Performance Analysis of two Machine-to-Machine Architecture Types in Vehicular Communications

Paulo Araújo
Centro ALGORITMI
University of Minho
Braga, Portugal
a70015@alunos.uminho.pt

Alexandre Santos, António Costa
Centro ALGORITMI & Dep. Informatics
University of Minho
Braga, Portugal
{alex, costa}@di.uminho.pt

Fábio Goncalves, Bruno Ribeiro
Centro ALGORITMI
University of Minho
Braga, Portugal
{b7207, b7214}@algoritmi.uminho.pt

Vadym Hapanchak, Oscar Gama
Centro ALGORITMI
University of Minho
Braga, Portugal
{b7768, b2583}@algoritmi.uminho.pt

Bruno Dias, Joaquim Macedo
Centro ALGORITMI & Dep. Informatics
University of Minho
Braga, Portugal
{bruno.dias, macedo}@di.uminho.pt

M. João Nicolau
Centro ALGORITMI & Dep. Inf. Systems
University of Minho
Guimaraes, Portugal
joao@dsi.uminho.pt

***Abstract*—A Vehicular Adhoc Network (VANET) is a conceptualisation that can be applied to the communications domain of an Intelligent Transportation System (ITS), where vehicles use On Board Units (OBU) to communicate with other vehicles or the road side infrastructure. The heterogeneity of the interconnected devices, the large number of devices inside and outside the vehicles and different degrees of device density, makes interoperability a challenging problem, making way to many machine-2-machine (M2M) autonomous communications. This paper discusses two different M2M architectures that may be used in ITS context and presents test results obtained from implementations of these M2M architectures on real OBU hardware, in the context of ETSI compliant safety applications.**

The experimental results were taken using two different oneM2M Vehicular Architectures Types in order to analyse their suitability to be used in a platooning car-following use case. For the car-following use case, OBUs interface the vehicle's Controller Area Network (CAN) bus and use Dedicated Short Range Communications (DSRC) for M2M communications to convey dynamic data to enable controlling the vehicle's speed and longitudinal distance. The experiments and corresponding result analysis has shown that the oneM2M architectures are suitable to be used for such car-following application, which requires latencies below $100ms$.

***Index Terms*—Intelligent Transportation Systems; Vehicular Adhoc Networks; Machine-2-Machine Architecture; On Board Unit**

I. Introduction

Vehicular networks are mobile networks that allow spontaneous communication between vehicles, infrastructure nodes or external participants, such as public transport passengers, pedestrians or other entities present in transport networks. Although it is not a new concept, the use of the Internet is still limited to personal use on transport networks (i.e.: navigation applications or streaming services), and therefore it is necessary to develop solutions that allow integration with the vehicles themselves. The Internet of Things (IoT) is a vast concept, currently in exponential growth, that defends the connection between objects of daily use to the Internet, allowing its interaction with other machines or servers and increasing their spectrum of use, appearing as an alternative capable of being used in a vehicle environment. The main goal of this work is to analyse the protocols, use cases and architectures used in the Vehicular Networks [1] and IoT [2], in order to implement applications making use of M2M interactions. Therefore, the standards defined by European Telecommunications Standards Institute (ETSI) for ITS [3] and oneM2M were used as a base to develop an applications with two distinct architectures, using a Application Public Interface (API) REST and the protocols HyperText Transfer Protocol (HTTP) and Constrained Application Protocol (CoAP), and their latencies have been tested using commercial OBU devices in a controlled environment.

The latency times in M2M interactions, either using HTTP or CoAP, should be analysed in order to test if the experimental results fit within the latency requirements defined by ETSI for different types of vehicular applications.

It would also be important to get and analyse experimental latency results for oneM2M standard architectures using Wi-Fi, and also using more efficient technologies for transmitting data in vehicular environments, such as 802.11p, DSRC, and Long-Term Evolution-Vehicular (LTE-V).

II. Related Work

IoT and M2M interactions have been making slow introduction into the market, mainly due to the lack of standards and protocols to guide industry and developers in the creation of products and services in a cohesive way [4]. In vehicular context one may use M2M communications, which, if applied to devices that have the ability to connect to a gateway or to an aggregation point, can create networks of interconnected devices through the Internet [5] and the oneM2M standard

978-1-7281-0270-2/19/$31.00 ©2019 IEEE

architecture [6], and specially those oneM2M architectures for Vehicular domain [7], are certainly a very important approach.

Advanced vehicular applications, related to V2X, such as Platooning [8] and Cooperative Adaptive Cruise Control (CACC) or even fast traffic accident rescue for emergency vehicles [9] may benefit from automated M2M communication standards.

Ploeg et al [10] describe the design and practical validation of a CACC system using a decentralized controller design with wireless communications and showing that the wireless communication link enables driving at small inter-vehicle distances, whereas string stability is guaranteed.

Milanés et al. [11] describe the design, development, implementation and testing of a CACC system, whose system design is based on controllers that determinate the maneuvers [12] in the platoon: the leader vehicle approaching maneuver and the car-following regulation maneuver. The solution aims to reduce significantly the gaps between the vehicles, taking advantage of information exchanged using DSRC [13]–[15] wireless communication. Additionally, the CACC improved the response time and platoon stability, when in comparison to the ACC system, proving that the system may be able to improve traffic flow and capacity.

Another work on CACC [16] showed that it is able to work safely and efficiently, ensuring a vehicle following at a close distance, and that the performance of the longitudinal controller employed for basic operations confirms the platoon's string stability and robustness. Also [17] reported experimental evaluation of vehicle-following control technology conducted using small electric vehicles.

In order to interface the vehicle's control and sensing systems, the industry uses normally the CAN bus standard, characterised by being robust, simple to configure and to have excellent fault tolerance capabilities, defined in the ISO 11898-X set of standards [18]. The ISO 11898-2 standard defines the high speed CAN (the most common standard), with data rates up to 1 Mbit/s with a possible bus length of 40m. Using the Flexible Data Rate frame format one can obtain bit rates higher than 1 Mbit/s and payloads longer than 8 byte per frame. Research works exist on using the CAN bus in order to get information and monitor the vehicle's speed, as e.g Salunkhe et al. [19] and Wagh et al. [20] where the development of vehicle speed controlled driving system for a semi-autonomous vehicle is presented.

This paper also uses an experimental setup that makes use of the speed and distance information, acquired periodically (each $100ms$) via the CAN bus, feeding it to the CACC control process so that the following vehicle is able to automatically adjust its velocity, keeping a safe distance to the front vehicle.

III. Machine-to-Machine in Vehicular Context

When devising an IoT solution for vehicular networks using the oneM2M framework, the architecture and transmission technologies should be considered more important than the choice of the application protocol itself. The oneM2M standard architecture [6] defines the entities - Application Entity (AE), Common Service Entity (CSE), Network Service Entity (NSE), Infrastructure Node (IN) - and also the links between entities. As shown in Figure 1, the Application Entity (AE) is able to communicate with Common Service Entity (CSE), which may or may not coexist in the same physical entity); CSEs are able to establish bidirectional links with other CSEs and also with Network Service Entity (NSE), enabling CSEs to use the services provided by those NSEs.

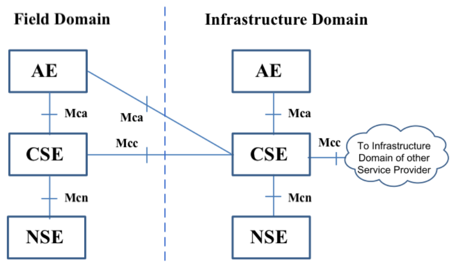

Fig. 1. oneM2M functional Architecture

For deployment in vehicular context, four high level oneM2M architecture types are defined [6], [7]: Vehicular Architecture Type 1 (VD-ARCH1), VD-ARCH2, VD-ARCH3 and VD-ARCH4. From the analysis of these architecture types, we found that architectures VD-ARCH1 and VD-ARCH2 were very similar and also that VD-ARCH3 would not be implementable in the laboratory as available OM2M software did not support the necessary two-way links between CSEs. So, the laboratory hardware setup for the performance analysis addressed implementations of the Vehicular Architectures Type 1 - `VD-ARCH1` - and Vehicular Architectures Type 4 - `VD-ARCH4`. Thus, these two specific oneM2M architectures were selected in order to investigate if the inclusion of RSUs (infrastructure connected nodes) in the communication path between OBUs is adequate to be used in vehicular domain use cases with specific delay constraints. Comparative studies were conducted to assess the performance of software applications deployed using each of those two architecture types, testing real use cases with commercially available vehicular hardware interfacing the vehicle's CAN bus.

A. Two Different Machine-to-Machine Architectures

VD-ARCH1: In the M2M architecture presented in Figure 2, the Application Service Node (ASN) and / or Middle Nodes (MN) are located in the vehicle's gateway, being able to connect to non-oneM2M Node (no-DN) or Application Dedicated Node (ADN) running in internal networks. ADNs and no-DNs are usually applications residing in devices with low energy resources that replicate the data received from entities having greater processing power. These applications would be good for listening the CAN bus or installed on any sensor not connected to the CAN (e.g. with a bluetooth connection, which could be permanently connected to the gateway). The gateway could be a mobile phone (e.g. from the driver), using Bluetooth to connect to other devices or the OBU of the vehicle, which would remain connected to the

CAN bus. This device would still be connected to an external network that would allow it to connect to oneM2M servers.

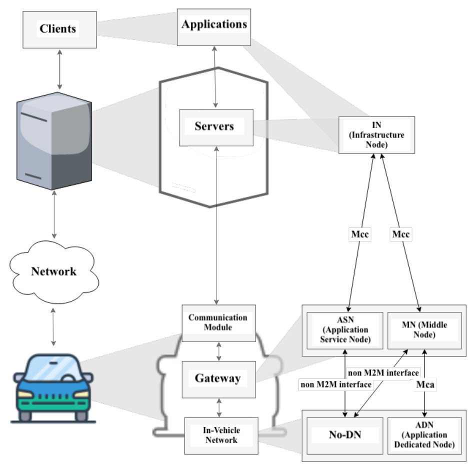

Fig. 2. VD-ARCH1 - Vehicular Architecture Type 1 for M2M cooperative applications

VD-ARCH4: Another M2M architecture is presented in Figure 3. Despite having a communication module in the vehicle, there is no gateway, so there is no MN in each vehicle. Thus, it is up to the RSU to maintain the MN that is connected directly to the internal network of the vehicle, by means of a wireless communication. This type of architecture is especially useful in integrating technology into vehicles that do not have built-in capabilities; it will enable the system to get information from a wider range of vehicles, not limited to those already equipped with OBUs.

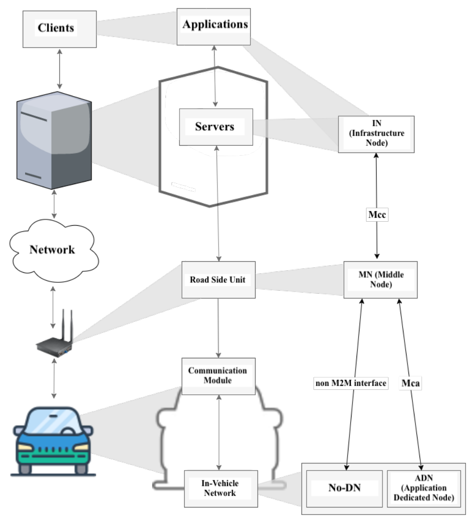

Fig. 3. VD-ARCH4 - Vehicular Architecture Type 4 for M2M cooperative applications

B. Vehicular Platoon as Machine-to-Machine Use Case

An example use case, where the oneM2M architectures may be used, is a platoon control protocol. In fact, a platooning ITS protocol [12] running on top of commercial OBU hardware has been tested by the authors and also a Cooperative Cruise Control velocity controller, running on top of the same hardware, has also been developed and field tested by the authors [16] within the same project; the Cooperative Cruise Control velocity controller is used by the follower vehicle, so that it attempts to match the leader's velocity. The velocity of the leader is transmitted to the followers by means of Cooperative Awareness Message (CAM) standard messages exchanged between leader and follower and has been tested using direct DSRC communications between OBUs, as shown in Figure 4.

These CAM messages are transmitted / published using IEEE 802.11p and may be received either directly by the OBUs or made available / published by RSUs connected to the infrastructure.

It is assumed that direct OBU connections, as depicted in Figure 4, lead to shorter latencies than when the communication path involves using OBU-RSU-OBU connections by means of VD-ARCH1 and VD-ARCH4 which lead to longer delays; so, for the analysis of the delays and maximum latencies, the laboratory tests for performance analysis use the latter (and longer delay) path case for communications, involving OBU-RSU-OBU communications.

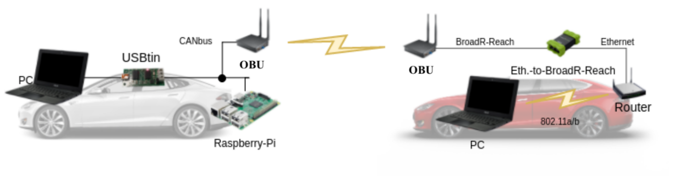

Fig. 4. Car following use case and its experimental testbed

In order to accomplish this, a CAN bus connection is used for collecting and parsing all in-vehicle messages. The OBU is used to generate standard CAM messages so that they can be received by the follower vehicle, allowing it to automatically control its own velocity and present a GUI with the current velocity and also the actual distance between vehicles, as shown in Figure 5 (taken from the prototype implementation). This GUI shows that the acelarator is at level 44, current vehicle speed is $73Km/h$ (motor at $3000RPM$) and the longitudinal distance between cars, leader and follower, is $186m$; the Wi-Fi green symbol means that IEEE 802.11p communication is available (beacons are being received). The follower vehicle adapts and controls its (ego) velocity and distance to the preceding vehicle by using the CACC methodology described in [16] to implement the longitudinal controller.

It should be noticed that vehicles are considered to have a perfect gearless electric motor where the current velocity depends only on the current motor revolutions per minute (rpm), not taking into account the vehicle dynamics or specific road conditions.

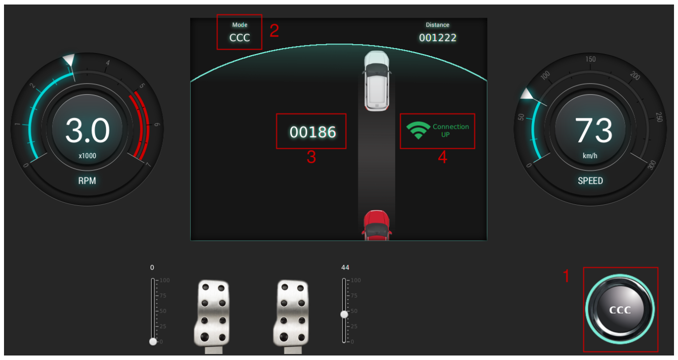

Fig. 5. Graphical UI at follower vehicle

IV. Performance Analysis of Machine-to-Machine in Vehicular Context

In order to assess the performance of the two Machine-to-Machine architectures discussed, based on the oneM2M standards, a set of tests have been carried out and the experimental latencies have been recorded. A target latency of $100ms$ has been used as reference, since it is the lowest value of latency required by ETSI in [21] for the use cases covered by architectures VD-ARCH1 and VD-ARCH4.

A. Testing Latencies in Different M2M Architectures

An experimental laboratorial setup has been established, using real hardware devices and radio communications, in order to determine what are the latencies obtained when M2M interactions among vehicles are accomplished, using several oneM2M standard architectures.

In order to emulate and bring the whole vehicular context into the laboratory environment a common workbench, yet configurable and as realistic as possible, has been established at the laboratory premises. This common laboratorial setup includes hardware devices and connections enabling the access to any CAN bus signals and to OBU nodes; software prototype development platform, eventually with a graphical interface, that interacts and controls the vehicles (via CAN bus, with sensors, actuators and the ability to simulate the movement of vehicles).

Figure 6 presents the common laboratorial setup established for the different M2M experiments, emulating the main vehicle components: the internal CAN bus interface, where in-vehicle messages (from sensors or into actuators) are exchanged and the in-vehicle OBU, which also interfaces the CAN bus, is used for over the air IEEE 802.11p communications using the V2X stack.

As seen in Figure 6 data from (and into) the vehicle control (sensors and actuators) is exchanged via the CAN bus interface and messages are exchanged between vehicles using IEEE 802.11p.

The internal network, identified in the standard, was implemented with the Common Setup (box on the left in Figure 7) connected to the OBU (no-DN) through the CAN interface; the gateway was developed with a MN settled at a platform

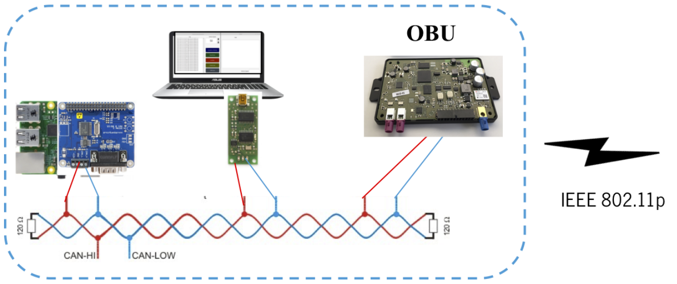

Fig. 6. Common Setup for M2M architectures testing, showing CAN bus and IEEE 802.11p interfaces

(PC or tablet) connected to the OBU through any local network (BroadR-Reach Ethernet, IEEE P802.3bw was used). This MN has also a Wi-Fi connection for Internet access to the server (IN-CSE), where all the data received from all nodes (via Mcc or Mca interfaces) is kept and made available.

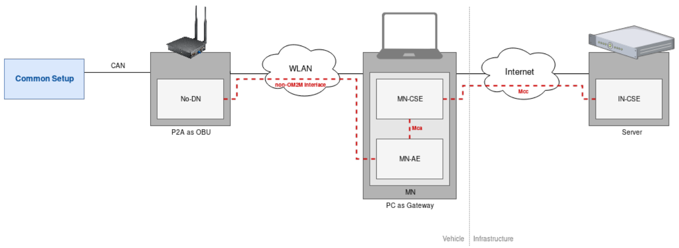

Fig. 7. Testbed for M2M interactions with VD-ARCH1

In order to monitor the server (IN-CSE) node and to be able to access its current status, an Internet web access has been settled, able to present all the data received and made available at the IN-CSE server.

The function of the gateway is to establish the link between the internal network of the vehicle and the external IN-CSE server. Although the oneM2M standard allows the gateway to implement CSE functions of type MN or ASN, MN implementation was chosen because it is more versatile as this allows connections to either MN or ASN nodes (being that ASN node is always an end node); this architecture makes possible connecting other nodes, from other different components within the vehicle.

As shown in Figure 7 the MN contains an application entity that binds to the MN-CSE through a Mca link and, simultaneously, also establishes a link (non oneM2M) to the internal network of the vehicle. This connection is accomplished by means of Simple Network Management Protocol (SNMP) requests sent to the vehicle's OBU (where a Management Information Base (MIB) has been implemented and an SNMP agent listen to requests). The service updates the MIB every $50ms$ with the current speed values received via CAN interface. The monitoring application was developed in Java and communicates with the MN-CSE through REST requests using Hypertext Transfer Protocol, as defined by oneM2M; externally, it is connected to the OBU through the implementation of a simple SNMP client accessing specific OID values, identifying the vehicle speed, within the MIB.

This testing configuration enables the evaluation of the time latency of this `VD-ARCH1` M2M architecture, and HTTP, when the leading vehicle accesses its CAN to get the ego speed value and the follower vehicle accesses this (leader) speed value by means of IEEE 802.11p messages.

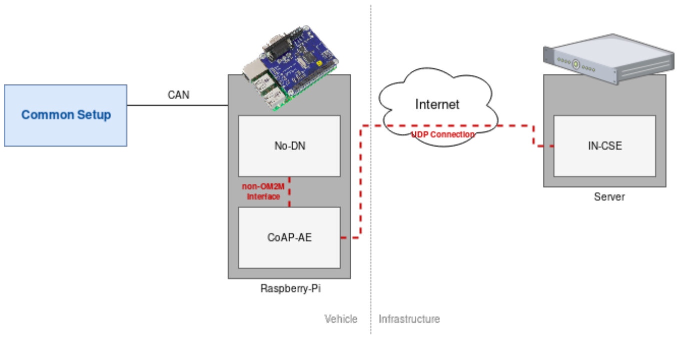

Fig. 8. Testbed for M2M interactions with VD-ARCH4

In order to test the performance of the `VD-ARCH4` another setup has been established, as seen in Figure 8. To access the CAN bus (and the data corresponding to ego-vehicle sensor and actuator devices) a PI-CAN hardware module, connected to a Raspberry-PI, was used and a Constrained Application Protocol based application has been added to the associated no-DN node; together, these elements then act as the node that accesses directly the internal network of the vehicle. This testing setup uses CoAP, instead of the HTTP based application, on account of the reduced capabilities of the Raspberry-PI hardware platform, enabling also the evaluation of the influence of the CoAP protocol (slightly lighter than HTTP, using UDP instead of TCP). The CoAP client application was also developed in Java. The leader speed values are updated and sent at a frequency of 10 Hz.

B. *Performance Analysis*

The two monitoring applications presented in section IV-A were developed in order to measure latency, one using HTTP and another using CoAP. The monitoring application is named IN-MONITOR and subscribes, through the use of a specific request to the server, the `Container Speed` values. Thus, whenever a `Content Instance` is created under the `Container Speed` in IN-CSE, a notification message is automatically forwarded to the IN-MONITOR application. Under normal conditions, the IN-MONITOR application would be running on a different device, running in a different computer or mobile device. However, in order to derive the M2M latency with high accuracy the devices' clocks should be fully synchronized[1]; so, as detailed below, the experimental results were taken having the IN-MONITOR application installed on the same device where the MN remains active; nevertheless, all the M2M communication and computation procedures are done independently by isolated processes.

[1]Outside laboratory environment, clock synchronization between nodes could be achieved by accessing the GPS signal in different devices; other methods (e.g. NTP synchronization) were considered inaccurate

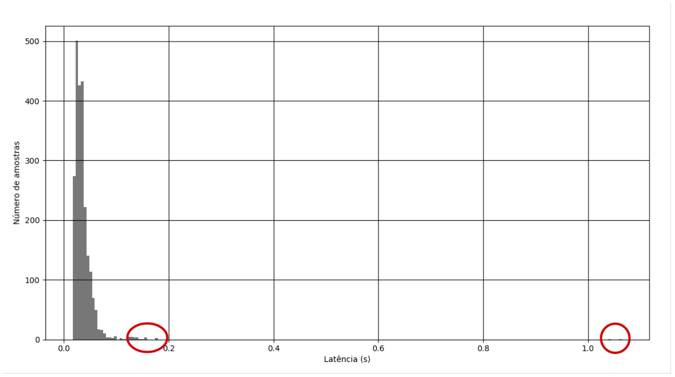

Fig. 9. VD-ARCH1 - Raw experimental latency results (outliers analysis)

The experimental setup enabled us to have several runs, capturing a large number of samples so that one could get more than 2000 consecutive samples for each architecture, leading to two data sets with 2312 samples each, that have been used for performance analysis; these samples were collected over different periods of time, always using the same hardware setup and always in the same laboratory environment.

The data set was produced by the IN-MONITOR applications, already referred, and were organised with a generic format and saved into a database file; the data set has been subsequently used to perform the statistical analysis of all the data collected, in order to obtain values such as average, median, standard deviation and 95% confidence interval, among others.

The graphic in Figure 9 presents the raw results obtained with `VD-ARCH1` architecture in the form of an histogram of the latency results, with a huge concentration of values falling into the interval $[0.0s \, .. \, 0.1s]$, although one may notice the existence of some values at a fairly large distant from this initial concentration, which later were proved to be outliers. The distribution is clearly asymmetrical, it is not a normal distribution, and so the characteristic statistical values for the distribution of all raw values have been computed, resulting in an average of $raw.average = 36.3ms$ with standard deviation of $raw.std = 34.0ms$ and having $23.0ms$ as the corresponding modal value.

The statistical significative values of the distribution, after outliers removal (for the 95% confidence interval, outliers removed if out of $[raw.average \pm 1.96 * raw.std]$) are presented in Figure 10 leading to a latency $average = 34.3ms$, with standard deviation $std = 22.0ms$; the analysis of the data set also showed that 50% of the samples present a latency in the interval $\in [25.0ms, 40.0ms]$.

For a complementary statistical analysis, a *Box and Whisker Plot* is presented in Figure 12 (lower plot). As shown, the minimum latency value obtained was $17.0ms$, with the four quartile values being: $25.0ms$ for the first quartile, $(40.0ms$ for the second and $40.0ms$ for the third quartile, with a

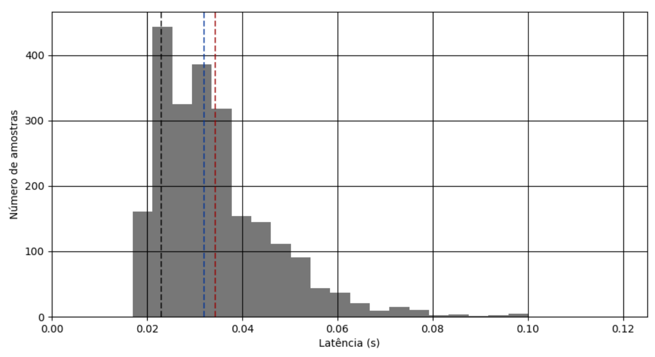

Fig. 10. VD-ARCH1 - Experimental latency results (outliers removed): $average = 34.3ms$, $std = 22.0ms$, 50% of the samples have latency $\in [25.0ms, 40.0ms]$

maximum latency of $62.0ms$, all values having an observed latency pretty under the $100ms$ limit.

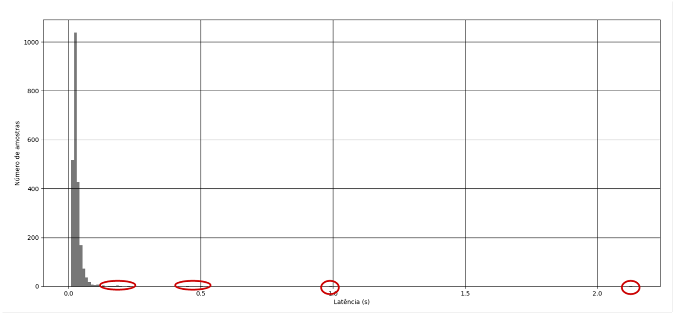

Fig. 11. VD-ARCH4 - Raw experimental latency results (outliers analysis): $average = 33.3ms$, $std = 27.0ms$

Similarly, the histogram for the sampled values of latency registered with the `VD-ARCH4` architecture is shown in Figure 11, which also exhibits an asymmetric distribution where the few outliers are identified by a circle along the X axis.

Values for the latency average, its standard deviation (as well as its median and modal values) for this asymmetric distribution have also been computed, leading to identify a positive asymmetric distribution having an average latency $raw.average = 33.3ms$, with standard deviation $raw.std = 27.0ms$. Again, outliers have been removed for the 95% confidence interval (outliers removed if out of $[raw.average \pm 1.96 * raw.std]$ interval) and the resulting values get slightly lower, with $average = 31.0ms$ and standard deviation down to $std = 14.2ms$.

The corresponding *Box and Whisker Plot* is also presented in Figure 12 (upper plot), being that the minimum latency value obtained was $11.0ms$, with the four quartile values being: $22.0ms$ for the first quartile, ($27.0ms$ for the second and $35.0ms$ for the third quartile, with a maximum latency of $54.0ms$. These results from `VD-ARCH4` architecture are slightly better than those obtained with `VD-ARCH1` architecture, confirming that all statistically significant values have latencies pretty under the $100ms$ limit.

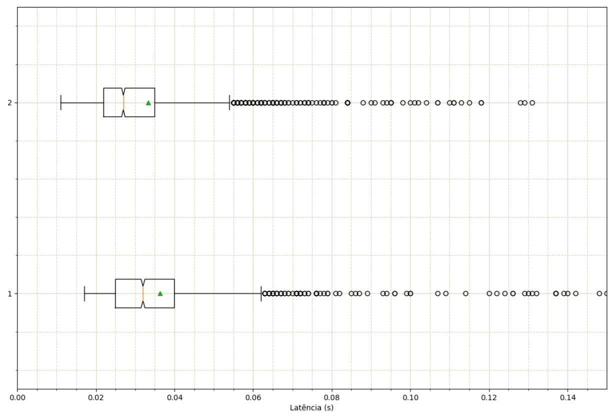

Fig. 12. Comparing VD-ARCH1 (lower plot) and VD-ARCH4 (upper plot) architectures - Box and Whiskers Plot results

Results bring up a tendency for the `VD-ARCH4` architecture to exhibit samples with lower latencies. This result is somehow as expected on account of the CoAP protocol, that is UDP based and presents also a header with lower overhead, which, consequently, also decreases the time for packet transmission; the HTTP protocol, with the corresponding TCP headers, has been used in the `VD-ARCH1` architecture, which has a bigger overhead and uses more resources, both at processing and transmission levels.

It may be stated that both M2M architectures studied, `VD-ARCH1` and `VD-ARCH4`, fit the expected latency requirements under the $100ms$ target, with a slight advantage presented by `VD-ARCH4` architecture, that exhibits smaller latency values, either at average and median values exhibiting also smaller dispersion of experimentally collected values.

V. Conclusions and Future Work

The adoption of M2M communication technologies for the implementation of intelligent solutions in day-to-day commercial products, in general known as Internet of Things, IoT, has undergone a big and very sloppy increase over the last few years. The introduction of IoT and M2M paradigms in the vehicular field, where successful end-user products have already been established, can be considered as an added value for all those involved in the modern world, whether they are owners of vehicles, passengers or even component suppliers. The work developed in this project, aims to contribute to a better understanding of the use of this technology within the universe of vehicular networks, addressing not only current standards but also analysing commercial implementations and testing of concrete solutions. Thus, vehicular application use cases and Machine-to-Machine architectures were implemented according to the standards oneM2M and tested with real hardware for vehicular On Board Units the objective in order to evaluate its performance and analyse if the latency obtained in laboratory experiments would be compatible with the use case application requirements.

In order to be able to establish and test some real laboratory scenarios this work has developed and established a hardware platform, referred as Common Setup for M2M, able to emulate the behaviour of a vehicle, including access to its internal CAN bus and establishing a simple vehicle dynamics model. Using this experimental setup it was possible to perform a consistent battery of tests aimed to evaluate the performance of two oneM2M architectures. Test results were entered into an experimental data set, analysed afterwards to get its main statistically significant characteristics related to message exchange concerning the vehicles' velocity, enabling the comparison between the two oneM2M architectures and also to analyse if the latencies were within the values required by ITS standard requirements.

Thus, after the tests performed, it was observed that both of the oneM2M architectures tested, `VD-ARCH1` and `VD-ARCH4`, using either HTTP or CoAP protocols, presented latency values well under the $100ms$ limit, as required by V2X applications defined by ETSI, even for safety use cases; the conditions under which the tests were carried out were not ideal and even better results are to be expected when using DSRC with outdoor antennas.

Comparing the architectures with each other, it is possible to perceive the superior performance of the `VD-ARCH4` architecture in relation to `VD-ARCH1` architecture, with respect to the latency values; however, as both architectures do meet the expected $100ms$ latency limit, the focus may shift to other parameters such as reliability or better adaptation to the environment. Depending on the use cases, other architecture's characteristics, such as scalability or even of the resources needed, may be of greater relevance.

The study of the performance of two other oneM2M architectures is planned, as well as extending the experiments into road field tests, where besides using real OBU hardware one could also use real vehicles in the machine-to-machine loop; this would be extremely useful, making possible to calibrate the behaviour of the Common Setup so that differences between laboratory and road experiments could be reduced. Also planned as future work is to repeat the performance evaluation with other technologies - such as LTE, LTE-V and even 5G - testing these oneM2M architectures in a similar way.

It would also be interesting to create a proxy for the use of 802.11p for oneM2M data transmissions in these architectures. This extension would allow the use of both OBUs and RSUs as Middle Nodes, without the need for intermediate gateways, increasing the scalability and reachability radius of the system.

Acknowledgment

This work has been sponsored by the Portugal Incentive System for Research and Technological Development. Project in co-promotion 002797/2015 (INNOVCAR 2015-2018), and also by COMPETE: POCI-01-0145-FEDER-007043 and FCT - Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2019.

References

[1] M. Gerla and L. Kleinrock, "Vehicular networks and the future of the mobile internet," *Computer Networks*, vol. 55, no. 2, pp. 457–469, 2011.

[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb 2014.

[3] *ETSI EN 302 665 V1.1.1 (2010-09): Intelligent Transport Systems (ITS); Communications Architecture*, ETSI Std., 2010.

[4] A. Whitmore, A. Agarwal, and L. Da Xu, "The Internet of Things - A survey of topics and trends," *Information Systems Frontiers*, vol. 17, no. 2, pp. 261–274, Apr 2015. [Online]. Available: https://doi.org/10.1007/s10796-014-9489-2

[5] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and N. D. Johnson, "M2M: from mobile to embedded Internet," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 36–43, 2011.

[6] *ETSI TS 118 101 V2.10.0 (2016-10): oneM2M; Functional Architecture*, ETSI Std., 2016.

[7] oneM2M, "ETSI TR-0026 V4.2.0 (2018-07): Vehicular Domain Enablement," ETSI, Tech. Rep., 2018.

[8] M. Segata, F. Dressler, R. Lo Cigno, and M. Gerla, "A simulation tool for automated platooning in mixed highway scenarios," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 389–392.

[9] J. C. Nobre, A. M. de Souza, D. Rosario, C. Both, L. A. Villas, E. Cerqueira, T. Braun, and M. Gerla, "Vehicular software-defined networking and fog computing: Integration and design principles," *Ad Hoc Networks*, vol. 82, pp. 172 – 181, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870518305080

[10] J. Ploeg, B. T. M. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," in *14th International IEEE Conference on Intelligent Transportation Systems, ITSC 2011, Washington, DC, USA, October 5-7, 2011*. IEEE, 2011, pp. 260–265.

[11] V. Milanés, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Trans. Intelligent Transportation Systems*, vol. 15, no. 1, pp. 296–305, 2014.

[12] B. Ribeiro, F. Gonçalves, A. Santos, M. J. Nicolau, B. Dias, J. Macedo, and A. Costa, "Simulation and testing of a platooning management protocol implementation," in *Wired/Wireless Internet Communications: 15th IFIP WG 6.2 International Conference, WWIC 2017, St. Petersburg, Russia, June 21–23, 2017*, Y. e. a. Koucheryavy, Ed. Springer International Publishing, 2017, pp. 174–185.

[13] *CEN EN 12834: Road transport and traffic telematics - Dedicated Short Range Communication (DSRC) - DSRC application layer*, European Committee for Standardization Std., 2003.

[14] *CEN EN 12795: Road transport and traffic telematics - Dedicated Short Range, Communication (DSRC) - DSRC data link layer: medium access and logical link control*, European Committee for Standardization Std., 2003.

[15] *CEN EN 12253: Road transport and traffic telematics - Dedicated Short Range, Communication (DSRC) - Physical layer using microwave at 5,8 GHz*, European Committee for Standardization Std., 2004.

[16] V. Hapanchak, A. Costa, J. Macedo, A. Santos, B. Dias, M. J. Nicolau, B. Ribeiro, F. Gonçalves, Ó. Gama, and P. Araújo, "Simulation and testing of a platooning cooperative longitudinal controller," in *Proc. 2nd EAI International Conference on Intelligent Transport Systems, INTSYS' 2018, Guimaraes, Portugal, November 21-23, 2018*. Springer, 2018.

[17] T. Ogitsu and M. Omae, "Design and experimental testing of vehicle-following control for small electric vehicles with communication," in *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, Feb 2015, pp. 586–590.

[18] I. S. Organization, *ISO 11898-1:2015 - Road vehicles – Controller Area Network (CAN) – Part 1: Data link layer and physical signalling*, ISO Std., 2015.

[19] A. A. Salunkhe, P. P. Kamble, and R. Jadhav, "Design and implementation of CAN bus protocol for monitoring vehicle parameters," in *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology*, May 2016, pp. 301–304.

[20] P. A. Wagh, R. R. Pawar, and S. L. Nalbalwar, "Vehicle speed control and safety prototype using controller area network," in *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*, June 2017, pp. 1–5.

[21] *ETSI TR 102 638 V1.1.1: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions*, ETSI Std., 2009.