2024 IEEE Intelligent Vehicles Symposium (IV)
June 2-5, 2024. Jeju Island, Korea

Co-simulate no more: The CARLA V2X Sensor

Daniel Grimm*, Marc Schindewolf*, David Kraus†, and Eric Sax*
Institut fuer Technik der Informationsverarbeitung, Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: *{name.surname}@kit.edu, †d.kraus@kit.edu

2024 IEEE Intelligent Vehicle Symposium (IV) | 979-8-3503-4881-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/IV55156.2024.10588488

***Abstract*— Due to the increasing automation and connectivity of future, software-defined vehicles, there is a growing interest in safety in this area. Vehicle-to-Everything (V2X) communication is a crucial aspect of this. However, testing V2X applications in the real environment poses a number of challenges, such as the cost of renting test tracks and integrating V2X technology. To develop and test V2X applications more cheaply and at early development stages, simulation is a key enabler. There are several open-source tools that can simulate V2X communication, but a comprehensive and easy-to-use solution that enables to simulate V2X communication combined with sensor data required for automated driving is missing. Therefore, this paper presents an open-source approach that extends the CARLA simulation platform with a new module to create a unified environment for developing V2X applications in simulation. Different parameters can be set individually to account for real-world sensor's differences. Mechanisms for generating, transmitting and receiving Cooperative Awareness Messages (CAM) are implemented. For the transmission, different propagation models are included, taking into account the outlines of vehicles and buildings: Line of sight (LOS), non-line of sight due to vehicles (NLOSv) and non-line of sight due to static objects (NLOSb). We perform a benchmark measurement with a varying number of simulated vehicles, indicating that simulating the V2X communication introduces only a small overhead to the simulated sensors, such as cameras. The code for the V2X additions to CARLA is available online.**

***Index Terms*— Automotive, Autonomous vehicles, V2X, Vehicular communication, Vehicle-to-Everything, CARLA, Simulation**

I. Introduction

The two major trends that are transforming the automotive industry are automated driving and connectivity to the environment and the Internet, both of these are software functions [1]. The software-defined vehicle is based on data-driven development, especially for automated driving functions. Along with this, the seamless exchange of information between vehicles and their environment, known as vehicle-to-everything (V2X) communication, can have a major impact on improvements in road safety, traffic efficiency and overall sustainability in transportation [2], [3], [4]. New communication standards and technologies constantly improve the capabilities of V2X communication, e.g. 5G-based V2X has shown lower latency than 4G V2X in real-world measurements [5].

As the demand for innovative V2X applications grows, there is an increasing need for efficient and cost-effective means of exploring the intricacies of V2X communications [2]. The limitations of real-world testing, including the unpredictability of traffic scenarios, the cost of conducting large-scale trials and the potential risks involved, emphasize the importance of a controlled and scalable environment that a simulation platform can provide. Simulated testing has advantages in the rapid prototyping and iterative testing enabled by this virtual test track.

For the development of automated driving functions, several commercial simulation tools are available [6], such as IPG CarMaker[1], Vector Informatik DYNA4[2], and Virtual Test Drive[3]. A variety of scenarios can be tested and reproduced in such virtual environments, ranging from rare and dangerous events to extreme weather conditions and uncommon traffic scenarios. For V2X simulation, in comparison, there is a corpus of open-source simulation tools, such as Veins [7]. From commercial side, Vector Informatik's CANoe comes with a V2X option[4]. The open-source V2X simulation approaches, however, focus more on protocol and technology evaluations, e.g. comparing 5G and ITS-G5 performance [8], not on early application prototyping, and are not integrated well with automated driving simulations.

While evaluating the V2X technology stack is an invaluable contribution, simulations should also enable to test connected automated driving functions based on a versatile sensor setup (e.g. cameras and lidar) and V2X communication. However, current works in this field focus on co-simulating several tools. Learning to handle a new tool always requires some time - who *wants* to struggle with more than one tool, that need to be connected and synchronized? Veins, for example, ships as a virtual machine to account for setup hurdles. In our opinion, it is more user-friendly and reduces the barrier to entry into a field of research if new algorithms can be tested within just one tool with an easy-to-use programming interface. Thus, this paper focuses on the introduction of an integrated open-source simulation framework as a solution to overcome the limitations of current co-simulation-based tools, such as [9], [10].

Problem: Currently, open-source V2X simulation requires co-simulation of several tools to fully cover all aspects of cooperative automation, from V2X messages to camera and

[1] https://ipg-automotive.com/en/products-solutions/software/carmaker/
[2] https://www.vector.com/int/en/products/products-a-z/software/dyna4/
[3] https://hexagon.com/de/products/virtual-test-drive
[4] https://www.vector.com/int/en/products/products-a-z/software/canoe/option-car2x/

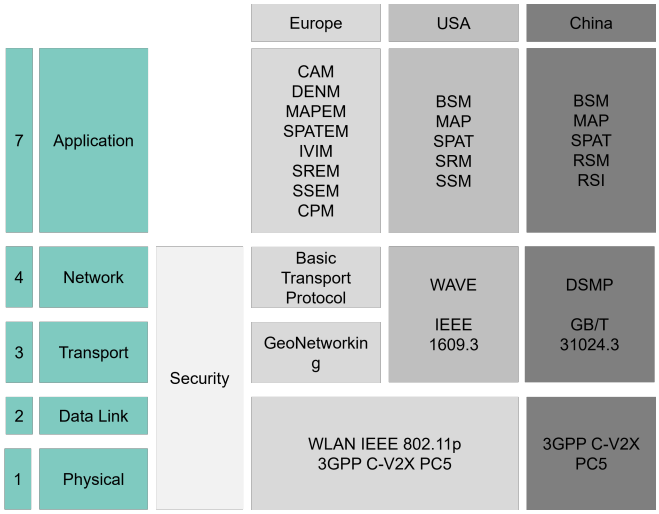

Fig. 1: Schematic representation of the protocol stacks in the various regions[5]

lidar sensors. While providing great solutions in their fields, combining tools is usually cumbersome and poses an entry barrier. Matching versions of different simulation tools from automated driving and wireless communication need to be set up together with an interfacing technology.

Aim: While perfectly suited to explore different V2X network, transport and access layers, coupling established wireless simulators with automated driving simulators is not putting the focus on enabling *application* research. This is our goal - *one* easy-to-use simulation tool to enable more research in cooperative automation.

Contribution: We extended the core CARLA server and Python API code base with a new *V2X Sensor*. This enables users to simulate V2V and V2I communication with Cooperative Awareness Messages (CAM) by following the common Python API procedures of CARLA sensors. The implemented path loss model considers both LOS and NLOS scenarios.

II. Background

A. Vehicle-to-Everything (V2X)

V2X communication is a technology that facilitates communication between vehicles and various elements of the transportation ecosystem. The ”X” in V2X encompasses different types of communication, including vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-pedestrian (V2P), and vehicle-to-network (V2N). This interconnected communication system is fundamental to the advancement of intelligent transportation and plays a critical role in the development of autonomous and connected vehicles.

Due to varying regulations worldwide, there are multiple standards for handling V2X. One of these is the European standard defined by the European Telecommunications Standards Institute (ETSI), that supports the CAM (cf. Fig. 1). The main purpose of a CAM is to create and maintain the awareness of vehicles on the road network to encourage cooperative behavior. CAM includes status and attributes details that are specific to the originating Intelligent Transportation System (ITS) station. The specific content varies depending on the type of ITS involved. In the case of a vehicle, the status information includes elements such as time, position, movement status and activated systems. The attribute information includes details about dimensions, vehicle type and role in road traffic [11].

A CAM consists of a standard ITS PDU header and multiple containers. The ITS PDU header contains the protocol version, the message type and the originating ITS station ID. For vehicles, a CAM typically contains a basic container with basic information, a high-frequency for dynamic data, and optionally a low-frequency container for static details. There may also be a special container with information specific to the role of the vehicle. CAMs generated by a Road Side Unit (RSU) must include a basic container and may include additional containers. The CAM generation frequency is managed by the CA basic service. The trigger conditions for a CAM are specified in Table I. [11]

Besides the mentioned CAM Message, there are several other important messages for V2X communication, cf. Fig. 1-Application.

TABLE I: Trigger conditions for CAM message

Component	Trigger conditions
Vehicle	Heading angle change $> 4°$ Position difference > 4m Speed change > 5m/s Time elapsed $>$ CAM Generation time Low Frequency Container Time Elapsed > 500ms
RSU	Time elapsed > 500ms

B. Channel models

Channel or propagation models play a crucial role in the design and assessment of wireless data transmission, aiming to replicate the distortion that signals experience while travelling between transmitters (TX) and receivers (RX). Wireless channels exhibit various effects, including attenuation, reflection, transmission, diffraction, scattering, and wave guiding. For instance, signal strength diminishes with increasing distance between TX and RX due to attenuation, while wave guiding, found in settings like urban canyons and tunnels, preserves signal strength by constraining its expansion. In the following, we summarize common models that are important for our work. In general, we consider three propagation scenarios: Line-of-Sight (LOS), Non-LOS due to vehicles in the propagation path (NLOSv), Non-LOS due to buildings or foliage in the propagation path (NLOSb).

1) Free space path loss: The Free Space Path Loss (FSPL) model is a fundamental concept in wireless communication that describes the attenuation of radio frequency

[5]based on `https://www.vector.com/de/de/know-how/v2x/`

signals as they propagate through free space. It is a simplistic but essential model to estimate the signal strength at different distance from a transmitting antenna. The FSPL (in dB) can be calculated using

$$\mathrm{FSPL}(d)/\mathrm{dB} = 20\log_{10}(d) + 20\log_{10}\left(\frac{4\pi}{\lambda}\right) \quad (1)$$

where d is the distance between the transmitter and receiver in meters, λ is the wavelength of the signal in meters (m).

2) Log-distance path loss: Extending FSPL to a more general case, where environmental conditions introduce additional damping and fading, the Log-distance path loss (LDPL) was derived.

$$\mathrm{LDPL}(d)/\mathrm{dB} = \mathrm{FSPL}(d_{\mathrm{ref}}) + 10\gamma\log_{10}\left(\frac{d}{d_{\mathrm{ref}}}\right) + X \quad (2)$$

where γ is an environment-dependent path loss exponent, d_{ref} is a reference distance, and X being a random (normal [12] or log-normal [13]) distributed variable that models fading.

3) Two-ray ground reflection: The Two-ray Ground Reflection loss (TRGL) model is a radio wave propagation model that considers the effects of reflections from the ground. In addition to the direct path between the transmitter and the receiver, the signal also travels indirectly by reflecting off the ground. This reflection introduces a delay and amplitude variation to the received signal. The different phase leads to constructive and destructive interference depending on the distance. The TRGL calculates to

$$\mathrm{TRGL}(d)/\mathrm{dB} = 20\log_{10}\left(\frac{4\pi d}{\lambda}|1+\Gamma e^{i\phi}|^{-1}\right) \quad (3)$$

with

$$\Gamma = \frac{\sin\theta - \sqrt{\epsilon_r - \cos^2\theta}}{\sin\theta + \sqrt{\epsilon_r - \cos^2\theta}} \quad (4)$$

$$\sin\theta = \frac{h_{\mathrm{t}} + h_{\mathrm{r}}}{d_{\mathrm{refl}}}, \quad \cos\theta = \frac{d}{d_{\mathrm{refl}}} \quad (5)$$

for the incidence angle θ and the relative permittivity ϵ_r of the road. The phase difference ϕ between the rays is

$$\phi = \frac{2\pi(d_{\mathrm{los}} - d_{\mathrm{refl}})}{\lambda} \quad (6)$$

As shown [14], using the full TRGL calculation instead of an approximation yields a more accurate model of free space propagation for V2V links.

4) Multiple Knife-edge diffraction: When the LOS is blocked, the wireless signal may reach its destination by bending around corners (diffraction). Especially in the NLOSv cases, diffraction of the wireless signal over the tops and between vehicles is of importance. A usual way to model this is treating each vehicle blocking the LOS as a knife-edge obstacle. Several obstacles may be combined by finding major and minor obstacles and calculating individual knife-edge loss for each, as e.g. implemented in the model of Boban et al. [12], [15] used in the Artery[6] simulator and recommended by ETSI [16]. Each vehicle that is considered as an obstacle contributes

$$L_{\mathrm{v}}/\mathrm{dB} = 6.9 + 20\log_{10}(\sqrt{(v-0.1)^2+1} + v - 0.1) \quad (7)$$

if $v > -0.78$, and 0 otherwise. v is approximated for small height differences between the sender, receiver and the obstacle with

$$v = \sqrt{\frac{2d}{\lambda}\cdot\frac{h}{d_{\mathrm{tx,v}}}\cdot\frac{h}{d_{\mathrm{v,rx}}}} \quad (8)$$

and h being the height of the obstacle vehicle, and $d_{\mathrm{tx,v}}$ and $d_{\mathrm{v,rx}}$ being the distances between sender and obstacle vehicle, and vehicle and receiver, respectively.

C. CARLA simulator

The CARLA Simulator [17] mainly consists of two parts: The simulator and the CARLA client, usually referred to as the server and the client for short (cf. Fig. 2). The CARLA server itself is a plugin of the Unreal Engine. The server and client communicate via a TCP connection so that sensor data can be received, and control commands sent via Python scripts that import the Python API. Several parts are required for sensor data to reach the Python script from the Unreal Engine. First, a sensor is an Unreal Engine Actor, as part of the CARLA plugin. In addition, an object for the representation of the sensor data and a serializer / deserializer for each sensor are defined in LibCarla. The sensor data object is passed to the C++ or Python API. To extend CARLA with new sensors, these three parts must therefore be developed.

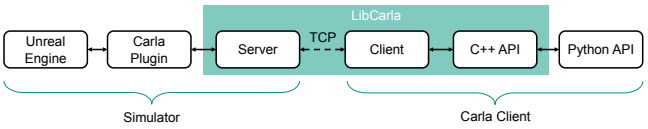

Fig. 2: Communication and data flow in CARLA[7]

III. Related Work

Established tools for combined simulation of vehicle mobility and communication are Artery [18] and Veins [7], for example. However, they focus on exploring the vehicular networking protocols, not on their integration with 3D environment simulations, which are the core of automated driving application research. One of the projects considering a broader scope is VSimRTI [19], which is the basis of Eclipse MOSAIC [10]. However, while MOSAIC incorporates several communication tools (e.g. ns-3, OMNeT++), vehicles are only simulated with Eclipse SUMO [20]. Accordingly, MOSAIC's functionality doesn't exceed Artery or Veins substantially. Choudhury et al. integrated VISSIM, MATLAB, and NS-3 for traffic modelling, traffic management applications, and communication network simulation [21]. However,

[6]http://artery.v2x-research.eu/

[7]according to `https://carla.readthedocs.io/en/0.9.15/img/pipeline.png`

the use of commercial software and the co-simulation of multiple tools pose challenges. In addition, sensor simulation for automated driving and advanced driver assistance systems (ADAS) is limited. Naumann et al. introduced an open-source simulation framework for traffic participants interactions using Robot Operating System (ROS) [22]. Focusing on interaction modeling and cooperative motion planning, perception was not considered. In addition, the simulation of the physical V2X communication is limited to specifying message drop or delays. Zhang and Masoud utilize the Gazebo Engine as a simulator for V2X, addressing realistic environment simulation, sensor simulation, and vehicle dynamics [23]. However, in comparison with simulators targeting automated driving such as CARLA and (LG)SVL, Gazebos virtual worlds are limited.

Besides, there are several works co-simulating CARLA with real or simulated V2X setups. Hardes et al. [24] combine OMNeT++ / Veins with CARLA using gRPC[8] and Protocol Buffers[9] as the coupling interface. Qiu et al. [25] also employ CARLA for cooperative driving, however, the implementation of path loss remains unclear. Cislaghi et al. [26] use ZeroMQ[10] to couple CARLA with OMNeT++. The approach was evaluated for a teleoperation use case over a 5G network. Another approach was presented by Zhao et al. [9]. They use SUMO, OMNeT++, and Unity3D to model the traffic, V2X communication and visualization. The tools are coupled with a specific communication protocol.

IV. The CARLA V2X Sensor

In the following, we present our concept for extending the state-of-the-art automated driving simulator *CARLA* [17] with a new *V2X Sensor*. CARLA was chosen as the starting point for our development, because it is an established open-source simulator for research in automated driving, based on the Unreal Engine. Aiming for an ADAS-focused simulation environment, the major requirements for our work were

- an open-source realization,
- a realistic 3D environment,
- vehicle dynamics simulation,
- ADAS sensor simulation,
- configurable channel model for wireless propagation,
- support for both standard messages and custom data,
- a simple API for control and configuration of the simulation.

CARLA covers a large amount of requirements, such as vehicle dynamics, ADAS sensor simulation, and comes with a ROS2 interface[11] and a Python API. While the ROS interface is helpful to connect to vehicle-internal networks and simulated in-vehicle applications, e.g. [27], external communication is not yet part of CARLA's simulation capabilities. In comparison to current research, focusing on co-simulation of several simulators, we make use of the inherent open-source nature of CARLA, and contribute a new feature to the open-source codebase, whose details are outlined in the following. Currently, a pull request to the main branch of CARLA with our contribution is open, the new *V2X Sensor*[12].

A. Overview

In our enhancement, a new sensor is introduced to CARLA that handles tasks in the facilities and access layer (cf. Fig.3). This sensor serves two roles: Managing the initiation and generation of messages according to the facilities layer's needs and simulation of propagation modes (LOS, NLOS) in the access layer. The implemented path loss model is configurable to the user's needs, e.g. to model real-world behavior. Specifically, our simulation focuses on the simulation of communication by CAM with its trigger conditions, which are provided for in the ETSI standard [11] (cf. TableI). Besides, we implement a custom message, that allows to send arbitrary strings at user-defined points in time utilizing the common path loss models. This message may be used to explore new cooperative applications, where custom data needs to be transmitted, that is currently not covered in standards.

In comparison to other simulators, we haven't implemented the network and transport layer yet, considering only broadcast communication. What distinguishes our approach from co-simulation based approaches, is that V2X applications can be written in Python without further configuration of the simulator. Users instantiate vehicles and parametrize and attach sensors to the vehicles via the Python API. This enables, for example, to simulate the interaction of camera- and V2X-equipped vehicles in combination with non-connected vehicles with a few lines of code.

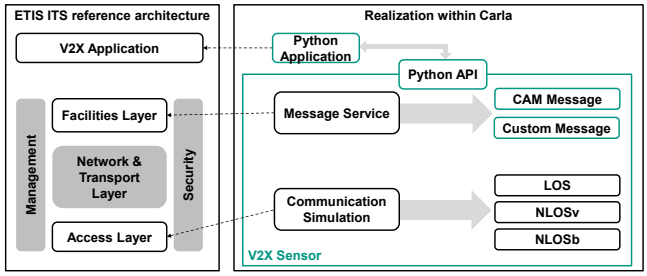

Fig. 3: V2X sensor representation in CARLA as per ITS Reference architecture

B. Propagation model

In the context of propagation simulation, the objective is to address the disparity between simplified statistical models and computationally more expensive geometry-based models. The presence of nearby buildings, foliage, and vehicles is taken into account to assess the line-of-sight conditions for each link, which has a substantial impact

[8] https://grpc.io/
[9] https://protobuf.dev/
[10] https://zeromq.org
[11] https://github.com/carla-simulator/ros-bridge
[12] https://github.com/carla-simulator/carla/pull/7490

on V2X communication [12], [28]. Besides LOS, NLOSb and NLOSv models, we incorporate a simple fading model for small-scale variations. In general, we follow Artery's implementation for the LOS, NLOsb, NLOSv and fading model, but enable the user to choose the Winner+ model [29] as an alternative. In sum, the propagation loss calculates to

$$L_{\text{total}} = L_{\text{PL}} + L_{\text{AG}} + L_{\text{SF}} \tag{9}$$

with L_{PL} being the path loss from the geometrical or the Winner+ model, L_{AG} being a constant, configurable offset due to the antenna gains of sender and receiver (default: $10\,\text{dB}$), and L_{SF} the small-scale shadow fading loss.

1) Propagation class calculation: As outlined, we employ propagation models for three different classes of propagation. Accordingly, we need to find out which model to use for a given link. The method used for this purpose is the `LineTraceMultiByObjectType` API provided by Unreal Engine[13]. This API traces a ray against the world, using specified object types, and returns information about hits between the specified source and destination points. Important to note are the used query parameters, where the `ECollisionChannel` enumeration defines the object classes that can be queried[14]. We use *ECC_WorldStatic*, *ECC_PhysicsBody*, *ECC_Vehicle* and *ECC_WorldDynamics*.

2) Geometry-based model: The geometry-based model uses three different calculations depending on the line-of-sight condition (LOS, NLOSv, NLOSb). For LOS links, the implementation involves the full two-ray ground reflection model (cf. Sec. II). The consideration of the height of the antennas is crucial in this context, as small differences in height between Tx or Rx results in significantly different interference relationships between the LOS and ground-reflected ray [14]. We also consider the height differences of the roads when calculating the two-ray model.

Modelling NLOSv links bases on the free-space path loss model, and incorporates the blockage loss per vehicle. A model for vehicles-as-obstacles is described in [12], where vehicles are modelled using the (multiple) knife-edge diffraction. This model calculates additional attenuation due to each of the vehicles blocking the LOS link. For computational reasons, we only consider the maximum knife-edge diffraction loss of the tops of all blocking vehicles instead of combining these losses.

For NLOSb links, the path loss is calculated using the Log-distance path loss model (cf. Sec. II). This model with appropriate path loss exponent and shadowing deviation was experimentally shown to model the path loss for V2V links in NLOSb scenarios [28]. Following [28], [16], we use $d_{\text{ref}} = 1\,\text{m}$, $PL(d_{\text{ref}}) = 47.86\,\text{dB}$, $\gamma = 2.7$ as default parameters for the model, where $PL(d_{\text{ref}})$ is the path loss at the reference distance ref and γ is the path loss exponent. However, the parameters can be configured by the user.

[13]`https://docs.unrealengine.com/4.26/en-US/API/Runtime/Engine/Engine/UWorld/LineTraceMultiByObjectType/`

[14]`https://docs.unrealengine.com/4.26/en-US/API/Runtime/Engine/Engine/ECollisionChannel/`

TABLE II: Small-scale fading standard deviations in (dB)

Link type	Highway	Rural	Urban
LOS	3.3	4.25	5.2
NLOSv	3.8	4.55	5.3
NLOSb	6.8	6.8	6.8

3) Winner+ model: For completeness, the Winner+ model [29] is specified as:

$$L/\text{dB} = \begin{cases} 32.4 + 20.0\log_{10}(d) + 20.0\log_{10}(f_c) & 1) \\ 38.77 + 16.7\log_{10}(d) + 18.2\log_{10}(f_c) & 2) \\ 36.85 + 30.0\log_{10}(d) + 18.9\log_{10}(f_c) & 3) \end{cases}$$

with frequency f_c in GHz, distance d in meters, and case 1) highway and NLOSv or LOS, 2) urban and NLOSv or LOS, and 3) NLOSb.

4) Shadow fading: A shadow fading loss is added to all three propagation classes by default, to account for small scale variations, as used in [15]. We employ a simple Gaussian model, with a standard deviation σ in dB depending on the propagation class (LOS, NLOSb, NLOSv) and a scenario parameter (urban, rural, highway) that is selected by the user. Adapted from [16], the parameters for the different scenarios are shown in Table II. Because [16] does not mention parameters for the rural scenario, we assume the average of highway and urban scenario standard deviation values. In addition, we take the NLOSb value for the urban scenario for the highway and rural scenario as well. Alternatively, the user may configure a custom standard deviation for the Gaussian model, that is used independently of the propagation class or scenario.

C. CARLA integration

To incorporate a new sensor into CARLA, the steps outlined in the official documentation[15] are applied. Accordingly, this section is intentionally more technical to facilitate third-party developers to extend our work with new V2X messaging services. Figure 4 depicts the inheritance relationship where the ASensor class is derived from the AActor class. ASensor introduces virtual functions such as `Set`, `PrePhysTick`, and `PostPhysTick`. These need to be implemented in the V2XSensor classes `AV2XSensor` and `ACustomV2XSensor`. The `AV2XSensor` class implements the CAM-generating sensor, while `ACustomV2XSensor` implements the custom V2X message. We separate the message generation functionalities, but both classes integrate the same path loss model class. Following the Unreal Engine naming convention, because the V2X sensors inherit from AActor, the V2X sensor classes are also prefixed by "A".

The static variables *V2XActorContainer* and *ActorV2XMessageMap* serve to build a shared view of all `AV2XSensors` (respectively `ACustomV2XSensors`),

[15]`https://carla.readthedocs.io/en/0.9.15/tuto_D_create_sensor/`

and the triggered messages in the current simulation step. The *CaService* object generates and triggers CAMs, while the *PathLossModel* object provides the different path loss models and the respective parameters.

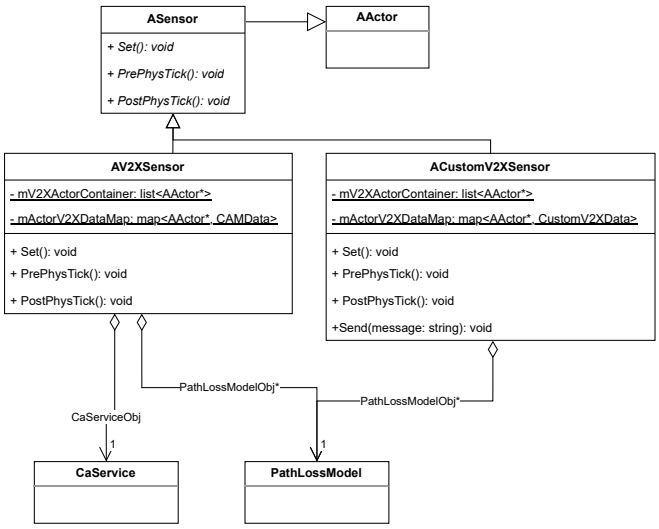

Fig. 4: Class diagram of the integration of the V2X sensor

During each simulation step, CARLA first executes `PrePhysTick()` for all objects (e.g., sensors), then `PostPhysTick()` is invoked. Accordingly, our approach *generates* messages for all V2XSensor objects during the `PrePhysTick()` phase (according to the triggering conditions), and *transmits* them during the `PostPhysTick()` phase, using the path loss model. For the custom V2X sensor, the user sets the message to be sent via the `send` function using the Python API. In the next simulation step, this message is broadcasted once to all other vehicles that are equipped with the custom V2X sensor.

D. Configuration possibilities

We expose several parameters of the *V2X Sensor* (cf. Table III) to the Python API to allow for customization and the evaluation of multiple sensors or scenarios. The transmit power and receiver sensitivity default values are derived from Cohda Wireless MK6 OBU [30], serving as reference points for real-world V2X sensors. A seed value can be used to ensure reproducible randomness. The filter distance is used to speed up the calculations on large maps, calculating the path loss model only for potential recipients within this distance. The transmission frequency is a parameter that is shared among the V2X sensor instances to ensure that all sensors are always configured with the same frequency.

As the *V2XSensor* and the *CustomV2XSensor* use the same path loss model, they can both be configured with the same set of parameters in this respect. It is possible to set the scenario (urban, rural, highway), and the used model (Winner+ or geometric model). As mentioned in section IV-B.4, the fading noise is dependent on the scenario and path loss class (LOS, NLOSv, NLOSb), or can be customized with the parameters `use_etsi_fading=false` and the custom fading standard deviation value. Finally, for the LDPL model, the path loss exponent and the reference distance (d_{ref}) can be set.

TABLE III: Default settings for configurable parameters

Parameter	Default Value (Options)
General parameters	
transmit_power	21.5 dBm
receiver_sensitivity	−99 dBm
frequency_ghz	5.9 GHz
noise_seed	0
filter_distance	500 m
Path loss model parameters	
combined_antenna_gain	10.0 dB
scenario	urban [rural, highway]
path_loss_model	geometric [winner]
d_ref	1.0 m
path_loss_exponent	2.7
use_etsi_fading	true [false]
custom_fading_stddev	0.0 dB
CAM-related parameters	
gen_cam_min	0.1 s
gen_cam_max	1.0 s
fixed_rate	false [true]

The CAM message generation can be configured with minimum and maximum delta times. In addition, a debug option `fixed_rate` is available that generates a CAM message in every simulation step. The data, that is sent in CAM messages, can also be noisy with Gaussian distribution. Here, the user can specify standard deviation and bias of the latitude, longitude, altitude, heading, and yaw rate. For the accelerations in all three axes and for the forward velocity, the user can set a standard deviation for a Gaussian noise.

V. Evaluation

To measure the performance of the simulation, we use a modified version of the performance benchmark script[16], which is included in the CARLA source code. We evaluated the performance on a desktop PC with a CPU Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz, 46.99 GB RAM and an NVIDIA GeForce RTX 3060 Ti GPU. The camera and lidar sensors send data with every tick, so the time between two data receipts is used to calculate the frame rate. Since the V2X sensor does not send a message to the client with every tick, it is not possible to calculate the frame rate in the same way when using the V2X sensor. This only can be realized if the parameter `fixed_rate` is set, as a message is then generated in every frame. Therefore, we measure different settings with the modified script:

1) Without V2X: Baseline performance on used hardware
2) With V2X sensor, CAM generation according to ETSI
3) With V2X sensor, CAM generation after each tick

In each of the settings 1)-3), two sensor setups are evaluated: one camera (1920x1080 pixels) or one lidar (500k points) per

[16] `https://carla.readthedocs.io/en/0.9.15/adv_benchmarking/`

vehicle. Town 01 was used, and the frame rate is averaged over the three weather settings ClearNoon, CloudyNoon and SoftRainSunset. The simulation was run for 1,200 ticks = 1 minute simulation time (with a fixed step size of $0.05\,\mathrm{s}$). Accordingly, for setting 3, every $50\,\mathrm{ms}$ in simulation time, a message is generated. A frame rate above 20 frames per second (FPS) would mean that we can simulate faster than real-time. We varied the number of vehicles in the simulation from two to 50.

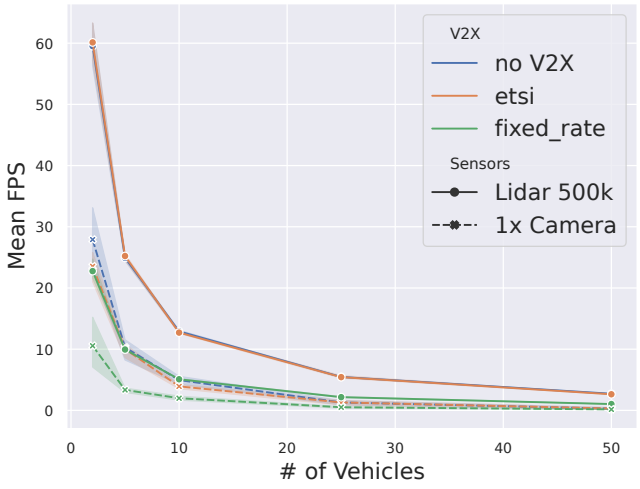

Fig. 5: Benchmark results per sensor setup and V2X setting

Fig. 5 shows the influence of the two sensor setups versus the influence of the V2X setting and the number of vehicles. Generally, simulating cameras instead of lidars introduces higher frame rate degradation than enabling V2X with ETSI triggering conditions. Triggering a CAM in every tick, on the other hand, would substantially decrease the performance.

TABLE IV: Mean lidar + camera FPS of the CARLA simulation with and without V2X

Setting	Mean FPS				
	2 Veh.	5 Veh.	10 Veh.	25 Veh.	50 Veh.
1) no V2X	43.72	17.61	8.93	3.38	1.50
2) ETSI	41.83	17.63	8.32	3.37	1.47
3) Fixed rate	16.66	6.65	3.54	1.34	0.60

The results (cf. Table IV) indicate, that the usage of the V2X Sensor introduces an overhead of 0.43% if CAM messages are generated according to the ETSI standard with two vehicles. If the V2X sensors generate a message in every tick, the performance degrades for two vehicles around 61.89% compared to not simulating V2X. The simulation speed generally degrades with the number of vehicles, which is expected due to the growing number of simulated sensors (cameras, lidars). However, the overhead of V2X stays below 0.7%, if triggered according to the ETSI standard, independent of the number of vehicles. This indicates that the influence of additional ADAS sensors on the simulation performance is much stronger than (standards-conforming) V2X simulation.

Given the qualitative evaluation in Table V, our approach combines V2X communication (CAM and custom messages) as well as sensors for ADAS and vehicle dynamics. To achieve similar capabilities with current approaches, it is necessary to combine several tools, such as within Veins or Artery (both require Sumo and OMNeT++), while missing the ability to simulate ADAS such as cameras. This would require a combination of the above tools with Carla (or another capable simulator).

TABLE V: Comparison of open-source tools to simulate V2X and ADAS

	# Req. tools	CAM	Custom Msg.	Access layer	Netw. and Transp. Layer	API	ADAS Sens.	Vehicle Dynamics
Veins [7]	2	✓	-	✓	✓	TraCI/ C++	-	(✓)
Artery [18]	2	✓	-	✓	✓	TraCI/ C++	-	(✓)
Carla [17]	1	-	-	-	-	Python	✓	✓
Carla+ Veins [24]	3	✓	-	✓	✓	combined (gRPC)	✓	✓
Ours	1	✓	✓	✓	-	Python	✓	✓

✓= fulfilled, (✓) = partially fulfilled, - = not fulfilled

VI. Conclusion

The simulation of sensor data, driving dynamics and wireless communication between vehicles and the environment is an important step in the development of automated vehicles. Especially in the early development phases, where rapid prototyping of algorithms is required, a simple interface to the simulation tool used is essential. Currently, co-simulations of several tools are mainly used in this environment, which represents a barrier to entry into this field. In our work, we present an alternative: the V2X sensor in CARLA. Without co-simulation, ADAS sensor data, vehicle dynamics and V2X communication can be simulated in a single tool. When simulating V2X communication in addition to ADAS sensors, such as camera and lidar, the simulation speed is largely unaffected, compared to simulating only camera or lidar. Our code is currently under review for integration into the main code base of CARLA. In the future, we would like to expose the V2X messages via the ROS interface of CARLA. In addition, the integration of latency models for the network and transport layer is planned to enable a more accurate simulation of V2X communication. Other V2X services such as Decentralized Environmental Notification Message (DENM) [31] and Collective Perception Message (CPM) [32] could also be implemented using the same development steps.

VII. Acknowledgment

The authors are grateful to Nishanth Martis for his great support in this project. The authors would like to thank the Ministry of Science, Research and Arts of the Federal State of Baden-Württemberg for the financial support of the projects within the InnovationCampus Future Mobility (ICM). This work has been funded by the German Federal Ministry of Education and Research (BMBF) in the project MANNHEIM-AutoDevSafeOps (Förderkennzeichen: 01IS22087J).

References

[1] O. Burkacky, J. Deichmann, and J. P. Stein, "Automotive software and electronics 2030: Mapping the sector's future landscape," 2019-07.

[2] J. Wang, Y. Shao, Y. Ge, and R. Yu, "A survey of vehicle to everything (v2x) testing," *Sensors*, vol. 19, no. 2, p. 334, Jan. 2019.

[3] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A tutorial on 5g nr v2x communications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1972–2026, 2021.

[4] M. Boehme, M. Stang, F. Muetsch, and E. Sax, "TalkyCars: A Distributed Software Platform for Cooperative Perception," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, Oct. 2020, pp. 701–707.

[5] D. Ficzere, G. Soós, P. Varga, and Z. Szalay, "Real-life v2x measurement results for 5g nsa performance on a high-speed motorway," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021, pp. 836–841.

[6] B. Schütt, M. Steimle, B. Kramer, D. Behnecke, and E. Sax, "A taxonomy for quality in simulation-based development and testing of automated driving systems," *IEEE Access*, vol. 10, pp. 18 631–18 644, 2022.

[7] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, January 2011.

[8] H. Hejazi and L. Bokor, "A survey on simulation efforts of 4g/lte-based cellular and hybrid v2x communications," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, 2021, pp. 333–339.

[9] X. Zhao, Y. Gao, S. Jin, Z. Xu, Z. Liu, W. Fan, and P. Liu, "Development of a cyber-physical-system perspective based simulation platform for optimizing connected automated vehicles dedicated lanes," *Expert Systems with Applications*, vol. 213, p. 118972, Mar. 2023.

[10] K. Schrab, M. Neubauer, R. Protzmann, I. Radusch, S. Manganiaris, P. Lytrivis, and A. J. Amditis, "Modeling an its management solution for mixed highway traffic with eclipse mosaic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 6, pp. 6575–6585, 2023.

[11] European Telecommunications Standards Institute (ETSI), "Intelligent transport systems (its); vehicular communications; basic set of applications; cooperative awareness service; release 2," *ETSI TS 103 900 V2.1.1*, 2023.

[12] M. Boban, T. T. V. Vinhoza, M. Ferreira, J. Barros, and O. K. Tonguz, "Impact of vehicles as obstacles in vehicular ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 15–28, 2011.

[13] European Telecommunications Standards Institute (ETSI), "5G; Study on channel model for frequencies from 0.5 to 100 GHz (3GPP TR 38.901 version 14.0.0 Release 14)," *ETSI TR 138 901 (V14.0.0)*, 2017.

[14] C. Sommer, S. Joerer, and F. Dressler, "On the applicability of two-ray path loss models for vehicular network simulation," in *2012 IEEE Vehicular Networking Conference (VNC)*, 2012, pp. 64–69.

[15] M. Boban, J. Barros, and O. K. Tonguz, "Geometry-based vehicle-to-vehicle channel modeling for large-scale simulation," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4146–4164, 2014.

[16] European Telecommunications Standards Institute (ETSI), "Intelligent transport systems (its); access layer; part 1: Channel models for the 5,9 ghz frequency band," *ETSI TR 103 257-1 V1.1.1*, 2019.

[17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[18] R. Riebl, C. Obermaier, and H.-J. Günther, "Artery: Large Scale Simulation Environment for ITS Applications," ser. EAI/Springer Innovations in Communication and Computing, A. Virdis and M. Kirsche, Eds. Cham: Springer International Publishing, 2019, pp. 365–406.

[19] T. Queck, B. Schünemann, I. Radusch, and C. Meinel, "Realistic simulation of v2x communication scenarios," in *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference*, ser. APSCC '08. USA: IEEE Computer Society, 2008, p. 1623–1627. [Online]. Available: https://doi.org/10.1109/APSCC.2008.23

[20] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, 2018.

[21] A. Choudhury, T. Maszczyk, C. B. Math, H. Li, and J. Dauwels, "An integrated simulation environment for testing v2x protocols and applications," *Procedia Computer Science*, vol. 80, pp. 2042–2052, 2016, international Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA.

[22] M. Naumann, F. Poggenhans, M. Lauer, and C. Stiller, "Coincar-sim: An open-source simulation framework for cooperatively interacting automobiles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[23] E. Zhang and N. Masoud, "V2xsim: A v2x simulator for connected and automated vehicle environment simulation," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[24] T. Hardes, I. Turcanu, and C. Sommer, "Poster: A case for heterogenous co-simulation of cooperative and autonomous driving," in *2023 IEEE Vehicular Networking Conference (VNC)*, 2023, pp. 151–152.

[25] H. Qiu, P. Huang, N. Asavisanu, X. Liu, K. Psounis, and R. Govindan, "Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving," in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '22, 2022.

[26] V. Cislaghi, C. Quadri, V. Mancuso, and M. A. Marsan, "Simulation of tele-operated driving over 5g using carla and omnet++," in *2023 IEEE Vehicular Networking Conference (VNC)*, 2023, pp. 81–88.

[27] D. Grimm, M. Schindewolf, and E. Sax, "Fleet in the loop: An open source approach for design and test of resilient vehicle architectures," in *2023 IEEE 15th International Symposium on Autonomous Decentralized System (ISADS)*, 2023, pp. 1–8.

[28] J. Karedal, F. Tufvesson, T. Abbas, O. Klemp, A. Paier, L. Bernadó, and A. F. Molisch, "Radio channel measurements at street intersections for vehicle-to-vehicle safety applications," in *2010 IEEE 71st Vehicular Technology Conference*, 2010, pp. 1–5.

[29] P. Heino, J. Meinilä, P. Kyösti, L. Hentila, T. Jämsä, E. Suikkanen, E. Kunnari, and M. Narandzic, "Cp5-026 winner+ d5.3 v1.0 winner+ final channel models," 01 2010.

[30] Cohda Wireless, "MK6 OBU product brief sheet." [Online]. Available: https://www.cohdawireless.com/wp-content/uploads/2023/06/CW_Product-Brief-sheet-MK6-OBU-6-23.pdf

[31] European Telecommunications Standards Institute (ETSI), "Intelligent transport systems (its); vehicular communications; basic set of applications; part 3: Specifications of decentralized environmental notification basic service," *ETSI EN 302 637-3*, 2023.

[32] ——, "Intelligent transport systems (its); vehicular communications; basic set of applications; cooperative awareness service; release 2," *ETSI TS 103 324*, 2023.