Vehicular Message Exchange in Cross-border Scenarios Using Public Cloud Infrastructure

Ricard Vilalta*, Ramon Casellas*, Roshan Sedar*, Francisco Vázquez-Gallego*, Ricardo Martínez*, Soumya Kanti Datta†, Mathieu Lefebvre‡, Frederic Gardes‡, Jean-Marc Odinot‡, Jérôme Härri†, Jesús Alonso-Zarate and Raul Muñoz

*Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA), Castelldefels, Spain
†EURECOM, Sophia-Antipolis, France
‡Orange, Paris, France

***Abstract*—Cross-border scenarios are of extreme importance in current research work on 5G networks for connected vehicles. Network services and applications for connected vehicles, which are in specific cases expected to run on Mobile Edge Computing (MEC) infrastructure, might experience problems through country borders due to inter-domain. Given that cross-border scenarios typically imply a change of network operator, deployed MEC services need to work in such environments and, where applicable, different entities need to synchronize their input/output vehicular messages and work seamlessly in such a multi-operator context. One of most significant MEC services is *vehicular message brokering*. It consists on the controlled publishing and notification mechanisms to create awareness to all subscribed vehicles concerning their position as well as other significant events that may arise, such as hazardous events.**

This paper presents an architecture and method for vehicular message exchange, based on current Intelligent Transport Systems (ITS) standards, and proposes a novel hierarchical message brokering approach with the purpose of solving cross-domain scenarios, which can be applied not only in the aforementioned cross-border case but also in other scenarios where there is no single domain (i.e., with multiple vendors). Message Queuing Telemetry Transport (MQTT) servers are used in a hierarchical approach (locating a parent MQTT broker in a public cloud) in order to demonstrate the feasibility of using them for cross-border scenarios. Latency results are obtained in order to evaluate the performance penalty of the proposed solution.

***Index Terms*—Cross-Border V2X communication, Mobile Edge Computing, Cooperative Collision Avoidance, MQTT, CAM, DENM.**

I. Introduction

Connected vehicles are expected to provide significant societal benefits, such as less pollution and improved safety [1]. 5G networks are providing the necessary substrate infrastructure to support connected vehicles. Key elements for connected vehicles scenarios are the messages exchanged among vehicles (known as Vehicle-to-Vehicle, V2V) and between vehicles and the infrastructure (Vehicle-to-Infrastructure, V2I), which need to be delivered in a timely and robust manner.

The work in [2] provides a clear overview on Cooperative-Intelligent Transport Systems (C-ITS) standards in Europe. Two main protocols and their messages have become the basis for C-ITS services: Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM). On the one hand, CAM is a periodic message that provides status information to interested actors, and it is typically transmitted periodically when the engine is running. On the other hand, DENM is triggered only to notify a safety-related event.

C-ITS infrastructure is typically deployed following the Mobile Edge Computing (MEC) architecture, which provides the control and management of the desired C-ITS applications [3]. The work in [4] provides a high-level description of the Anticipated Cooperative Collision Avoidance (ACCA) system architecture and functionalities. One of the most important components of this architecture is a message broker, which is responsible for the distribution of CAM and DENM messages. A well known message broker architecture has been proven using the Message Queuing Telemetry Transport (MQTT) protocol. MQTT [5] is a lightweight simple messaging protocol based on a publish/subscribe model, designed for constrained devices and low-bandwidth. It has been used extensively in IoT applications. A central entity called MQTT Broker is responsible for message delivery.

In this work, we consider a cross-border scenario that involves multiple operators running different MEC applications (such as an MQTT broker). A hierarchical approach is needed to overcome multi-domain issues between borders. Several authors have demonstrated the usage of MQTT hierarchy in Internet of Things (IoT) scenarios [6]. In this paper, we apply the hierarchical MQTT approach to vehicular message brokering for the first time. With this objective, we describe the necessary architecture, the implementation of a proof-of-concept and evaluated results regarding latency overheads.

The remainder of this paper is organized as follows. In Section II, we provide an state of the art on MQTT message brokering for Vehicular-to-Everything (V2X) communications. In Section III, we describe how vehicular DENM and CAM messages are generated by a custom experimental On-Board Unit (OBU) software application. In Section IV, we present the architecture proposed for vehicular message exchange in a cross-border scenario. Later, in Section V, we provide examples of generated messages and the experimental validation of the proposed cross-border architecture. Finally, Section VI concludes the paper.

II. Related Work

The work in [7] presents the use of the IoT message protocol MQTT for transmitting accumulated and live V2X data from vehicles to a centralized big data platform. The usage of MQTT introduces a small overhead regarding network bandwidth and works on top of the standard TCP/IP protocol. Messages are transferred via an MQTT broker. The MQTT broker component is running next to the global database. According to the publish/subscribe mechanism, the component (i.e., vehicle) periodically publishes location and network performance related topics, which are then received by registered MQTT clients of 2D and 3D visualization applications.

Several research projects (e.g., 5G-DRIVE [8], 5G-Carmen [9]) are introducing MQTT brokers in the cloud in order to collect all traffic information in an efficient manner to deploy geographical services with minimal overhead. In the cloud broker, the data is organized according to geographical-related topics, which allows messages exchange without the need to share the client location. This approach provides privacy and saves communication bandwidth.

The authors in [10] have proposed the usage of virtualization technologies in the vehicular domain, which provides flexibility and reliability in real deployments, where mobility and processing needs may be an issue. The SURROGATES solution proposes to virtualize vehicle OBUs and creates a novel MEC layer with the aim of off-loading processing from the vehicle and serving data-access requests. Thus, this work opens a novel path regarding the virtualization of end-devices in the Intelligent Transportation Systems (ITS) ecosystem. This hierarchical approach of multiple hierarchical levels can be also observed in MQTT.

For example, the work in [6] proposes an integrated architecture based on the use of a public cloud infrastructure, considering a hierarchical approach for MQTT messsage publication and subscription. It demonstrates the feasibility of using a public cloud infrastructure to host a hierarchical parent MQTT broker. The objective of this paper is to apply this hierarchical approach to V2X cross-border scenarios.

III. Vehicular message generation

This section describes a custom experimental OBUs software application, which has been built with the idea of implementing in-vehicle OBU functionalities, including vehicular message generation and reception.

The OBU application can be deployed on a general-purpose computer with external connections to a Global Navigation Satellite System (GNSS) receiver and to the Controller Area Network (CAN) bus of the vehicle by means of an On-Board Diagnostic (OBD) interface adapter (e.g., ELM327 OBD-II). The OBU application can readily be configured onto a Linux system and is based on OpenC2X [11], which is an open-source software implementation of the Cooperative-Intelligent Transport System protocol stack (i.e., ITS-G5) defined in the ETSI ITS specification [12]. The interoperability of OpenC2X with a commercial OBU from Cohda Wireless has been demonstrated in [13] by sending and receiving CAM and DENM messages between OpenC2X and the Cohda Wireless OBU.

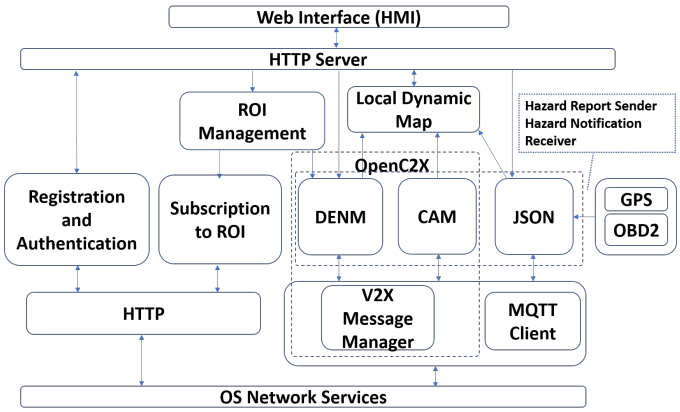

Figure 1: Software architecture of the OBU application.

The software architecture of the OBU is depicted in Figure 1. As it can be observed, it is composed of several independent software modules that communicate with each other using an asynchronous messaging library. In what follows, the core functionalities of each module are described.

The Web Interface module is a graphical user interface (GUI) that acts as the Human Machine Interface (HMI) of the vehicle, facilitating the interaction between the driver and the OBU application. Through the Web Interface, the driver is able to: initiate the registration and authentication process with the MEC-hosted back-end service; configure the subscription to the back-end service in order to receive DENMs associated to hazards located within a certain geographical area; and emulate the detection of a roadside event by triggering the transmission of a DENM to the back-end service. In addition, the Web Interface shows the current position of the vehicle, the location of road hazards and the position of other vehicles on a Local Dynamic Map (LDM). Furthermore, the Web Interface displays a notification message every time that a new DENM is received.

The HTTP server module facilitates the communication of the Web Interface with the rest of the modules of the OBU. The HTTP server receives HTTP requests from the Web Interface and sends commands to the corresponding software modules, which execute the required operations. As it can be observed in Figure 1, the HTTP server communicates with: the Registration & Authentication module, to initiate the registration and authentication process with the back-end service; the Subscription module, to configure the subscription parameters and initiate a new subscription to the back-end service; the DENM/JSON module, to trigger the transmission of a new DENM to the back-end service; and the LDM module, to query the positions of the vehicle, road hazards and other vehicles.

The Registration and Authentication module communicates with the back-end service via HTTP in order to register and authenticate the vehicle in the service. The Registration and Authentication module implements an HTTP client to send

REST commands that contain the credentials of the vehicle. In response to the request, the back-end service sends an authentication token that will be attached for any subsequent requests to the back-end service and will be used to verify the client by the back-end.

The Subscription to Region Of Interest (ROI) module configures the geographical coordinates and dimensions of the ROI, the types of events occurred in that region, and the duration of the subscription to the back-end service. The subscription to a ROI is performed using an HTTP POST request. It is required to notify the back-end service every time there is a change of ROI or when the previous subscription has already expired.

The DENM Service module is in charge of encoding and decoding DENM messages. In the current implementation, this module generates a new DENM when it receives a command from the Web Interface through the HTTP server. The DENM Service module fills the fields of the DENM with the geographical position (i.e., latitude and longitude) provided by the GNSS module and other in-vehicle sensors information provided by the OBD2 module. The DENM message is passed to the V2X Message Manager to add the GeoNetworking and BTP headers. In addition, the generation of DENMs could be triggered by external modules such as smart sensors for collision avoidance. Regarding the reception of DENM messages sent by the back-end service, the DENM Service module extracts the GeoNetworking header, decodes the fields of the DENM and forwards the relevant data of the event (type, position) to the LDM module.

The CAM Service module is in charge of encoding and decoding CAM messages serving as location updates for vehicle tracing at the back-end service. The CAM messages are triggered periodically. The position of the vehicle is provided by the GNSS module and the speed is provided by the OBD2 module. The CAM Service module fills the fields of the CAM message at the facilities layer, as defined by ETSI-ITS specification [14] and passes it to the V2X Message Manager where GeoNetworking and BTP headers are added. The CAM Service module also receives CAM messages sent by other vehicles and forwards the relevant data to the LDM module.

In order to communicate with entities such as Central Traffic Managers or additional back-end servers, an additional communications interface and protocol has been included in the OBU. It is based on the use of JavaScript Object Notation (JSON) text format to serialize CAM and DENM messages, and sent to other interested parties using the MQTT protocol.

The main concept is that clients connect and authenticate to a MQTT broker, susbcribe to the topics of interest (topics can be pre-defined based on geographic ROI, ITS protocol, etc.) and publish messages to the corresponding topic. We use a direct translation of CAM and DENM message mapping ASN.1 objects to JSON fields. Consequently, the MQTT Client module facilitates the communication via the MQTT protocol among the CAM module and DENM module and servers hosting back-end services. The MQTT module registers into an MQTT broker allocated in the distributed edge cloud, and it publishes on specific predefined topics the CAM and DENM messages generated by the CAM and DENM modules. The MQTT Client module subscribes to several topics in order to receive CAM and DENM messages published by the back-end service and forwards the incoming CAM and DENM messages into their respective CAM and DENM module. The JSON encoding follows closely the ASN.1 specification for DENM messages. In the example detailed in Listing 1, we show a simplified version of the DENM message in JSON format.

Listing 1: Simplified JSON DENM message

```
{
  "message":{
    "management_container":{
      "detection_time":503253332000,
      "event_position":{
        "altitude":1340,
        "confidence":{},
        "latitude":486263556,
        "longitude":22492123
      },
      "reference_time":503253330000
    },
    "message_id":1,
    "protocol_version":1,
    "situation_container":{
      "event_type":{
             "cause":1,
             "subcause":6
      },
      "information_quality":4,
      "linked_cause":{
        "cause":94,
        "subcause":2
      }
    }
  },
  "context":"etsi",
  "message_base64":"AQEAAA...",
  "origin":"on_board_application",
  "source_uuid":"",
  "timestamp":"1580723993599",
  "type":"denm",
  "version":"0.3.0"
}
```

The V2X Message Manager module facilitates the communication via the UDP protocol among the CAM Service and DENM Service modules and the back-end service. The V2X Message Manager adds the GeoNetworking and BTP headers, encapsulates the CAM and DENM messages into UDP packets and sends them to a UDP server allocated in the back-end. The V2X Message Manager receives UDP packets from the back-end service, decapsulates the CAM and DENM messages, extracts the GeoNetworking and BTP headers and sends them into their respective CAM Service and DENM

Service modules.

The LDM stores in a SQL-based database the information contained in incoming CAM and DENM messages. This data is read from the Web Interface module for real-time monitoring and can also be accessed for off-line analysis. In order to improve the performance, the latest data is stored in a cache.

The GPS Service module connects via Universal Serial Bus (USB) to an external GNSS receiver to collect periodic position updates (i.e., latitude, longitude, altitude) and forwards these data to the CAM Service, DENM Service and JSON/MQTT Client modules. In order to perform reproducible experiments, previously recorded GPS data can also be fed into the application in form of a trace. Instead of acquiring real positions from the GNSS receiver, the OBU application can read the position of the vehicle from a trajectory file that contains a sequence of geographical coordinates with associated time-stamps.

The OBD2 Service module communicates with the vehicle's on-board network (i.e., CAN bus) by means of an OBD interface adapter connected to an USB port of the OBU. The OBD2 module reads speed and acceleration from the OBD interface adapter and forwards these data to the CAM Service and DENM Service modules.

IV. Cross-border vehicular message exchange

In this section, we describe the proposed architecture for inter-vehicular message exchange shown in Figure 2. As stated in the previous section, JSON can be used for encoding CAM and DENM messages, becoming the payload of MQTT messages that can be sent (published) to a configured MQTT broker. For this, once these JSON messages have been generated, the MQTT client is responsible for publishing such messages to the MEC-based MQTT broker. In turn, this MQTT broker acts as child-broker (cMQTTi) running in a specific domain (1 or 2 in Figure 2) to enable a hierarchy of brokers, these MEC MQTT brokers are also able to export (publish) and receive (subscribe) the MQTT topics and JSON messages from a designated parent MQTT broker (pMQTT).

A public cloud is used to run an instance of this parent MQTT broker, responsible for forwarding (publishing) messages in the different topics to other involved child MQTT brokers and as well as for subscribing to the relevant topics in order to be notified by such brokers. In summary, this architecture allows overcoming the difficulties in multi-domain interconnection of MEC services and JSON messages are disseminated across multiple domains through multiple topic subscriptions. In the example of Figure 2, the MEC contained in Domain 1 receives vehicular messages from Domain 2, and vice-versa, and is able to propagate these messages to vehicles connected to its own domain.

Figure 3 presents the proposed sequence diagram for exchange of messages between vehicles in different domains. It consists of two different phases: initialization and run-time. The initialization phase consists on public cloud discovery, connection, authentication and subscription from the different child brokers located in MEC servers, which are related to

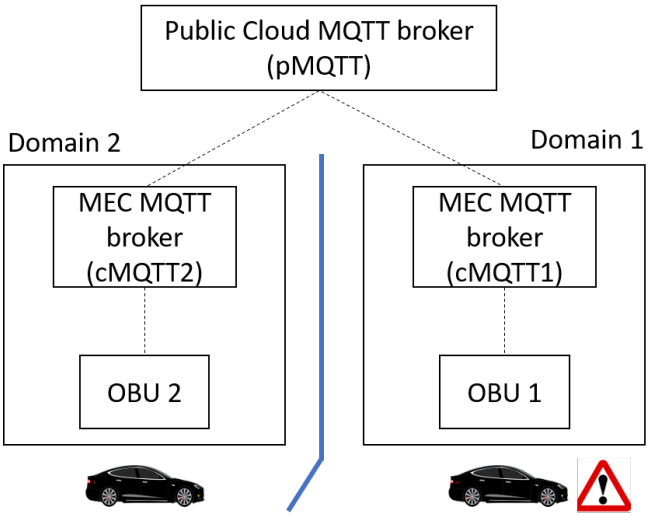

Figure 2: Proposed hierarchical MQTT architecture.

domain. In Figure 3 we depict subscription mechanism in initialization phase. Once it is completed, the run-time phase starts. Whenever a safety-related event is detected, a DENM JSON message is generated and transmitted from the OBU to its associated MEC, e.g., from Car1 to cMEC1. Then, cMEC1 propagates the message towards the pMQTT. The pMQTT disseminates the received message towards the rest of connected child MQTT brokers, and finally, once the child MQTT brokers receive the message, it is propagated to the subscribed OBUs.

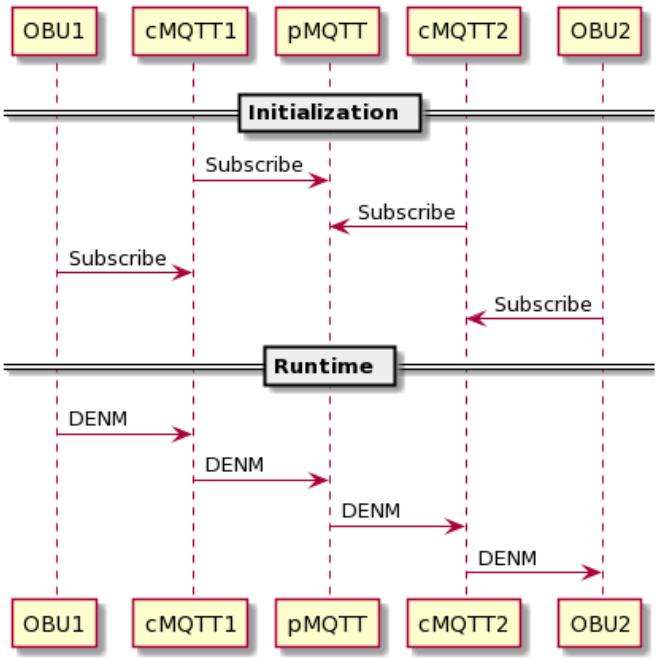

Figure 3: Initialization and Run-time Workflow of the proposed hierarchical architecture.

The proposed architecture can be applied to more complex scenarios, including multiple levels of hierarchy, where a parent message broker of hierarhical level H, acts also as child message broker of hierarchical level H+1. Also, hierarchy fault-tolerant strategies could also be considered, such as fat-tree [15].

Currently, each child message broker and public cloud broker are deployed by its own domain service orchestrator (NFV/MEC). This separates administratively each domain, while it provides the necessary cross-border interoperability. It is left for further research the design of a multi-domain orchestration mechanism that is able to deploy services in multiple administrative domains as well in the public cloud.

V. Experimental Evaluation

The evaluation is carried out in the CTTC 5G end-to-end experimental platform [16], which integrates heterogeneous wireless/optical networks, distributed cloud and IoT devices. In this work, we use two MEC servers and cloud resources for the deployment of the proposed architecture. Measurements have been obtained by using a dedicated 1 GB Ethernet connection between MEC servers and cloud resources. This provides a baseline scenario to benchmark the introduced latency of our implementation.

Firstly, we focus on the validation of the OBU, and finally, we validate the proposed message exchange architecture and evaluate its performance in terms of latency of message distribution.

A. OBU Validation

In order to validate the functionalities of the OBU application described in Section III, we have used the Wireshark protocol analyzer in order to capture and analyze CAM and DENM messages transmitted by the OBU over UDP as unicast packets to the back-end service deployed on the MEC infrastructure. Figure 4 shows an example of a sequence of CAM messages sent over UDP. Typically, CAM messages are being generated periodically upon the change of vehicles' dynamics such as the speed, change of direction, and so forth. For the sake of demonstration, as shown in Figure 4, we have set 3.0s as the periodicity to generate CAM messages.

Similarly, we have used Wireshark to capture and analyze the validity of CAM and DENM messages transmitted by the OBU over MQTT. Figure 5 shows an example of capture of a DENM message sent over MQTT in JSON format.

As shown in Listing 1, the JSON object contains top-level fields such as the source Universally Unique Identifier (uuid) of the entity generating the message, the type of message that is being encoded (e.g., "cam" or "denm"), the timestamp of the object (in milliseconds) when the message was generated since the common Unix Epoc (1970/01/01). It then includes multiple information objects. In every DENM message there is a management container that conveys, notably, the GPS position of the event (with optional confidence) and an situation container, that encodes the cause (and sub-cause) of the event. Standard causes may include accidents, road works, adverse weather conditions, emergency vehicles, etc.

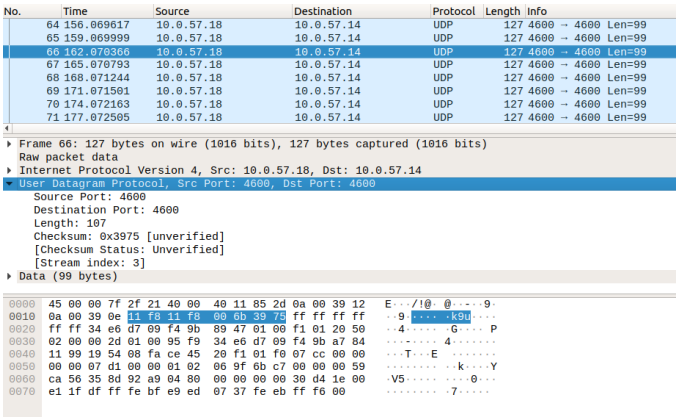

Figure 4: Wireshark capture of CAM messages sent over UDP.

B. Cross-border message exchange validation and evaluation

We have captured traces of messages transmitted between the pMQTT and two child MQTT brokers (cMQTT1 and cMQTT2) in order to validate the operation of the hierarchical MQTT architecture. Figure 5 shows the initialization and run-time phases with different time references for the exchange of messages between vehicles in domains 1 and 2. As it can be observed in Figure 5, the duration of the initialization between the parent and a child MQTT broker is of 1 millisecond. Then, it can be observed that the overhead latency in message exchange is of 0.2 milliseconds approximately. We describe as overhead latency, as the delay introduced from receiving a message at cMQTT1 and processing it in cMQTT2. This delay refers to the overhead introduced by this proposed hierarchical approach.

Figure 6 shows a Cumulative Density Function (CDF) of the message exchange latency overhead between cMQTT and pMQTT brokers. For this figure, we sampled 100 measurements and compute the cumulative histogram.

It can be observed that the introduction of hierarchical

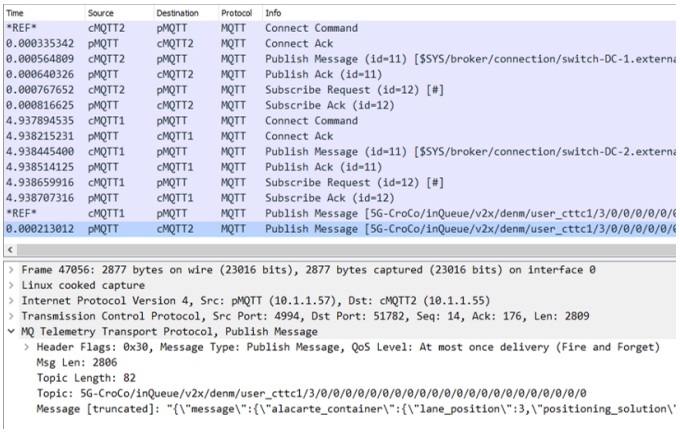

Figure 5: Wireshark capture of MQTT messages exchange in a cross-border scenario.

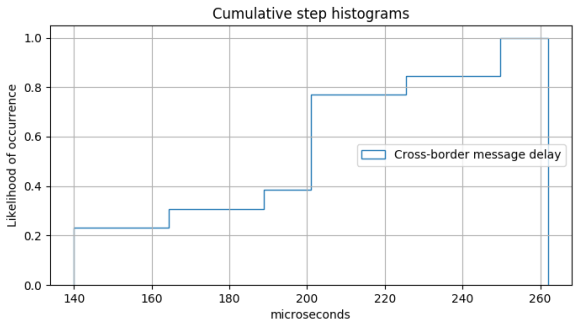

Figure 6: CDF of cross-border message exchange delay.

message exchange for cross-border scenarios comes with a penalty of around 0.2 milliseconds in latency overhead in the described testbed.

VI. Conclusion and Future Work

We have evaluated the feasibility of the proposed hierarchical vehicular message exchange solution and we can conclude that the message propagation delay does not affect the system, and might be of interest to consider to use it for cross-border deployments. This work is the basis for the future improvement of cross-border vehicular message exchange. The proposed solution shall be measured against complexity issues, such as cost and deployment of the public cloud (related to some traffic authority).

Several key features will be incorporated in future upgrades. The first update will include MQTT topic publication and subscription based on location-awareness and ROI selection. Second update will include dynamic deployment of MEC and Public Cloud applications using Service Orchestrator. The third update will be the introduction of multiple physical layer interfaces, such as 5G-NR and 802.11p. the proposed solution.

Acknowledgment

This work is part of 5GCroCo project that has received funding from the European Union H2020 Research and Innovation Programme under grant agreement No. 825050. It has also been partially funded by SPOT5G (TEC2017-87456-P) and AURORAS (RTI2018-099178) and by Generalitat de Catalunya under Grant 2017 SGR 891 and FEMIoT project, which is funded by the European Regional Development Fund (ERDF). EURECOM acknowledges the support of its industrial members - BMWGroup, IABG, Monaco Telecom, Orange, SAP, ST Microelec-tronics and Symantec.

References

[1] M. Fallgren, M. Dillinger, J. Alonso-Zarate, M. Boban, T. Abbas, K. Manolakis, T. Mahmoodi, T. Svensson, A. Laya, and R. Vilalta, "Fifth-generation technologies for the connected car: Capable systems for vehicle-to-anything communications," *IEEE vehicular technology magazine*, vol. 13, no. 3, pp. 28–38, 2018.

[2] A. Festag, "Cooperative intelligent transport systems standards in europe," *IEEE communications magazine*, vol. 52, no. 12, pp. 166–172, 2014.

[3] R. Vilalta, S. Vía, F. Mira, R. Casellas, R. Muñoz, J. Alonso-Zarate, A. Kousaridas, and M. Dillinger, "Control and management of a connected car using sdn/nfv, fog computing and yang data models," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 378–383.

[4] F. Vázquez-Gallego, R. Vilalta, A. García, F. Mira, S. Vía, R. Muñoz, J. Alonso-Zarate, and M. Catalan-Cid, "A mobile edge computing-based collision avoidance system for future vehicular networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2019, pp. 904–905.

[5] R. Light, "Mosquitto: server and client implementation of the mqtt protocol," *Journal of Open Source Software*, vol. 2, no. 13, p. 265, 2017.

[6] Y.-T. Lee, W.-H. Hsiao, C.-M. Huang, and T. C. Seng-cho, "An integrated cloud-based smart home management system with community hierarchy," *IEEE Transactions on Consumer Electronics*, vol. 62, no. 1, pp. 1–9, 2016.

[7] F. Pinzel, J. Holfeld, A. Olunczek, P. Balzer, and O. Michler, "V2v- and v2x-communication data within a distributed computing platform for adaptive radio channel modelling," in *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, 2019, pp. 1–6.

[8] M. Kutila, L. Nykänen, and M. Lankinen, "5g-drive: Eu china c-v2x collaboration," in *8th Transport Research Arena, TRA 2020-Conference cancelled*, 2020.

[9] N. Slamnik-Kriještorac, H. C. C. de Resende, C. Donato, S. Latré, R. Riggio, and J. Marquez-Barja, "Leveraging mobile edge computing to improve vehicular communications," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2020, pp. 1–4.

[10] J. Santa, P. J. Fernández, J. Ortiz, R. Sanchez-Iborra, and A. F. Skarmeta, "Surrogates: Virtual obus to foster 5g vehicular services," *Electronics*, vol. 8, no. 2, p. 117, 2019.

[11] S. Laux, G. S. Pannu, S. Schneider, J. Tiemann, F. Klingler, C. Sommer, and F. Dressler, "OpenC2X - An Open Source Experimental and Prototyping Platform Supporting ETSI ITS-G5," in *8th IEEE Vehicular Networking Conference (VNC 2016), Demo Session*. Columbus, OH: IEEE, December 2016, pp. 152–153.

[12] ETSI, "TS 102 636-3 - V1.1.1 - Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking," March 2010.

[13] F. Klingler *et al.*, "Field Testing Vehicular Networks using OpenC2X," in *15th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys 2017), Poster Session*. Niagara Falls, NY: ACM, June 2017, pp. 178–178.

[14] ETSI, "ETSI EN 302 637-2 V1.3.1 - Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications," September 2014.

[15] D. Sima and P. Kacsuk, *Advanced computer architectures*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[16] R. Munoz, J. Mangues-Bafalluy, R. Vilalta, C. Verikoukis, J. Alonso-Zarate, N. Bartzoudis, A. Georgiadis, M. Payaro, A. Perez-Neira, R. Casellas *et al.*, "The cttc 5g end-to-end experimental platform: integrating heterogeneous wireless/optical networks, distributed cloud, and iot devices," *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 50–63, 2016.