Internet of Vehicles Interoperability using CoAP Protocol for Streaming Applications

Victor Emanuel F. C. Borges, Dalton C. G. Valadares and Danilo F. S. Santos

***Abstract*— The Internet of Vehicles is widely applied in the context of connected systems nowadays. A vehicular network, a structure that provides vehicle-to-everything communication, can solve many issues in IoV. However, it deals with connectivity restrictions such as volatility, latency, bandwidth, and interoperability. To deal with these issues, applications can use lightweight IoT protocols such as the Constrained Application Protocol (CoAP). This paper presents a practical use case implemented using the CoAP protocol, considering a Multi-access Edge Computing (MEC) scenario, which runs cloud services at the network's edge and performs specific interoperable tasks. The evaluated scenario considers a V2X Streaming Service application that uses the V2X Information Service, an API to facilitate the interoperability between V2X communications and external applications. The study concludes that CoAP can fulfill the scenario's connectivity requirements and facilitate interoperability between IoT services.**

***Keywords*— Internet of Vehicles, Vehicular Network, Constrained Application Protocol, V2X Information Service**

I. Introduction

The Internet of Vehicles (IoV) is growing rapidly in today's connected world. IoV provides communication between vehicles and road infrastructure by using vehicular networks as its network structure. It can help people to get real-time traffic information, making the trip more comfortable and convenient, and even guiding drivers to avoid traffic accidents [1]. Currently, many car manufacturers are investing in IoV for their products, such as General Motors, BMW, Mercedes Benz, Nissan, and Volkswagen [2].

Vehicular networks can solve the majority of issues in the IoV context. However, since vehicles are in motion at all times, vehicular networks deal with volatility and the need for instant responses, especially in safety-related applications. This way, vehicular networks have important connectivity requirements that need to be taken care of, such as low latency, high bandwidth, and high availability [3]. Ensuring these requirements can be crucial in critical decision moments, such as in a collision-avoidance service. In addition, there must also be interoperability between IoV devices and other IoT services, such as weather and urban traffic applications.

A previous study evaluated application-level protocols that could enhance these services [4]. The authors evaluated the following protocols: Hypertext Transfer Protocol (HTTP), the most used protocol on the Internet applied to various contexts such as websites and general context web applications; Message Queuing Telemetry (MQTT), a protocol designed for devices with resource constraints or limited network bandwidth that make connections with remote locations; and Constrained Application Protocol (CoAP), a lightweight protocol based on the HTTP for use between devices with constrained resources demanding low power, between devices and general nodes on the Internet, and between devices on different constrained networks both joined by an Internet.

The study concluded that communication using CoAP, which has a lightweight structure widely used in IoT solutions, presents relevant results in terms of latency and bandwidth compared with standard solutions such as the Basic Safety Message (BSM), the communication pattern of vehicular networks. The experiment also showed that CoAP can still favor interoperability considering the connectivity requirements.

Interoperability between different systems is a problem frequently cited in IoT and IoV studies. Interoperability is the skill of applications and devices to communicate with each other. Sensors and devices embedded in vehicles, such as Road Side Units and Application Units, built by different manufacturers use different protocols [5].

This challenge can be difficult when adding other IoT services, such as smart manufacturing, smart government, mobility/wi-fi, smart health, smart farming, and smart transportation. IoV can integrate these areas, generating even a concern about security, privacy, complexity, and standardization.

Nowadays, IoT architectures are built on heterogeneous standards, such as Constrained Application Protocol (CoAP), Message Queueing Telemetry Transport (MQTT), Sensor Web Enhancement (SWE), Lightweight M2M (LWM2M), oneM2M, and other proprietary and different interfaces [6].

Given that, most IoT services and applications offer heterogeneous ways to transfer data and communicate with objects. These distinct ways can cause many interoperability problems when developers create cross-platform and cross-domain services. It can also be a barrier to business opportunities, especially for startup companies with a low budget that cannot pay to provide solutions across multiple platforms, being able to provide applications only for a restricted number of platforms. Figure 1 presents an example of different application domains that need to handle interoperability.

In this sense, the current literature presents works that deal with the interoperability problem. Blackstock et al. [7] defend the use of IoT hubs to aggregate things using web protocols and suggest a staged approach to interoperability. Xiao et al. [8] propose an interoperability framework to enable device users to interoperate with heterogeneous devices of

Victor Emanuel F. C. Borges, Dalton C. G. Valadares and Danilo F. S. Santos; Future Connected Systems, Embedded Systems and Pervasive Computing Laboratory, Federal University of Campina Grande, Campina Grande-PB, Brazil, e-mails: {victor.borges, danilo.santos, dalton.valadares}@embedded.ufcg.edu.br.

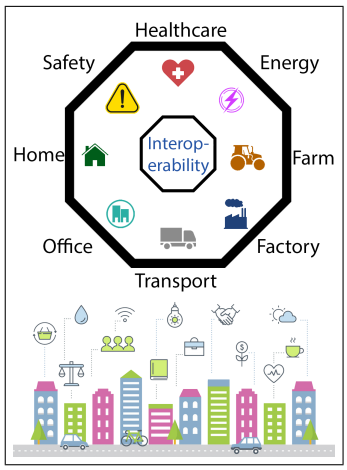

Fig. 1. Interoperability among domains.

different contexts with consistent semantics and syntax. In this solution, the authors provide a new separation strategy, a device representation method for real and virtual devices, and a device transformability model to guarantee the proper transformation of device syntax and semantics.

We can see interoperability in IoT and IoV is still a major challenge currently, and this is why we explore it in this article. Thus, we decided to validate CoAP with a practical use case implementation in an Edge Computing scenario and analyze how this protocol behaves, especially in non-safety applications, which benefit the most, with services at the edge [9]. In this article, we present an implemented application to evaluate CoAP in a vehicular scenario that uses the V2X Information Service (VIS) API [10] in a streaming service application. The VIS API is a Multi-access Edge Computing (MEC) service that aims to facilitate the interoperability of V2X communications between multiple networks and a multi-access environment, acting as an intermediary to exchange user information with the service. Therefore, the VIS API assists the devices' communication in a vehicular network with external services at the network edge. We used this API to facilitate communication between the Road Side Unit and the V2X Streaming Service application.

The remainder of this article is organized as follows. Section II presents related work on interoperability applied to IoV services. Section III brings an overview of CoAP, VIS, and MEC concepts. Section IV provides the validation plan of the experiment, including the requirements. Section V describes our implementation. Section VI describes the execution of the experiment and shows the results. Finally, Section VII concludes this paper.

II. Related Work

In the field of interoperability applied to IoV, King et al. [11] suggested a translation protocol with cognitive resources that makes possible translating communication between LTE sidelink device-to-device, mostly used in cellular messages, and DSRC packets of IEEE 802.11p pattern, allowing interoperability between these two types of technologies. On the other hand, Baltar et al. [12] proposed using a Multi-access Edge Computing (MEC) infrastructure to enable interoperability between different communication systems in vehicular scenarios, putting them to operate in the same band at the same location.

Casademont et al. [13] presented a project, called H2020 Caramel Project, following the MEC concepts that could provide the necessary performance for multi-radio communications inside V2X environments. Hadiwardoyo et al. [14] proposed a testbed for vehicular scenarios focused on interoperability across external communications.

Finally, we noticed little relevant research about interoperability in IoV scenarios, and none of them approaches CoAP communication in specific IoV scenarios, which is the focus of this research.

III. Background

This section briefly describes the main concepts approached in this work: CoAP, VIS API, and MEC.

A. Constrained Application Protocol (CoAP)

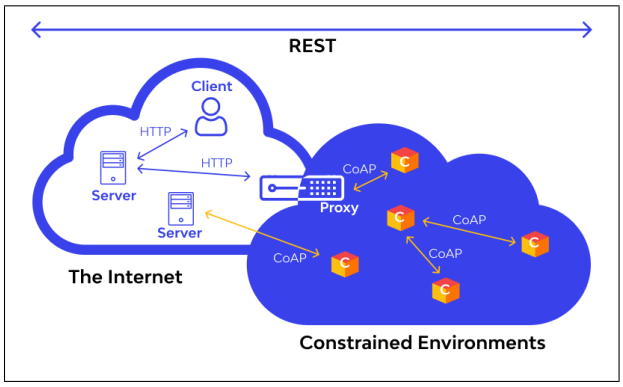

Fig. 2. CoAP and HTTP communication [15].

It is a communication protocol at the application level widely used in IoT solutions, especially when the devices have constrained resources [16]. The protocol works for device-to-device applications, such as the vehicle-to-vehicle communication of IoV. CoAP follows a communication model similar to HTTP but with particularities such as REST communication. In device-to-device communications, CoAP makes both entities act as client and server at the same time. The way requests and responses work is similar to HTTP. However, CoAP handles the information exchange asynchronously over a datagram-oriented transport. For this reason, CoAP uses connectionless transport protocols such as UDP [17]. Figure 2 presents the difference between HTTP and CoAP communication models.

Regarding the message format, a CoAP message starts with a header of a 4-byte fixed length, containing version information, type, token length, a response code, and a message ID. Then the message has a token value of variable length, which can be between 0 and 8 bytes in length, in addition to options and payload values, which are optional [18]. The main advantage of the CoAP lies in the way it is lightweight, working well for devices with restrictions, whether related

to connectivity, hardware resources, or consumption power. Therefore, it is most often used in IoT projects, as there are many restrictions in all areas, including vehicular networks.

B. V2X Information Service (VIS)

The MEC Vehicular-to-Everything (V2X) Information Service (VIS), or V2X Information Service, is an API in order to facilitate V2X interoperability in a multi-vendor, multi-network and multi-access environment. It describes the V2X related information flows, required information and operations.

The service consumers communicate with VIS over the VIS API to get the necessary V2X service provisioning information for the visiting PLMN in support of inter-PLMN service continuity. Both the MEC applications and the MEC platform may consume the VIS; and both the MEC platform and the MEC applications may be the providers of the V2X information [19].

The VIS API supports both queries and subscriptions (publish / subscribe mechanism) that are used over the RESTful API or over alternative transports such as message bus.

C. Multi-access Edge Computing (MEC)

Multi-Access Edge Computing (MEC) moves the computing of traffic and services from a centralized cloud to the edge of the network and closer to the customer. Instead of sending all data to a cloud for processing, the network edge analyzes, processes, and stores the data. Collecting and processing data closer to the customer reduces latency and brings real-time performance to high-bandwidth applications [20].

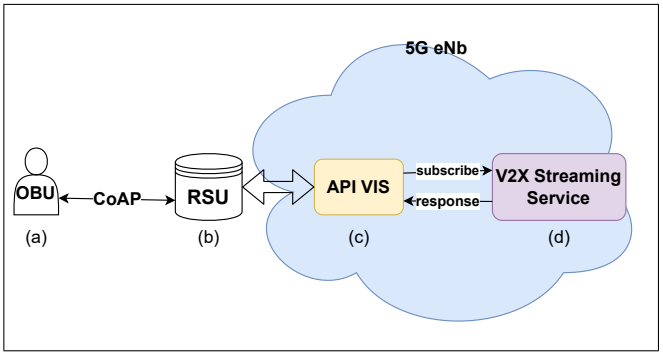

Fig. 3. Experiment validation plan.

MEC has some particular caracteristics such as proximity, high bandwidth, virtualization and ultra-low latency. It also offers cloud-computing capabilities and an IT service environment at the edge of the network. Professionals typically implement MEC with data centers that are distributed at the edge. Applications at the edge require a high bandwidth and low latency environment. To achieve that service providers create distributed data centers, or distributed clouds. The resources that make up a cloud can reside anywhere—from a centralized data center to a cell site, a central office, an aggregation site, a metro data center, or on the customer premises. The MEC platform enables distributed edge computing by processing content at the edge using either a server or a CPE.

A software-defined access layer could also be used as an extension of a distributed cloud. Most edge computing initiatives are being developed using open source hardware and software that leverage cloud and virtualization paradigms, including SDN and NFV.

IV. Validation Plan

Figure 3 presents the scenario considered for the validation. The objective of the experiment is to perform a CoAP communication between an On Board Unit (OBU) present in a vehicle (Fig. 3 (a)) and a Road Side Unit (RSU) of the road infrastructure (Fig. 3 (b)). This RSU, in turn, communicates with the network edge through the API VIS (Fig. 3 (c)), which mediates the entities involved.

In terms of validation, the VIS API thus makes the interoperability between the RSU, positioned at the edge of the network, and an application V2X Streaming Service (Fig. 3 (d)), categorized as non-safety-related, thus favoring the use of CoAP in communication. The API, in turn, requests information from the streaming service via a subscribe message and receives a return via a response message. It is important to note that the V2X Streaming Service can be on a 5G network, which favors MEC services such as the VIS API [21].

To compare with CoAP, we planned the same scenario considering the implementation with HTTP for the communication between OBU and RSU. With this experiment, we could compare both protocols in terms of latency and verify that the CoAP delivers adequate performance.

To guide the validation objective, we defined the following requirements for the experiment:

- *Requirement 1* - The communication of the edge V2X Streaming Service with the OBU, and the OBU with V2X Streaming Service, must occur end-to-end, with the CoAP protocol acting on the communication and the VIS API operating as a mechanism for information exchange between the external environment and the vehicular network.
- *Requirement 2* - The CoAP protocol must deliver a delay consistent with the HTTP results without drastically deviating in mean and standard deviation values in terms of latency.

In order to achieve the validation, the experiment results should meet both requirements. To fulfill them, we created an implementation related to the validation plan, which is detailed in the following subsection.

V. Implementation

Figure 4 shows the implementation schematic of the solution project. Based on the description of the VIS API offered by ETSI, we used Python 3 with the Flask[1] framework, which helps in the creation of APIs.

For this experiment, we have implemented only two VIS API services. The first one is the “Publish V2X Message”, which is used for the API to perform a publish of a message transmitted to another service. This method is of type POST

[1]https://flask.palletsprojects.com/en/2.0.x/

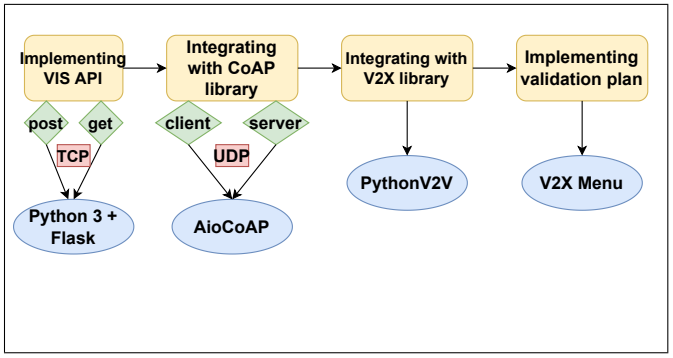

Fig. 4. Solution project implementation diagram.

and receives a JSON file with the message content data, encoding format, message type, and default organization [22]. The other implemented VIS API service is the "Provisioning Info", which is used for the API to query provisioning information for vehicular communication. The method is of type GET and receives an ID from the OBU or RSU to consult [23].

After the VIS API is implemented, the next step is to integrate the implementation with a CoAP communication library. For this, we used AioCoAP,[2] as it is a complete library that uses native asynchronous methods to facilitate simultaneous operations while maintaining an easy-to-use interface. In addition, we implemented AioCoAP in Python 3, which facilitated the integration with the implementation project.

The CoAP adaptation took place through two code classes: one representing the CoAP client and the other representing the CoAP server. While the CoAP client defines the protocol request path for the OBU to request information from the service at the edge, the CoAP server defines the opposite path, that is, when the V2X Streaming Service sends its information to the OBU. Both classes present in AioCoAP are instantiated in the main class of the V2X Menu and called when communications involving entities occur.

As the last step, after integrating the code with the CoAP classes from AioCoAP, we performed the integration with a vehicle communication library. We chose the PythonV2V[3] library, as it offers an easy-to-understand structure, in addition to allowing the creation of vehicles and RSUs to communicate. Another advantage of this library is the presence of some applications, such as the Streaming Service application used in this experiment. The only disadvantage of PythonV2V is its implementation in Python 2, requiring some adaptations to integrate with the implementation's Python 3.

To integrate the PythonV2V library, it was first necessary to instantiate the Vehicle and RSU classes and call the method that communicates between them. After that, we also instantiated the V2X Streaming Service application in the main class directly from the StreamingApp module of the referred library. Finally, we carried out the integration between CoAP and VIS communications with the libraries.

After we performed all the integrations, we implemented the validation project, which instantiates a vehicle OBU and an RSU, and allows communication from the RSU to the OBU with information from the application V2X Streaming Service, and a request communication from the OBU to the RSU, both going through the VIS API. We named the solution created for this experiment as V2X Menu.[4]

Finally, to allow comparisons with the HTTP protocol, we implemented a variation of the V2X Menu using HTTP instead of CoAP. In this variation, we integrated the code with an HTTP communication library. For this, we used "HTTP for Humans"[5] library. As it is also implemented in Python 3, the library facilitates integration with our implementation and AioCoAP.

VI. Execution & Results

To obtain greater statistical fidelity, we executed each of the two communication options (streaming service for OBU or OBU for streaming service) a total of 150 times with CoAP and 150 times with the HTTP variation, generating 600 samples from the experiment, an amount sufficient for an acceptable statistical level.

In each execution of the experiment, we executed the CoAP server of the AioCoAP library and the Flask application developed to simulate the VIS API on the machine. With these two services running, it is possible to run the main code of the V2X Menu implementation. When starting the V2X Menu, we can choose between sending a streaming message to the vehicle or requesting vehicle information for the streaming service. In addition, we can choose the protocol the application will use, whether CoAP or HTTP. When choosing one of the options, the scenario's objects are instantiated, and then the communication is carried out successfully in case it returns "Message sent successfully".

The experiments demonstrate that the requirements were met. Regarding "Requirement" 1, the communications from the V2X Streaming Service to the OBU and from the OBU to the V2X Streaming Service were performed correctly in all samples of the experimental validation. The connection to the streaming service and the communication with the CoAP happened as expected, where all executions got the return "Message sent successfully" from the VIS API. The path was followed end-to-end according to the validation plan flow, with the CoAP protocol acting on the communication and through the VIS API intermediary with the post function of the Publish V2X Message.

Regarding "Requirement 2", the results of replications delay were compiled and calculated in terms of mean delay, standard deviation, and 90% confidence interval. Table I shows the average results of V2X Menu communications in milliseconds. These results show that the average delay is higher in HTTP, either in the communication between OBU to RSU or between RSU to OBU. However, both cases remain in the expected range of average delay for vehicular applications [24]. With these results, it is possible to conclude that using CoAP in communication delivers better results than HTTP, with delays tending to be smaller and without being outside the expected range of average delay, satisfying this requirement.

[2]https://aiocoap.readthedocs.io/en/latest/
[3]https://github.com/islamdiaa/V2V
[4]https://github.com/LDVictor/V2XInformationServiceAPI
[5]https://requests.readthedocs.io/en/latest/

TABLE I
AVERAGE RESULTS OF V2X MENU COMMUNICATIONS (MS).

Protocol	Scenario	Delay	SD	CI (90%)
CoAP	RSU-OBU	8,1065	2,4976	[5,6089; 10,6041]
	OBU-RSU	10,4676	2,6891	[7,7785; 13,1567]
HTTP	RSU-OBU	30,6680	26,0237	[4,6443; 55,6917]
	OBU-RSU	23,4137	17,9130	[5,5007; 41,3267]

This experiment concludes that CoAP, together with the VIS API, can provide interoperability between an external service at the edge of the network and vehicular devices. Therefore, CoAP is suitable for a scenario of applications at the edge of the network, as it favors communication with its great interoperability capacity, which is enhanced through the use of APIs such as VIS.

A. Potential Threats

It is important to point out that, since the implementation is a restricted experiment, in a simulated environment, in real experiments the results may behave differently. Furthermore, communication delays in a real environment, both with the use of HTTP and with the use of CoAP, tend to be slightly higher due to unpredictability, signal attenuation and communication range distances.

VII. CONCLUSION & FUTURE WORK

This paper presents an experimental solution in a scenario of simulation which uses CoAP as the application protocol and a V2X Streaming Service as an application running in the network. First of all, an introduction was explained and an overview of the concepts used in the experiment was presented. About the simulation, a validation plan was planned, with an OBU and a RSU communicating with each other, and the API V2X Information Service doing the intermediary between the streaming application running in the edge and the vehicular network running outside the edge. The implementation was done with modules of Python 3 + Flask, AioCoAP and PythonV2V to make CoAP implementation, and module HTTP for Humans to make HTTP variation for comparison, integrating them to create V2X Menu, the name of the final implementation. The execution was done successfully and the results fulfill the defined requirements. As future work, it is important to investigate this experimental validation with other types of applications and in other scenarios, such as vehicle platooning.

ACKNOWLEDGMENT

The authors would like to thank VIRTUS - Research, Development and Innovation Center, and the Informatics Postgraduate Program (COPIN), from Federal University of Campina Grande for supporting this research.

REFERENCES

[1] KOMBATE, Damigou et al. *The Internet of vehicles based on 5G communications.* In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2016. p. 445-448.

[2] TODD, Felix. *Connected Cars: Ten companies using the Internet of Things to improve driving experience.* Available in: https://www.ns-businesshub.com/technology/connected-cars-iot/. NS Business, 2018.

[3] BORGES, Victor Emanuel F. C. et al. *Survey and Evaluation of Internet of Vehicles Connectivity Challenges.* In: 2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM). IEEE, 2020. p. 1-6.

[4] BORGES, Victor Emanuel F. C. *Avaliação de protocolos para a internet das coisas em redes veiculares.* Available in: http://dspace.sti.ufcg.edu.br:8080/jspui/handle/riufcg/23612.

[5] HUSSAIN, Shaik Mazhar et al. *A review of interoperability issues in Internet of vehicles (iov).* International Journal of Computing and Digital Systems, v. 8, n. 01, p. 73-83, 2019.

[6] BRÖRING, Arne et al. *Enabling IoT ecosystems through platform interoperability.* IEEE software, v. 34, n. 1, p. 54-61, 2017.

[7] BLACKSTOCK, Michael; LEA, Rodger. *IoT interoperability: A hub-based approach.* In: 2014 international conference on the internet of things (IOT). IEEE, 2014. p. 79-84..

[8] XIAO, Guangyi et al. *User interoperability with heterogeneous IoT devices through transformation.* IEEE Transactions on Industrial Informatics, v. 10, n. 2, p. 1486-1496, 2014.

[9] WANG, Kai et al. *Enabling collaborative edge computing for software defined vehicular networks.* IEEE Network, v. 32, n. 5, p. 112-117, 2018.

[10] ETSI. *MEC - V2X Information Service API.* Available in: https://forge.etsi.org/rep/mec/gs030-vis-api.

[11] KING, Heather; NOLAN, Keith; KELLY, Mark. *Interoperability between DSRC and LTE for VANETs.* In: 2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES). IEEE, 2018. p. 1-8.

[12] BALTAR, Leonardo Gomes; MUECK, Markus; SABELLA, Dario. *Heterogeneous vehicular communications-multi-standard solutions to enable interoperability.* In: 2018 IEEE Conference on Standards for Communications and Networking (CSCN). IEEE, 2018. p. 1-6.

[13] CASADEMONT, Jordi et al. *Multi-radio v2x communications interoperability through a multi-access edge computing (mec).* In: 2020 22nd International Conference on Transparent Optical Networks (ICTON). IEEE, 2020. p. 1-4.

[14] HADIWARDOYO, Seilendria et al. S*mart Highway: An Interoperable V2X Testbed for Connected and Autonomous Mobility Services in Belgium.* In: 27th ITS World Congress, 11-15 October, 2021, Hamburg, Germany. 2021. p. 2080-2091.

[15] JIN, Wen-Quan; KIM, Do-Hyeun. *Implementation and Experiment of CoAP Protocol Based on IoT for Verification of Interoperability.* The journal of the institute of internet, broadcasting and communication, v. 14, n. 4, p. 7-12, 2014.

[16] *RFC 7252 - The Constrained Application Protocol (CoAP).* Available in: https://tools.ietf.org/html/rfc7252.

[17] RATHOD, Digvijaysinh; PATIL, Sunit. *Security analysis of constrained application protocol (CoAP): IoT protocol.* International Journal of Advanced Studies in Computers, Science and Engineering, v. 6, n. 8, p. 37, 2017.

[18] CHEN, Nanxing et al. *Ensuring interoperability for the Internet of Things: Experience with CoAP protocol testing.* Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije, v. 54, n. 4, p. 448-458, 2013.

[19] WANG, Jian et al. *A survey of vehicle to everything (V2X) testing.* Sensors, v. 19, n. 2, p. 334, 2019.

[20] Juniper Networks. *What is multi-access edge computing.* Available in: https://www.juniper.net/us/en/research-topics/what-is-multi-access-edge-computing.html.

[21] ILIEV, Teodor B. et al. *LTE eNB traffic analysis and key techniques towards 5G mobile networks.* In: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2017. p. 497-500.

[22] ETSI Forge. *VIS - publish_v2x_message.* Available in: https://forge.etsi.org/swagger/ui/.

[23] ETSI Forge. *VIS - uu_unicast_provisioning_info.* Available in: https://forge.etsi.org/swagger/ui/.

[24] ILIEV, Teodor B. et al. *LTE eNB traffic analysis and key techniques towards 5G mobile networks.* In: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2017. p. 497-500.