This is an author-created postprint version of the paper:
S. Lopez, M. Sepulcre, M. Fornell, D. Quiñones, J. Espinosa, J. Gozalvez, P. Moga, "ETSI standard-compliant Collective Perception Service for Connected Automated Driving", Proc. FISITA 2023 World Congress, Barcelona (Spain), 12-15 September 2023

ETSI standard-compliant Collective Perception Service for Connected Automated Driving

Santiago Lopez[1], Miguel Sepulcre[1], Miguel Fornell[2], David Quiñones[2*], Jose Espinosa[1], Javier Gozalvez[1], Pere Mogas[2]

[1]Universidad Miguel Hernandez de Elche (UMH), Avda. Universidad s/n, 03202, Elche, Alicante, Spain
E-mail: santiago.lopezc@umh.es, msepulcre@umh.es, jose.espinosac@umh.es, j.gozalvez@umh.es
[2]Idneo Technologies SLU, Carrer Rec de Dalt, 3, 08100, Mollet del Vallès, Barcelona, Spain
E-mail: miguel.fornell@idneo.com, david.quinones@idneo.com, pere.mogas@idneo.com

DOI (FISITA USE ONLY)

ABSTRACT: Cooperative (or collective) perception enables connected automated vehicles and traffic infrastructure to wirelessly exchange information about the objects they perceive with their onboard sensors. It is thus conceived to increase the perception capabilities of connected automated vehicles, improving traffic safety and efficiency. However, the implementation of cooperative perception is challenged by the need to exchange updated information, which requires minimizing the latency in the transmission and reception of collective perception messages. In this context, the main objective of this paper is the implementation, validation and testing of a standard-compliant Collective Perception Service (CPS) following the ETSI specifications recently published. The proposed implementation is validated under varying conditions and its scalability is analyzed in a laboratory environment.

KEY WORDS: connected and automated vehicles, collective perception, cooperative perception, V2X communications, CPS standard, ETSI specification, C-V2X communications, 5G New Radio, embedded hardware.

1. Introduction

Automated vehicles are equipped with onboard sensors to perceive their surrounding environment. However, their perception capabilities are often hindered by the presence of obstacles, such as other vehicles or buildings, and by adverse weather conditions. These limitations can significantly degrade the perception capabilities of automated vehicles, and hence negatively influence their safety and driving efficiency.

In order to improve their perception capabilities, connected automated vehicles (CAVs) could wirelessly exchange sensor data with nearby vehicles and infrastructure nodes. This concept is known as cooperative perception, collective perception, or cooperative sensing and relies on the use of V2X (Vehicle to Everything) communications. By receiving information from other vehicles about objects beyond their own sensing range using collective perception, CAVs enhance their perception of the surrounding environment. Collective perception can also improve the object detection accuracy and also enhance the confidence in identifying detected objects.

To enable the interoperability among different implementations, the European Telecommunications Standards Institute (ETSI) has recently published a Technical Specification to define the so-called Collective Perception Service (CPS) [1]. The CPS is part of the Facilities layer of the C-ITS architecture defined by ETSI. The CPS specification is one of the first standards of the so-called Release 2 set of ETSI specifications. In addition, CPS is a key Day 2 service for sensing driving in the C2C-CC (Car-to-Car Communication Consortium) roadmap [2]. The CPS specification defines the format of the Collective Perception Message (CPM) that is used to exchange the information, and the CPM generation rules. These rules determine when vehicles and infrastructure nodes should generate a new CPM and the information it should contain.

Collective perception has been studied to date mainly through simulation studies, that were conducted in parallel to the standardization process. These studies have been required to guide the design of the technical solution adopted in the ETSI specifications. For example, the performance of collective perception is studied in [3] with the focus on a decentralized data association and fusion process. A performance analysis is also conducted in [4] to quantify the safety improvement produced by collective perception. The work in [5] proposed and analyzed a first set of CPM generation rules that were later used as a basis in the standardization process. The work in [6] proposed the Look-Ahead mechanism to improve the efficiency of the CPM generation that was also adopted in the specification [1]. To mitigate the transmission of redundant information in CPMs, the work in [7] evaluates and compares different redundancy mitigation algorithms. The work in [8] evolves the CPM generation rules proposed by ETSI with different techniques to improve the system's efficiency and scalability, and evaluates them in congested scenarios.

While simulation-based studies are relevant for the design and understanding of the operation of the CPS, real-world implementations are also required to validate the approved specifications and investigate its implementation feasibility and challenges. The implementation of cooperative perception is

challenged by the need to exchange updated information [9]. This requires minimizing the latency in the generation, transmission and reception of CPMs. The latency could be influenced not only by the wireless exchange of the information, but also by the processing of the information at the transmitting and receiving nodes. All these components are hard to accurately model in simulations and therefore their study requires real-world experimentation.

In this context, the main objective of this paper is the implementation, validation and testing of an ETSI standard-compliant Collective Perception Service. To the authors' knowledge, this is the first real-world implementation of the standardized CPS following the recently approved specification [1]. All the existing studies are based on the Technical Report published by ETSI in 2019, like [10] or [11]. The proposed implementation is validated in a laboratory environment with a series of tests and its scalability is studied under different conditions. Our work opens the door to more advanced studies with real-world experiments, that help identify potential issues that require revisions in the next version of the ETSI specification.

2. Collective Perception

2.1. Collective Perception Message

ETSI defines the format of the CPM in ASN.1 [1]. The specification of the message format is crucial for different providers/manufacturers to be able to exchange CPMs. Figure 1 illustrates the standardized CPM format. As it can be observed, the CPM is composed of a *header* and a *payload*. The *header* is common to other existing C-ITS messages defined by ETSI, such as the CAM (Cooperative Awareness Messages) and DENM (Decentralized Environmental Notification Message). The *header* contains the message ID, the protocol version, and the ID of the transmitting C-ITS station that generates the CPM. The *payload* is composed by the *managementContainer* and a set of *cpmContainers*. The *managementContainer* provides basic information about the transmitting vehicle or RSU (Road Side Unit) and the message, such as the CPM reference time, the reference GNSS position of the vehicle or RSU, message segmentation information, and the range of the CPM message rate that is expected or planned to be generated. Regarding the *cpmContainers*, the CPM must include one *originatingVehicleContainer* or *originatingRsuContainer*. These containers include more specific information about the transmitting vehicle or RSU, that is not included in the *managementContainer*, such as the orientation angle of the vehicle, or the map reference used by the RSU. The remaining *cpmContainers* are:

header	protocolVersion	
	messageID	
	stationID	
payload	managementContainer	referenceTime
		referencePosition
		segmentationInfo
		messageRateRange
	cpmContainers	originatingVehicleContainer or originatingRsuContainer
		perceivedObjectContainer
		sensorInformationContainer
		perceptionRegionContainer

Figure 1. CPM format defined by ETSI [1]

- *perceivedObjectContainer*: it contains information about the total number of perceived objects at the time the CPM is generated, and information about the specific perceived objects that are included in the CPM, such as their position, speed, acceleration, type (vehicle, animal, person, etc.) or their dimensions. The CPM can contain up to 255 perceived objects.
- *sensorInformationContainer*: it contains information about the capabilities of the sensors or data fusion systems used by the transmitting vehicle or RSU to generate the information about the perceived objects. This container can include information such as the sensor ID, the sensor type or the shape of the perception region and its confidence. It can contain up to 128 elements.
- *perceivedRegionContainer*: it contains information about the transmitter's perception capabilities, offering additional (often dynamic) details to the information provided in the sensor information container described above. It can contain up to 255 elements.

As specified in [1], the CPM must be encoded and decoded using UPER (Unaligned Packed Encoding Rules) [12].

2.3. CPM Generation Rules

ETSI also defines in [1] the rules that the CPS must follow for the generation of CPMs. These generation rules determine when the transmitting C-ITS station must generate a new CPM and what information (containers) should be included. All the generated CPMs must contain the *header* and the *payload* must include the *managementContainer* and one *originatingVehicleContainer* or *originatingRsuContainer*. The generation rules are then defined to determine the content of the *perceivedObjectContainer*, the *sensorInformationContainer* and the *perceptionRegionContainer*. The *sensorInformationContainer* is included once every second. The *perceptionRegionContainer* is included whenever a relevant deviation of the available dynamic perception capabilities is produced, with respect to the static perception capabilities described in the *sensorInformationContainer* or in previously included *perceptionRegionContainers*. The *perceivedObjectContainer* contains information about a subset of the perceived objects. Only a subset is normally included because the transmission of all the perceived objects in all the generated CPMs could saturate the radio channel [13][14][15]. To this aim, the CPM generation rules differentiate between two types of objects. Type-A objects include pedestrians, bicyclists, light VRU (Vulnerable Road User) vehicles, animals or VRU with profiles *groupSubclass* or *otherSubclass*. Type-B objects are objects of any other class (i.e., vehicles, or VRUs with profile motorcyclist). All detected objects of Type-A are included in a *perceivedObjectContainer* every 500 ms. If the perceived object is

of Type-B, it will be included in the *perceivedObjectContainer* if at least one of the following conditions is satisfied:

- If it is detected for the first time.
- If the difference between the current position and the position it had when it was included in a CPM is greater than 4 meters.
- If the difference between its current speed and the speed it had when it was included in a CPM is greater than 0.5 m/s,
- If the orientation of the velocity vector has changed by more than 4 degrees since the last time it was included in a CPM.
- If the time elapsed since the last time the object was included in a CPM has exceeded 1 second.

The CPM generation rules must be periodically checked every *T_GenCpm*. A CPM is generated if the rules for including a *perceivedObjectContainer*, a *sensorInformationContainer* or a *perceptionRegionContainer* are satisfied. If not, at least one CPM is generated every second, even if it does not contain any object.

Advanced functionalities are also defined in the standard for the inclusion of objects in the CPM, such as the one known as Look-Ahead [6], defined in section 6.1.2.4 [1]. This functionality allows anticipating the inclusion of an object in a CPM if it is expected to meet any of the conditions in the next checking instant. With this advanced functionality, larger CPM messages can be transmitted by grouping detected objects. This avoids the transmission of frequent and small CPMs that generate a significant overhead [6].

Following the CPM generation rules defined by ETSI, the time interval between consecutive CPM generation events, ΔT, can be calculated with the following equation in scenarios with one detected object of Type-B (e.g. a vehicle) that is moving at a constant speed, v, in meters per second.

$$\Delta T(ms) = \min\left(1000, T\_GenCpm \cdot \left\lceil \frac{4 \cdot 1000}{v \cdot T\_GenCpm} \right\rceil\right) \qquad (1)$$

3. Implementation

The CPS developed in this study has been implemented in C++ as part of a novel and scalable V2X stack. The core of the V2X stack is based on Vanetza[1]. Vanetza is an open-source implementation of the ETSI C-ITS protocol suite, that only includes simplified versions of CA (Cooperative Awareness) and DEN (Decentralised Environmental Notification) basic services. Our V2X stack adds the implemented CPS and is ready to integrate additional C-ITS services in upcoming developments. The proposed CPS generates CPMs following the ASN.1 format defined by ETSI [1]. CPMs are populated with information about detected objects, onboard sensors and perception regions following the ETSI CPM generation rules. This information is obtained from other sources that are part of the vehicle (e.g. the ADAS – Advanced Driver Assistance System) using a dedicated interface implemented as part of this study. When a new CPM is generated, the corresponding protocol headers are appended, and the CPM is transmitted. On the receiver side, all the CPMs are processed and sent back to the perception system, making use of the implemented interface.

3.1. Architecture

The CPS architecture has been designed to comply with ETSI specifications while ensuring its scalability. The architecture of the implemented CPS is illustrated in Figure 2. As it can be observed, the CPS has two interfaces: an internal interface for communication with other elements of the V2X stack, and an external interface for the transmission and reception of CPMs to/from other vehicles or RSUs. The main functions of the implemented architecture can be divided into the following three groups (from right to left in the columns of Figure 2):

- Functions for checking the CPM generation rules and the generation of CPMs.
- Functions for managing information received through the internal interface (e.g. information about perceived objects from the ADAS).
- Functions for the management of the CPMs received from other C-ITS stations through the external interface.

These functions are described in the next subsection.

3.2. Functions

When the CPS is launched, it creates an events handler (*CpmEventsHandler* type), which will manage the events that will affect the generation of CPMs. This corresponds to the first group of functions previously listed and illustrated in the right column in the architecture (Figure 2). As part of the *CpmEventsHandler*, a *timer_loop*() function is executed periodically (by default every *T_GenCpm* = 100 ms). This function will periodically call the *generate_request*() function. It triggers the execution of the *check_need_cpm*() function that checks if a CPM should be generated or not according to the generation rules defined in the standard and what information it should contain. In particular, this function identifies the detected objects that meet any of the conditions for generating a CPM. If at least one object meets one or more conditions, or if the last CPM was generated at least one

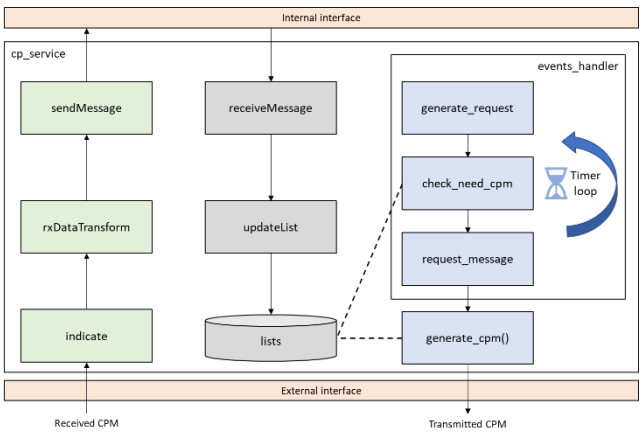


Figure 2. Architecture of the implemented CPS.

[1] https://www.vanetza.org/

second ago, a CPM is generated. To generate a CPM, the *request_message*() function is executed, which in turn calls the *generate_cpm*() function. This last function is in charge of populating the CPM containers to be sent, and UPER encode the message to deliver the result to the lower layers of the V2X stack (Transport and Network layers). In addition, this function updates the list of previously sent objects, which will be needed to check the CPM generation rules in the next iteration of the timer loop.

The second group of functions that are part of the architecture of the CPS (central column of Figure 2) is responsible for processing and storing the information provided by other internal elements of the system (e.g. ADAS) for the population of CPMs. In particular, these functions are used to maintain updated lists of perceived objects and sensor information. These lists can be accessed by the *CpmEventsHandler* for checking the CPM generation rules and generating a CPM. The input function is *receiveMessage*(), which will call the *updateList*() function.

The third group of functions (left column of Figure 2) is responsible for managing and processing the CPMs received through the external interface (from another V2X stack in another vehicle or RSU). The *indicate*() function is launched upon receipt of a CPM. In this function, *rxDataTransform*() is executed to transform the received CPM data into our internal data format. The result is sent through the internal interface to the internal elements/entities that are interested in received CPMs (for example, the ADAS).

4. Evaluation

The implemented CPS has been extensively evaluated through multiple tests in a laboratory environment, prior to its integration in a embedded Telematic Control Unit prototype to perform field tests with connected automated vehicles. These tests have been designed for the functional validation of the implemented CPS and to evaluate its scalability.

4.1. Cross-validation

Cross-validation is important to ensure the compatibility and the interoperability of the proposed implementation. The alignment of the implementation with standards is key to ensure its compatibility and interoperability. To this aim, our implementation has been cross-validated against third parties. This cross-validation is key to ensure its alignment with the ETSI specification, and that the CPMs generated with our implementation could be decoded by other implementations of the same standard, and vice versa. To perform this cross-validation, we have used the ASN1.*C Tools for C* software from the company OSS Nokalva, and in particular the *ASN.1 Studio*. This is a commercial professional tool that allows the generation of scripts in C from the ASN.1 files for the generation of CPMs, and generate CPMs automatically by choosing containers and randomly assigning values to the data elements. The user can select the encoding method, being UPER the one required by ETSI for radio transmission and XER the one we used to display the content since it is user readable.

To validate that the CPMs generated by our implementation can be decoded in *ASN.1 Studio*, we generated multiple CPMs in a *GTest* in the V2X stack and stored the result of their UPER encoding. The resulting CPMs in hexadecimal format were introduced in *ASN.1 Studio* and successfully decoded, showing a perfect matching of the content of each data field and data element of the CPM generated.

To validate that the CPMs generated using *ASN.1 Studio* can be decoded using our implementation, we follow a similar procedure but in the opposite direction. We generated multiple CPMs in *ASN.1 Studio* with randomized content following the standard ASN.1 format, and stored the result of the UPER encoding. The resulting CPMs in hexadecimal were ported to a *GTest* in the V2X Stack to decode them. Our implementation was able to successfully decode all of them and accurately extract their content.

4.2. Functional validation

The functional validation of the implemented CPS has been performed through a series of tests designed to verify the ETSI CPM generation rules. Each test emulates a scenario with one or more static or moving detected objects of different types (vehicles or pedestrians). The implemented tests are listed in Table 1. A *bash* script was implemented to automate their execution. Each test required running two V2X stack instances with the implemented CPS, each of them in a Docker container. Additionally, one external script was in charge of emulating the data generation from the ADAS system associated with one of the V2X Stack instances. This script is initiated after a random time interval between 1s and 3s and periodically creates and sends information about the perceived objects (every 50 ms) to the CPS. Depending on the test (see Table 1), the object type (vehicle or pedestrian) and its dynamics (stopped or moving) are differentiated. The CPS is configured with T_GenCpm = 100 ms so that the CPM generation rules are checked every 100 ms.

Table 1. Tests used for the evaluation of the implemented CPS

Test	Detected objects	Expected output
#1	One stopped vehicle	One CPM per second including the information about the vehicle
#2	One moving vehicle	One CPM every ΔT calculated with equation (1)
#3	Two moving vehicles	One vehicle included every ΔT_1 and the other every ΔT_2 following (1)
#4	One pedestrian	One CPM every 500ms including the pedestrian
#5	One moving vehicle and one pedestrian	The vehicle included every ΔT and the pedestrian every 500ms

Figure 3 depicts the time evolution of the events during the execution of test #1, in which there is only one detected object. The figure differentiates the events of received updates of detected objects, the checking of the CPM generation rules, and the CPM generation events. Once the V2X Start is initiated, CPMs are generated with a time interval of 1 seconds, since no objects are detected. The first update event with information about the detected object is received after 1.6s approximately, and the CPS generated a new CPM in the following check event. As a result, the first CPM that contains information about the detected object is generated with a time interval of 600ms. Since this first CPM containing information about the detected object, a new CPM is generated every second because the detected object is stopped. In this case, all the CPMs also contain information about the sensors, as it must also be included once per second.

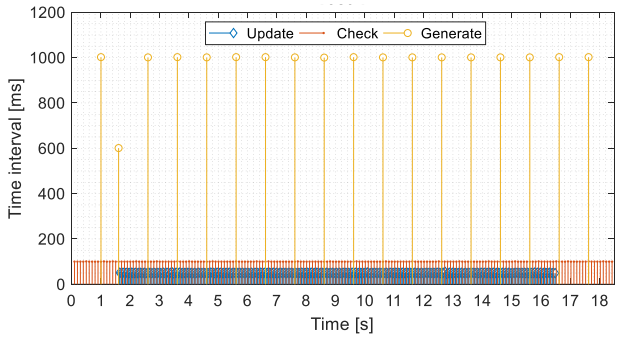


Figure 3. Time evolution of events for test #1.

The results corresponding to test #2 are shown in Figure 4 considering that the speed of the detected object is $v = 60$ km/h. At this speed, the CPMs should be generated every 300 ms according to equation (1). The results shown in Figure 4 clearly show that the generation rules are correctly implemented and validated for this speed, since all the CPMs are generated every 300 ms as long as the object is detected and new updates are received. When the object is not detected (no updates are received), CPMs are generated once per second, as can be observed in the figure. The CPM generation interval only deviates to 400 ms the first time the object is detected, since a new CPM is generated immediately following the ETSI generation rules.

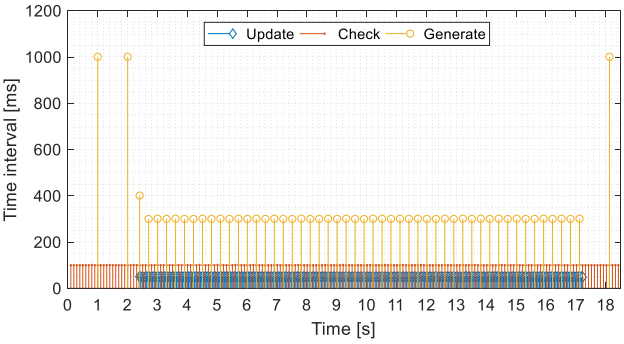


Figure 4. Time evolution of events for test #2 for $v = 60$ km/h.

Different speeds have been analyzed in test #2 and the resulting average CPM generation interval is compared in Figure 5 with the theoretical interval that should be obtained following the ETSI specifications (equation (1)). The results demonstrate and validate the implemented CPS for different speeds of detected objects.

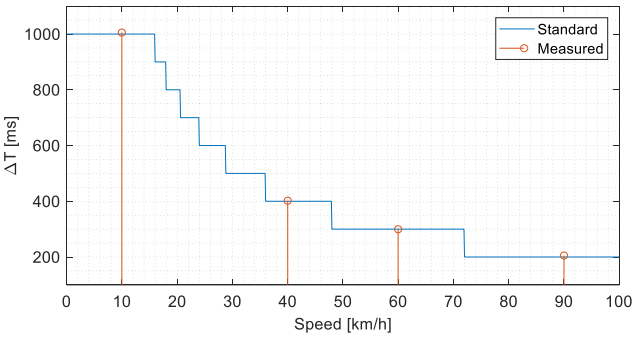


Figure 5. Theoretical and average time between CPMs generated for test #2 for different object speeds.

Test #3 considers two detected objects that are moving at different speeds. Both objects are vehicles. The speed of the first object is 60 km/h, and it therefore satisfies the CPM generation rules every 300 ms. The speed of the second object is 90 km/h and satisfies the rules every 200 ms. Figure 6 depicts the expected CPM generation pattern considering that both objects are initially detected at the same time. A CPM must be generated if at least one of the objects satisties the conditions. As a consequence, the CPMs should be generated in this scenario with the following sequence of intervals in milliseconds: {200, 100, 100, 200}. This sequence should be repeated over time. The execution of test #3 results in the events illustrated in Figure 7, that correspond to the correct CPM generation pattern illustrated in Figure 6. Figure 8a shows that effectively around 50% of the generated CPMs had an interval of 100 ms and the remaining 50% were generated with an interval of 200 ms.

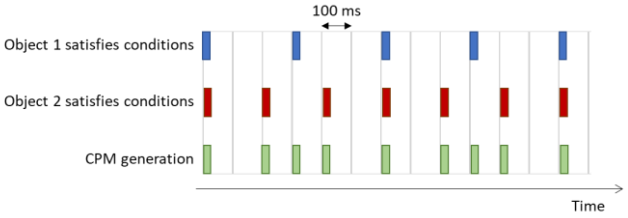


Figure 6. Expected CPM generation pattern in test #3 that considers two vehicles moving at 60 km/h and 90 km/h.

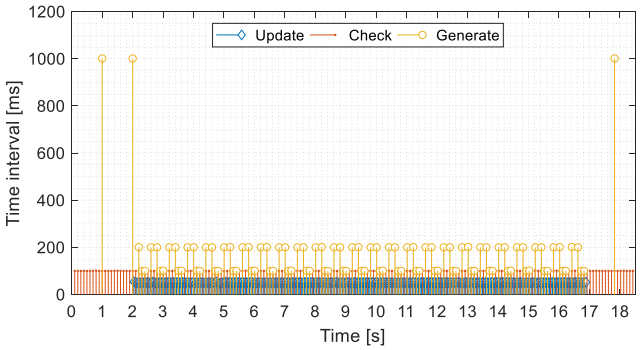
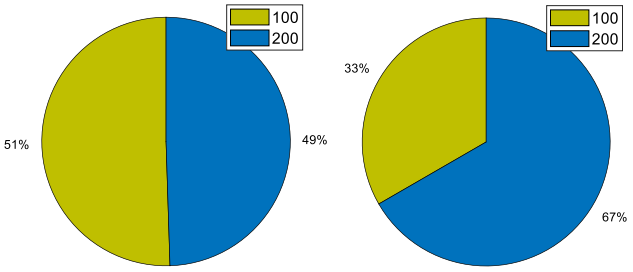


Figure 7. Time evolution of events for test #3.



(a) Test #3 (b) Test #5

Figure 8. Distribution of CPM generation intervals. Legend in milliseconds.

The results obtained for test #4, that considers a single detected object (pedestrian), are similar to the results of test #1. However, in test #4 all the CPMs generated that contain information about the pedestrian are generated with an interval of 500 ms. The time evolution of the events for test #4 is depicted in Figure 9. As it can be observed, only the first CPM that contains information about the object (i.e. the first CPM after the first update event) is generated with a different time interval (200ms in this case), since the generation rules dictate that a new CPM must be generated as soon as a new object is detected.

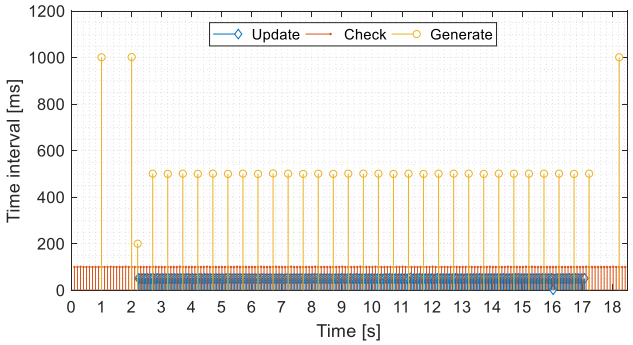


Figure 9. Time evolution of events for test #4.

Test #5 considers one object of Type-A (pedestrian) and one object of Type-B (vehicle) moving at 90 km/h. As illustrated in Figure 10, the pedestrian will satisfy the conditions every 500 ms, and the vehicle will satisfy them every 200 ms. As a result, the CPMs generated should follow the pattern {200, 200, 100, 100, 200, 200} that is also illustrated in Figure 10. The sequence of events associated with test #5 is presented in Figure 11. The results show that the CPMs generated follow the expected pattern. In fact, approximately 1/3 of the CPMs are generated with an interval of 100 ms and 2/3 with an interval of 200 ms, as illustrated in Figure 8b (this figure excludes the first CPM that contained information about the detected objects, which was generated with a time interval of 500ms, as shown in Figure 11).

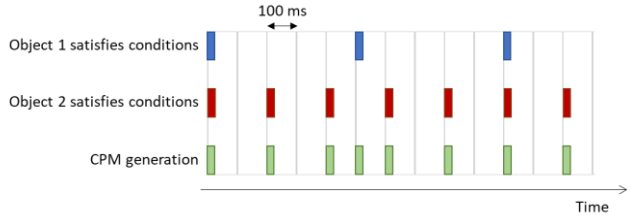


Figure 10. Expected CPM generation pattern in test #5 that considers one pedestrian and one vehicle moving at 90 km/h.

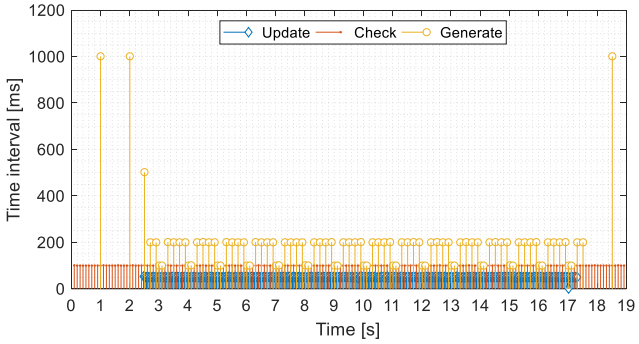
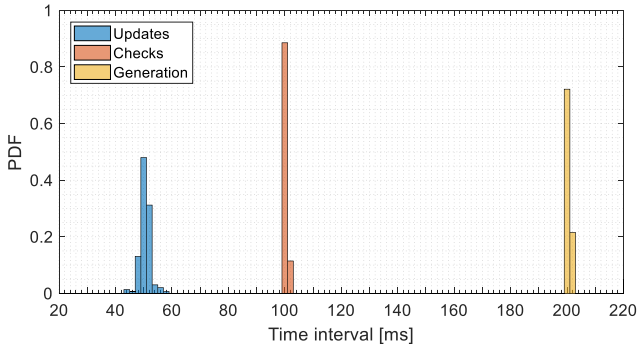


Figure 11. Time evolution of events for test #5.

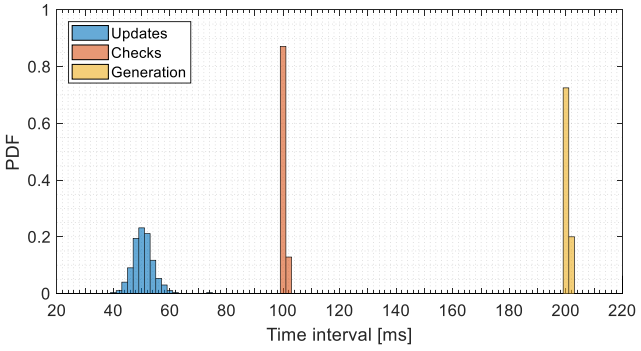
4.3. Scalability analysis

Scalability tests have also been performed to evaluate the implemented platform. The goal is to measure the latency introduced by the CPS in different parts of its internal architecture and help identify potential bottlenecks under different levels of stress at the transmitter side. The tests have been run in real-time in a laboratory environment with a laptop equipped with Linux Mint 20.3 Cinnamon, a CPU AMD Ryzen 3 5300U (2.6GHz) and 5.6GB of RAM. In these tests, two vehicles have been emulated, both of them transmitting and receiving CPMs. To this aim, in the same laptop two V2X stack instances are executed for the exchange of CPMs, each of them in a Docker container. In addition, the script that emulates the data generated by the ADAS of one of the V2X stack instances is also executed. Different experiments have been conducted by varying the number of detected objects, which is a key factor affecting the latency of the different components of the internal architecture of the CPM.

The first scalability analysis conducted evaluates the capacity of the implemented CPS in the considered set-up to schedule the events at the corresponding time intervals. Figure 12 shows the PDF (Probability Density Function) of the time between consecutive events for a test with multiple detected objects (vehicles). Figure 12a considers 5 objects and shows that the reception of updates about detected objects does not exactly occur at periods of 50ms, but certain deviation exists. This deviation is mainly due to the time required to prepare and send the objects information to the CPS. The results also show that the checks of the CPM generation rules are triggered every 100-105 ms in all cases. Similarly, the CPM generation events occur normally between 200 and 206 ms. Figure 12b depicts the same metrics when considering a scenario with 20 detected objects. The comparison of the results presented in Figure 12a (5 objects) and Figure 12b (20 objects) shows the small impact of the increase of the number of objects on the scheduling of the events. The higher difference is in the interval between updates about the detected objects, which is actually related to the capability of the ADAS emulation script to regularly generate new object updates, and does not significantly depend on the CPS implemented. Figure 13 concludes this first demonstration of the realtime performance of our implementation by showing the time evolution of the events for 20 detected objects.



(a) 5 detected objects



(b) 20 detected objects

Figure 12. PDF of time interval between events for a test with multiple detected objects of vehicle class.

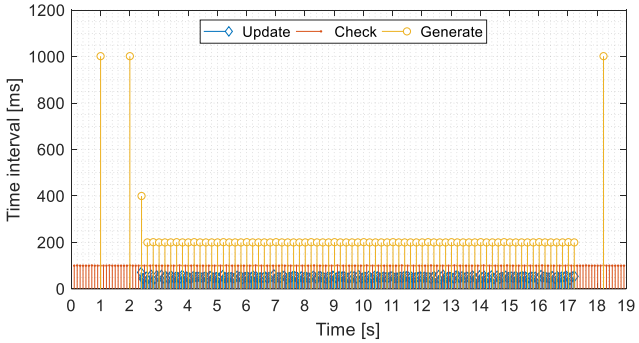


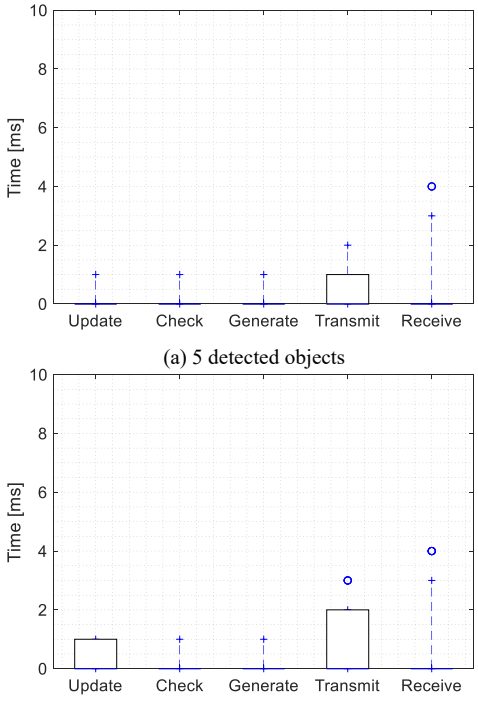
Figure 13. Time evolution of events for a test with 20 detected objects of vehicle class moving at $v = 90$ km/h.

The second scalability analysis conducted quantifies the time required to perform the main internal tasks identified in the CPS architecture. They are defined as follows:

- Update: time required to receive and process an update with new information about detected objects.
- Check: time required to check the CPM generation rules and determine if a new CPM must be generated.
- Generate: time required to populate a CPM and generate it.
- Transmit: time required to transmit the CPM to the lower layers.
- Receive: time required to receive and process a CPM to send it through the internal interface of the CPS.

Figure 14 shows the time required to execute each of these five tasks in two different tests, each of them with 5 objects (Figure 14a) and 20 objects (Figure 14b). The results are presented using boxplots, where the rectangle shows the IQR or interquartile range (75^{th} and 25^{th} percentiles), the vertical dashed lines represent the 5^{th} and 95^{th} percentiles, and the remaining samples are shown using markers (circles in this case). The resolution of the measured time intervals is in milliseconds. The results show that, with 5 detected objects (Figure 14a), the time required to execute the different functions for receiving new updates from the ADAS, checking the CPM generation rules and generating a CPM is lower or equal than 1 ms. The transmission of the CPM requires some more time in average, but it requires always less than 2 ms. The results also show that the reception of a CPM usually requires less than 1 ms, but in some cases the time required is between 3 ms and 4 ms.

With 20 detected objects (Figure 14b), the time required to execute these functions increases. We observe an increase of the time required to process a new update with information about the detected objects, but it does not go beyond 1 ms. Similar results are observed for the function that checks the CPM generation rules and for the function that generates CPMs. The figure shows that the function that transmits the CPMs requires 2 ms in 95% of the cases, and this time only increases up to 3 ms for some of the transmitted CPMs. The time required to receive a CPM is maintained.



(a) 5 detected objects

(b) 20 detected objects

Figure 14. Boxplot of the time required to execute different CPS functions for a test with multiple detected objects of vehicle class.

5. Conclusions

This paper presents the implementation of an ETSI standard-compliant Collective Perception Service (CPS) for connected and automated vehicles. To the authors' knowledge, this is the first implementation of the recently approved ETSI specification on collective perception. Extensive functional validation tests as well as cross-validation and scalability experiments have been conducted in a laboratory environment. Our findings show the importance of maintaining constrained delays to ensure the prompt generation and transmission of CPMs within the designated intervals outlined by the ETSI specification.

The results obtained also demonstrate the potential of the implemented CPS to be integrated in a Telematic Control Unit (TCU) for connected and automated vehicles. In fact, our research paves the way for conducting evaluations using embedded hardware devices through real-world experiments. Our future work will include the adaptation of the implemented CPS and the V2X stack to work over C-V2X and its evaluation using dedicated embedded 5G New Radio TCU from Idneo. Real-world experiments will be performed using these TCUs with the implemented V2X stack running over C-V2X as part of the InPercept project. These real-world experiments will consider two equipped vehicles in different use cases recommended by the C2C-CC [16]. These use cases could include the warning of potential collision with an occluded pedestrian to a vehicle with intention to turn right in a intersection, or the notification of the presence of an occluded car in the opposite lane of a two-way road in order to avoid a frontal collision during overtaking. These experiments could also include autonomous vehicle decision changing, such as the activation of emergency brakes or speed decrease. The main challenge of the real-world implementation and testing will be to

maintain acceptable scalability levels given the hardware limitations. These experiments will be used for the identification of potential issues that could require the revision of the ETSI specification.

References

[1] ETSI ITS, “Intelligent Transport System (ITS); Vehicular Communications; Basic Set of Applications; Collective Perception Service; Release 2”, ETSI TS 103 324 V2.1.1, May 2023.

[2] CAR 2 CAR Communication Consortium, “Guidance for day 2 and beyond roadmap”, Document number: 2072 v1.2, July 2021.

[3] D. D. Yoon, B. Ayalew and G. G. M. Nawaz Ali, "Performance of Decentralized Cooperative Perception in V2V Connected Traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6850-6863, July 2022.

[4] F. A. Schiegg, I. Llatser, D. Bischoff, G. Volk, “Collective Perception: A Safety Perspective”, Sensors 21(1), 159, 2020.

[5] H. H. J. Günther, B. Mennenga, O. Trauer, R. Riebl, L. Wolf, “Realizing collective perception in a vehicle”, *Proc. IEEE Vehicular Networking Conference (VNC)*, Columbus, Ohio, USA, pp. 1-8, December 2016.

[6] G. Thandavarayan, M. Sepulcre and J. Gozalvez, "Generation of Cooperative Perception Messages for Connected and Automated Vehicles", *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16336-16341, Dec. 2020.

[7] Q. Delooz et al., “Analysis and Evaluation of Information Redundancy Mitigation for V2X Collective Perception”, *IEEE Access*, vol. 10, pp. 47076-47093, 2022.

[8] G. Thandavarayan, M. Sepulcre, J. Gozalvez, B. Coll-Perales, “Scalable cooperative perception for connected and automated driving”, *Journal of Network and Computer Applications*, Volume 216, 103655, July 2023.

[9] G. Volk, Q. Delooz, F. A. Schiegg, A. Von Bernuth, A. Festag and O. Bringmann, "Towards Realistic Evaluation of Collective Perception for Connected and Automated Driving," *Proc. IEEE International Intelligent Transportation Systems Conference (ITSC)*, Indianapolis, IN, USA, pp. 1049-1056, 19-22 Sept. 2021.

[10] Y. Asabe, E. Javanmardi, J. Nakazato, M. Tsukada and H. Esaki, “AutowareV2X: Reliable V2X Communication and Collective Perception for Autonomous Driving”, *Proc. IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, Florence, Italy, June 2023.

[11] P. Almeida, A. Figueiredo, P. Rito, M. Luís and S. Sargento, "On the Real Evaluation of a Collective Perception Service", *Proc. IEEE/IFIP Network Operations and Management Symposium (NOMS 2023)*, Miami, FL, USA, pp. 1-6, 08-12 May 2023.

[12] Recommendation ITU-T X.691 (2021-02): "Information technology - ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)".

[13] H. -J. Günther, R. Riebl, L. Wolf and C. Facchi, "Collective perception and decentralized congestion control in vehicular ad-hoc networks," *Proc. IEEE Vehicular Networking Conference (VNC)*, Columbus, OH, USA, 8-10 Dec. 2016.

[14] ETSI ITS, “Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS); Release 2", TR 103 562 V, Dec. 2019.

[15] G. Thandavarayan, M. Sepulcre and J. Gozalvez, "Analysis of Message Generation Rules for Collective Perception in Connected and Automated Driving", *Proc. IEEE Intelligent Vehicle Symposium*, Paris (France), June 2019.

[16] CAR 2 CAR Communication Consortium, “Use Cases”, Document number: 2097 v1.0, March 2023.

Acknowledgement

This work is partially funded by *Centro para el Desarrollo Tecnológico y la* Innovación (CDTI) through the InPercept project (*Percepción Inteligente para los Vehículos Autónomos y Conectados*) Ref. PTAS-20211011. InPercept project is also supported by Spanish *Ministerio de Ciencia e Innovación* and receives funds from NextGenerationUE program.