Vehicular network open simulator with hardware-in-the-loop support. *Procedia Computer Science*, 151:61–68, 01 2019.

[5] Zhengwei Bai, Guoyuan Wu, Matthew J Barth, Yongkang Liu, Emrah Akin Sisbot, Kentaro Oguchi, and Zhitong Huang. A survey and framework of cooperative perception: From heterogeneous singleton to hierarchical cooperation. *arXiv preprint arXiv:2208.10590*, 2022.

[6] C-Roads Platform. M4.3 Annual deployment overview report 2023. Technical report, C-Roads Platform, 2023.

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on Robot Learning*, pages 1–16. PMLR, 2017.

[8] Eleobert. dbscan, 2023.

[9] Chris Ernst. dbscan-python, 2019.

[10] ETSI. Intelligent Transport Systems (ITS); Communications Architecture. Standard ETSI EN 302 665 V1.1.1 , 2010.

[11] ETSI. Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band. Standard ETSI EN 302 663 V1.2.1 , 2013.

[12] ETSI. Intelligent Transport Systems (ITS); Users and applications requirements; Part 1: Facility layer structure, functional requirements and specifications). Standard ETSI TS 102 894-1 V1.1.1 , 2013.

[13] ETSI. Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM). Standard ETSI EN 302 895 V1.1.1 , 2014.

[14] ETSI. 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, 2019.

[15] ETSI. Etsi experts complete specifications for vulnerable road users, nov 2020.

[16] ETSI. Intelligent Transport Systems (ITS); Vulnerable Road Users (VRU) awareness; Basic Set of Applications; Part 2: Functional Architecture and Requirements definition; Release 2. Standard ETSI TS 103 300-2 V2.1.1 , 2020.

[17] ETSI. Intelligent Transport Systems (ITS); Vulnerable Road Users (VRU) awareness; Basic Set of Applications; Part 2: Specification of VRU Awareness Basic Service. Standard ETSI TS 103 300-3 V2.1.1 , 2020.

[18] ETSI. Intelligent Transport System (ITS); Vehicular Communications; Basic Set of Applications; Collective Perception Service;. Standard ETSI TS 103 324 V2.1.1 , 2023.

[19] European Commission. Eu road fatalities drop 3% in 2024, but progress remains slow, mar 2024.

[20] European Commission, CORDIS. Holistic and adaptivE Interface Design for human-technology Interactions (HEIDI). https://cordis.europa.eu/project/id/101069538, n.d.

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

[22] Maria Gkemou, Francesco Biral, Ioannis Gkragkopoulos, Giammarco Valenti, Ioannis Tsetsinas, Evangelos Bekiaris, and Andrea Steccanella. Cooperation between roads and vehicles: Field validation of a novel infrastructure-based solution for all road users' safety. *Safety*, 7(2), 2021.

[23] Hamdan Hejazi and László Bokor. Modeling and evaluation of cooperative vulnerable road user protection schemes in realistic c-its environments. *Computer Networks*, 246:110396, 2024.

[24] Nehad Hameed Hussein, Chong Tak Yaw, Siaw Paw Koh, Sieh Kiong Tiong, and Kok Hen Chong. A comprehensive survey on vehicular networking: Communications, applications, challenges, and upcoming research directions. *IEEE Access*, 10:86127–86180, 2022.

[25] ITU-T. Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation. ITU-T X.680 (02/2021), 2021.

[26] ITU-T. Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER). ITU-T X.691 (02/2021), 2021.

[27] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[28] Rahima Khanam and Muhammad Hussain. What is yolov5: A deep look into the internal features of the popular object detector, 2024.

[29] Julian Kunkel, Jonathan Decker, and Stefanie Mülhausen. Decice: Device-edge-cloud intelligent collaboration framework, 2023.

[30] UCLA Mobility Lab. Opencda yaml file definitions, 2021.

[31] Roger Labbe. Kalman and bayesian filters in python, 2020.

[32] Xiaofeng Liu and Arunita Jaekel. Congestion Control in V2V Safety Communication: Problem, Analysis, Approaches. *Electronics*, 8(5):540, 2019.

[33] Silas Lobo, Leonardo Barbosa Da Silva, and Christian Facchi. To cluster or not to cluster: A vru clustering based on v2x communication. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2218–2225, 2023.

[34] Sohaib Malik, Muhammad Junaid Khan, Muhammad Ali Khan, and Hisham El-Sayed. Collaborative perception–the missing piece in realizing fully autonomous driving. *Sensors*, 23(18):7854, sep 2023.

[35] Gaurang Naik, Biplav Choudhury, and Jung-Min (Jerry) Park. Ieee 802.11bd & 5g nr v2x: Evolution of radio access technologies for v2x communications. *arXiv preprint arXiv:1903.08391*, 2019.

[36] University of Wolverhampton. Requirements and soteria architecture, nov 2023.

[37] University of Wolverhampton. Vrus and micro-mobility safety services v1, may 2024.

[38] Inc. Ouster. Os1 mid-range high-resolution imaging lidar datasheet, rev. 2021. Datasheet, Ouster Inc., 2021. Available: https://data.ouster.io/downloads/datasheets/datasheet-revd-v2p0-os1.pdf, Accessed: Sep. 22, 2025.

[39] Jelili Oyelade, Itunuoluwa Isewon, Olufunke Oladipupo, Onyeka Emebo, Zacchaeus Omogbadegun, Olufemi Aromolaran, Efosa Uwoghiren, Damilare Olaniyan, and Obembe Olawole. Data clustering: Algorithms and its applications. In *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*, pages 71–81, 2019.

[40] Roberto Pegurri, Diego Gasco, Francesco Linsalata, Marco Rapelli, Eugenio Moro, Francesco Raviglione, and Claudio Casetti. Van3twin: the multi-technology v2x digital twin with ray-tracing in the loop, 2025.

[41] F. Raviglione, C. M. Risma Carletti, M. Malinverno, C. Casetti, and C. F. Chiasserini. ms-van3t: An integrated multi-stack framework for virtual validation of V2X communication and services. *Computer Communications*, 217:70–86, 2024.

[42] Raphael Riebl, Hendrik-Jörn Günther, Christian Facchi, and Lars Wolf. Artery: Extending veins for vanet applications. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 450–456, 2015.

[43] Carlos Mateo Risma Carletti, Claudio Casetti, Jérôme Härri, and Fulvio Risso. Ms-van3t-carla: An open-source co-simulation framework for cooperative perception evaluation. In *2024 19th Wireless On-Demand Network Systems and Services Conference (WONS)*, pages 93–96, 2024.

[44] Pedro Rito, Ana Almeida, Andreia Figueiredo, Christian Gomes, Pedro Teixeira, Rodrigo Rosmaninho, Rui Lopes, Duarte Dias, Gonçalo Vítor, Gonçalo Perna,

Miguel Silva, Carlos Senna, Duarte Raposo, Miguel Luís, Susana Sargento, Arnaldo Oliveira, and Nuno Carvalho. Aveiro tech city living lab: A communication, sensing and computing platform for city environments, 07 2022.

[45] SAE International. Taxonomy and Definitions for Terms Related to Cooperative Driving Automation for On-Road Motor Vehicles. Technical Report J3216, SAE International, 2020.

[46] SAE International. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Technical Report J3016, SAE International, 2021.

[47] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011.

[48] Staff writers. Motorcyclist safety: Connected motorcycle pilot. https://imoveaustralia.com/project/motorcyclist-safety-connected-motorcycle-pilot/, mar 2023.

[49] The OpenCDA Team. Logic flow of opencda, 2025.

[50] Marc Torrent-Moreno, Jens Mittag, Paolo Santi, and Hannes Hartenstein. Vehicle-to-vehicle communication: Fair transmit power control for safety-critical information. *IEEE Transactions on Vehicular Technology*, 58(7):3684–3703, 2009.

[51] Felipe Valle, Daniel Bleckert, Linus Frisk, Oscar Amador Molina, Elena Haller, and Alexey Vinel. Non-negotiated implicit etsi vam clustering, 2025.

[52] WHO. 2nd UN Global Road Safety Week, 2013.

[53] Edmir Xhoxhi, Vincent Albert Wolff, Yao Li, and Florian Alexander Schiegg. Vulnerable road user clustering for collective perception messages: Efficient representation through geometric shapes, 2024.

[54] Runsheng Xu, Hao Xiang, Xu Han, Xin Xia, Zonglin Meng, Chia-Ju Chen, and Jiaqi Ma. The opencda open-source ecosystem for cooperative driving automation research, 2023.

[55] Zheng Xu, Xiaomeng Wang, Xuesong Wang, and Nan Zheng. Safety validation for connected autonomous vehicles using large-scale testing tracks in high-fidelity simulation environment. *Accident Analysis & Prevention*, 215:108011, 2025.

[56] Jeonghyeon Yun, Cheolsoon Lim, and Byungwoon Park. Inherent limitations of smartphone gnss positioning and effective methods to increase the accuracy utilizing dual-frequency measurements. *Sensors*, 22(24), 2022.