Service Provisioning in Vehicular Networks through Edge and Cloud: an Empirical Analysis

Zakaria Laaroussi*‡, Roberto Morabito*, Tarik Taleb‡§
*Ericsson Research, Jorvas, Finland.
‡Dep. of Communications and Networking, Aalto University, Espoo, Finland.
§Oulu University, Oulu, Finland.
Email: *firstname.lastname@ericsson.com, ‡firstname.lastname@aalto.fi

***Abstract*—Vehicular Cloud Computing is a network infrastructure paradigm that has been largely used in the vehicular systems landscape for improving drivers' experience. In particular, the higher computational resources made available by cloud computing technologies have helped in coping with the tremendous growth of data traffic exchanged within vehicular networks. However, the advanced development of such infrastructure, together with the relentless proliferation of services and applications characterized by heterogeneous and demanding requirements, has led to redefine the way in which cellular-based vehicular networks assist vehicular communications. As an example, Multi-Access Edge Computing (MEC) is an emerging network paradigm that can be exploited also in vehicular scenarios to foster a more effective and flexible service delivery. Although in literature the migration of vehicular systems towards a MEC-based approach has been already envisaged giving rise to the concept of Vehicular Edge Computing, a not fully investigated aspect is represented by the lack of experimental insights that shed light on the actual feasibility of this emerging network infrastructures. In this paper, we try to fill the gap in this respect by presenting an extensive empirical analysis performed through a vehicular system testbed. In particular, our work aims at providing empirical insights on the advantages that an edge cloud-based service provisioning can enable in comparison to a centralized cloud-based approach. Besides, by focusing only on the transmission of small-sized workload—i.e. with payload comparable to the one produced by In-Vehicle's sensors—this work also aims at evaluating the suitability of different application layer protocols (HTTP, CoAP, and MQTT) in this peculiar context. In the performance analysis, additional aspects have been also considered, including the impact of vehicle's speed as well as scalability issues.**

I. Introduction

Vehicular cruising has never been easier, enjoyable and safer than today, thanks to the possibility of connecting vehicles to the Internet. In fact, nowadays vehicles have the possibility to exploit a high amount of diversified services, besides the possibility to interact with a set of heterogeneous entities that vary from other vehicles, transportation and network infrastructures, pedestrian, etc. [2]. In this respect, the European Telecommunications Standards Institute (ETSI) has defined what are the services required to build up an Intelligent Transportation System (ITS), together with a set of requirements that such services need to satisfy [1]. In such context, it becomes clear how the connectivity becomes a key aspect for enabling smart and highly-performing transportation systems. With specific reference to the automotive context, a key ITS enabler is represented by the network communication, as it enables vehicles to interact among themselves as well as with other entities. Those interactions can occur for different purposes such as data exchange and information sharing with the surrounding connected world. In the majority of cases, the communication relies on cellular connectivity using 3rd Generation Partnership Project (3GPP) technologies. Alternatively, non-3GPP technologies (e.g., Wi-Fi) can be also employed. Another key aspect that must be constantly considered in the automotive context is the possibility to ensure an effective service provisioning. In particular, guaranteeing low-latency communication can become crucial, especially for services that need to satisfy demanding performance requirements. In this regard, the possibility of placing the service provisioning at the network edge, by exploiting the Multi-Access Edge Computing (MEC) concept also in vehicular scenarios, can be complementary to what the Vehicular Cloud Computing is already capable to deliver [13, 11]. In this work, our main goal is to empirically study—through the setup of a vehicular system testbed—whether a service provisioning issued at the network edge can generate performance benefits in comparison to service provisioning from centralized cloud. We focus on vehicle communications enabled by 4G network, by also characterizing the performance of several application layer protocols and taking into account further factors that might influence the performance (e.g., vehicle's speed and scalability).

The main contributions of this paper can be summarized as follows:

- We design and implement a testbed, which allows us to faithfully reproduce a vehicular communication system, by following the recommendations made by main standardization entities.
- For the particular case of small-size workload transmission, we empirically demonstrate the performance difference achieved by an edge-based service provisioning when compared to a cloud-based service provisioning.
- Taking into account a wider availability of application layer protocols, we extensively evaluate, through several field tests, suitability and performance trade-off of three most-widely used protocols, namely MQTT, CoAP, and HTTP.

To the best of our knowledge, this is the very first empirical contribution that, through the implementation of a vehicular system testbed, tries to shed light on the suitability of multiple protocols in this specific context, by considering also possible compromises due to different service provisioning setup (edge vs. cloud). The remainder of this paper is organized as follows. Section II contextualizes our area of investigation and thoroughly describes what is the scenario taken into account in our work. In Section III, we provide a detailed description of the testbed implementation. Section IV presents the result of our extensive performance evaluation. Section V browses the existing scientific literature in two respects: application layer protocol comparison and emerging edge-based vehicular systems. Finally, conclusions and future work are drawn in Section VI.

II. Research Context

A. Vehicle Service Provisioning

The automotive scenario has dramatically changed in the last decades. In past years, the driver experience was mainly limited to the vehicle's body design, engine, and ergonomic, without including any communication with the outside world. Differently, nowadays vehicles take part of more sophisticated ecosystems, by enabling therefore an enhanced quality of experience. This clear shift has led to the definition of a new ecosystem known as Internet of vehicles (IoV), in which the communication between vehicles and other entities enable a fully-connected ecosystem scaling along with the growing number of vehicles [6]. The concept of IoV has been matter of common interest in many standardization organizations such as ETSI and 3GPP [8]. A functional IoV demands the availability of various communication technologies, in order to give to the vehicles the possibility of exchanging information with other IoV components (e.g., data center, edge entities, other vehicles, service providers, and pedestrians). Enabling such interconnection between those distinct components spawned the concept of Vehicle-to-Everything (V2X) communication, which is nowadays considered as a key enabler concept for the deployment of effective IoV and automotive ITS. Clearly, the increasing complexity of vehicular system infrastructures becomes more challenging if we consider the big amount of data traffic generated within it. In fact, nowadays connected vehicles require a continuous stream of data for benefiting of a varied set of services. Within the vehicles, data are elaborated by means of sophisticated On-Board Unit (OBU) and data generator applications are usually characterized by distinct requirements and the quality of service that must be guaranteed varies from case to case. As an example, for the execution of a critical application, it is of vital importance to ensure a seamless provisioning and low-latency communication. In this respect, the IoV shall bear also a performing and scalable service provisioning in such a way to satisfy the aforementioned requirements.

B. Scenario

The vehicular scenario that we are considering is an implementation of one ITS service defined by ETSI [3]. In particular, we are referring to the specific use case of software update over the air (SoTA) [4]. In this specific use case, Vehicle to Cloud (V2C) communications are usually exploited in order to enable the update of services that are realistically running in different OBUs of a vehicle. The possible updates can include e.g. security patches, service features update, Electronics Control Units (ECUs) updates, etc. In such scenario, data centers are obviously playing a crucial role in hosting services issued by the different providers and shall support all V2C interactions expected in different use cases. However, the emerging concept of Multi-Access Edge Computing is leading to a change in the way vehicular network infrastructures are designed. As a consequence of this, we deliberately revise the ETSI scenario by considering two different approaches for issuing the service provisioning: (i) edge-based and (ii) cloud-based. In the first case, the service provider interacts with a vehicle (or vice versa) through a MEC server placed between base station and data center and therefore, closer to the vehicle. In the second case, the vehicle interacts directly with a remote data-center through cellular network base stations. In both cases, 4G connection is used for the communication vehicle-to-edge and vehicle-to-data center. Fig. 1 depicts the aforementioned scenario. The sub-system is comprised of a vehicle, a base station, and a MEC server, defining the so-called *vehicular edge computing area*.

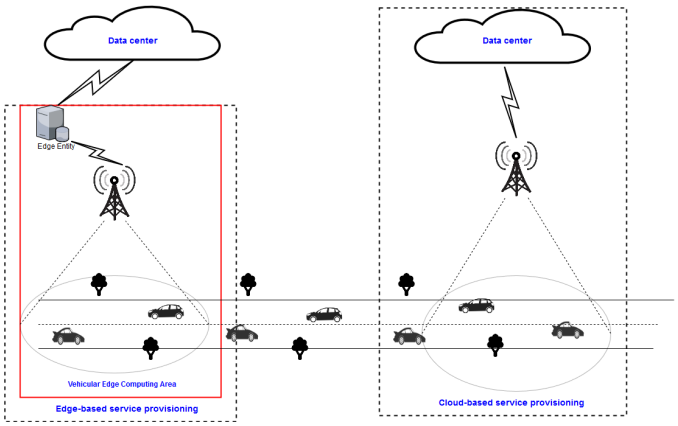

Fig. 1: Analyzed scenario.

III. Methodology and Experimental Setup

This section introduces the methodology and experimental setup adopted for performing the empirical investigation. More specifically, it highlights the implementation choices for the setup of a vehicular system testbed, from both hardware and software perspectives.

A. Testbed setup

In order to reproduce the scenario depicted in Fig. 1 in a real environment, we set up a network infrastructure facility that is distributed across two different sites. Fig.2 shows the entire testbed setup. In this setup, it is possible to identify two main domains: a *data center area* and a *vehicular edge computing area*. The most significant features of the two areas

are outlined below, together with the environment setup of the additional vehicle's On-Board Unit (OBU) at our disposal.

Data center area – the data center facility used for our work is located in Lund (Sweden), located 850 km from the vehicular edge computing area (Jorvas, Finland). In the data center, we set up an OpenStack environment in which a virtual instance is running and makes available the necessary tools (e.g., application layer protocols libraries) for enabling the vehicle-to-cloud communication.

Vehicular edge computing area – the vehicular edge computing area is deployed in Jorvas (Finland), nearby Ericsson Finland facilities. The core element of the VEC area is the edge entity, which consists of a server machine with the following characteristics: computer model Dell Precision T5500, with Intel Xeon X5560 processor (8M Cache, 2.80 GHz, 4 physical cores, 8 threads), 12 GB memory (3x4 GB – 1333 MHz DDR3), and a 10 Gbps Network Interface Card connected to the Internet. The edge entity is deliberately deployed close to the area where the vehicle is cruising, and can be considered as the equivalent of a Road Side Unit (RSU), although characterized by higher computational capabilities. The edge server is interfacing with the base station through the mobile network of a Finnish operator. This also implies that there is no local breakout between the base station and the edge server, fully relying on the network setup provided by the mobile network operator.

In-Car OBU – although nowadays most vehicles can feature very sophisticated OBU that are designed in order to provide a set of multi-purpose services, we decided to set up an auxiliary OBU whose sole purpose was to interact either with the data center or the edge entity. The supplementary OBU is deployed in a general-purpose board, such as Raspberry Pi 3 (RPi3[1]). Connectivity is provided by a system combining a Sixfab base shield[2] and a Quectel EC25 Mini PCIe 4G/LTE module[3]. Both boards are in turn connected to the RPi3 through GPIO interface.

It must be clarified that both the OpenStack instance running in the data center and the edge entity are characterized by the same software environment. The same underlying operating system is used, as well as the libraries utilized for enabling the connection between In-car OBU, edge entity, and data center.

B. Application layer protocols.

As already mentioned in Section II-B, the availability of different application layer protocols has increased the possibility of choosing what protocol needs to be used as an enabler of the service provisioning tasks. Obviously, the choice of one protocol over another highly depends on the particular analyzed use case and on how service providers choose to handle the workload generated with the use case itself. The main scope of the following analysis is an empirical characterization of three of the most widespread application protocols in the vehicular context, which introduces several constraints mainly due to aspects relating to mobility. In our analysis, we focus on data exchange and communication occurring between vehicles and remote entities (e.g., deployed in the edge and/or in the cloud), in which small sized payloads (e.g., hundreds of bytes) are OTA transmitted. The reproduced scenario is analogous for all the application protocols under evaluation. That is, given an equal payload, we estimate what are the average values of throughput and latency during the communication between the interacting entities. Furthermore, we want to understand whether complementary factors (e.g., vehicle's speed) can tangibly affect the performance. It is worth emphasizing that being aware of the design and implementation differences between the different evaluated protocols, the aim of our performance study is not to determine which protocol has the upper hand. Rather, our work has the ambition to provide different empirical insights to the whole research community regarding the effectiveness of such protocols in the vehicular context.

MQTT – Mosquitto[4] is an open source implementation of the message broker that implements the MQTT protocol. In our scenario, the MQTT broker runs either in the edge entity or in the data center. In order to estimate the performance during the communication, we take advantage of a benchmark tool[5] that allows publishing messages to a MQTT broker. Through the benchmark tool, several protocol parameters can be customized (e.g., message size, number of clients connected to the broker, and QoS). The MQTT messages are published by the In-Car OBU where the broker is running (i.e. edge server or data center).

HTTP – the Apache HTTP Server Project[6] is one among the most frequently used open source implementation of an HTTP (Web) server. The evaluation of the HTTP server is done by means of the benchmarking tool ab[7]. In our testbed, the HTTP server operates either in the edge entity or in the data center. It serves the requests issued from the In-Car OBU.

COAP – we use a C implementation of CoAP called libcoap[8] for instantiating CoAP instances. The library allows to easily run both coap-client and coap-server instances. The first one simply interacts with different resources on a remote server, while the second one is a basic server application that provides different CoAP server features. Similarly to MQTT and HTTP, CoAP clients run either in the edge entity or in the data center and interact with a CoAP server entity that is executed in the In-Car OBU. The benchmarking tool CoAPBench[9] has been employed for sending CoAP GET requests towards the target CoAP server.

IV. Measurements Results and Analysis

In this section, we present the results of our empirical analysis. It is organized into different subsections according to separate case studies. It is worth emphasizing that for each case study, we perform the comparison between edge and cloud. With the aim of improving the readability of the paper, we

[1]https://www.raspberrypi.org/
[2]http://sixfab.com/product/raspberry-pi-3g-4glte-base-shield-v2/
[3]http://sixfab.com/product/quectel-ec25-mini-pcle-4glte-module/
[4]https://mosquitto.org/
[5]https://github.com/krylovsk/mqtt-benchmark
[6]https://httpd.apache.org/
[7]https://httpd.apache.org/docs/2.4/programs/ab.html
[8]https://libcoap.net/
[9]https://www.eclipse.org/californium/

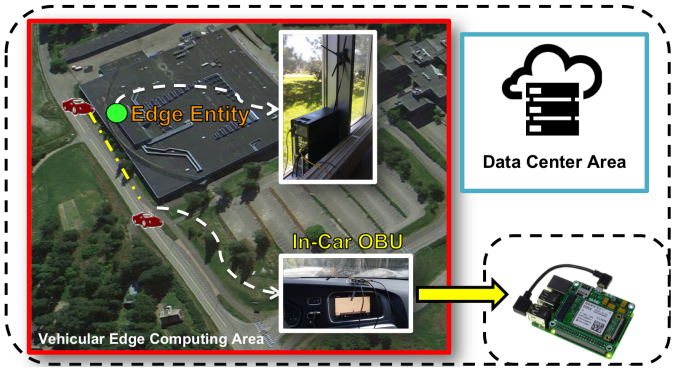

Fig. 2: Testbed setup.

report the most significant results achieved during our analysis. Each measurement is repeated at least 5 times and the results show the average value and related standard deviation.

Impact of Vehicle's speed. The main goal of this evaluation is to assess the impact of vehicle's speed on the performance in terms of throughput and latency for each of the analyzed protocols. From this outcome, it becomes also possible to quantify the performance variation between edge-based provisioning and cloud-based provisioning. Finally, we aim at identifying the most efficient protocol for each specific case study.

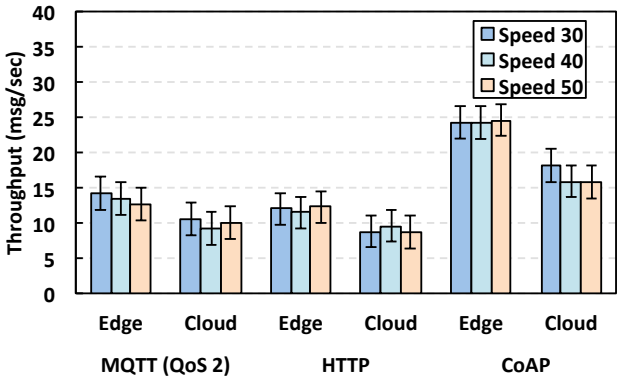

Fig. 3: Throughput variation as a function of vehicle's speed (km/h).

Fig. 3 shows the throughput in terms of the number of messages per second. The main insights that can be derived from this test are three-fold. First, it can be noticed that there is no strict dependence between a vehicle's speed increase and throughput, as the latter remains approximately at the same value. However, it is worth mentioning that the vehicle's speed variation is not very high. Indeed, in the vehicle's cruising area, it was not possible to drive faster because of speed limits. By this analysis, it cannot be determined whether higher speeds could lower the throughput performance. Second, it clearly emerges how CoAP is outperforming both MQTT (with QoS equal to 2) and HTTP. In more details, CoAP throughput is double the HTTP one, while MQTT performs slightly better than HTTP – on average, in the order of 16% better. Finally, it appears clear how the edge-based service provisioning brings tangible advantages when compared to the cloud-based approach. However, this performance gain is not uniform but ranges from approximately 20% (HTTP with vehicle's speed 40 km/h) up to 55% (CoAP with vehicle's speed 50 km/h).

The latency analysis, conducted in Fig. 4, provides similar insights. Effectively, CoAP is largely outperforming both MQTT and HTTP, achieving a latency of approximately 40ms in the case of edge-based provisioning and regardless the vehicle's speed. It is also worth highlighting how a cloud-based provisioning does not worsen the latency performance in a very tangible way (approximately 20%) when compared to the edge case.

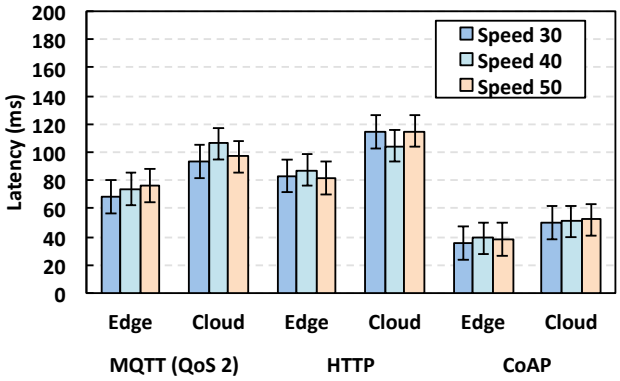

Fig. 4: Latency variation as a function of vehicle's speed (km/h).

As a matter of fact, this percentage difference between edge and cloud remains similar also in the case of MQTT and HTTP. A further insight about this latency evaluation is represented by the similar performance that MQTT and HTTP achieve. However, it must be noticed that we are comparing the case of MQTT with QoS 2, which is supposed to introduce a higher latency delay when compared to alternative QoS setup – part of the following empirical investigation also focuses on the QoS impact in MQTT-enabled communication.

Impact due to number of connected clients – in this second analysis, we are referring to a use case in which the OBU makes available a set of heterogeneous services and applications, which can vary from engine control, driving assistance, customer relationship management (CRM), vendor relationship management (VRM), monitoring and services, to infotainment. The availability of a high number of services implies a potentially high number of interactions between providers and In-Car OBUs. Consequently, from the empirical point of view, we aim to evaluate the impact, still in terms of throughput and latency, produced by a higher number of clients connected to the same In-Car OBU. From this analysis, we can learn how much the throughput of a single client is affected when a higher number of clients are taking part in the communication.

Fig. 5 shows the average throughput per client when one and ten clients connect to the same OBU. As a general rule, we can observe how the average throughput delivered to each client decreases when the number of connected clients grows. However, such a decrease differs from a protocol to another. CoAP remains the most efficient protocol with an approximately 10% throughput decrease. The performance of

MQTT degrades by nearly 15%, while HTTP introduces a tangible throughput reduction which stands around 60%. In addition, the performance degradation does not depend on how the service is being provisioned, i.e, from edge or cloud.

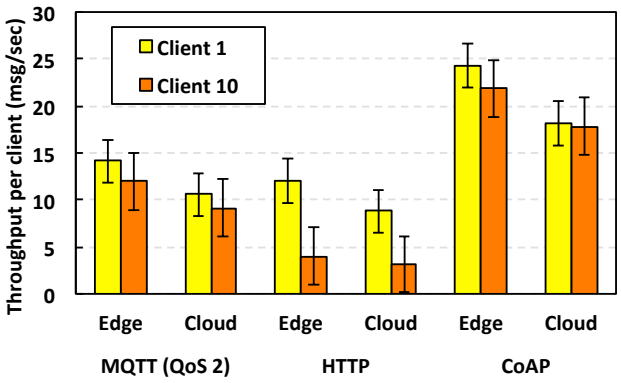

Fig. 5: Throughput variation as a function of the number of clients connected with the In-Car OBU.

From the latency perspective (Fig. 6), all the protocols introduced a performance degradation similar to the one observed for the throughput. In particular, by considering the same number of instances processed per client, each MQTT client requires on average approximately 18% more time to publish the same number of messages, while each HTTP client needs nearly twice the amount of time. Before proceeding to the next analysis, it is worth mentioning that in the aforementioned evaluation, packet loss has not been measured.

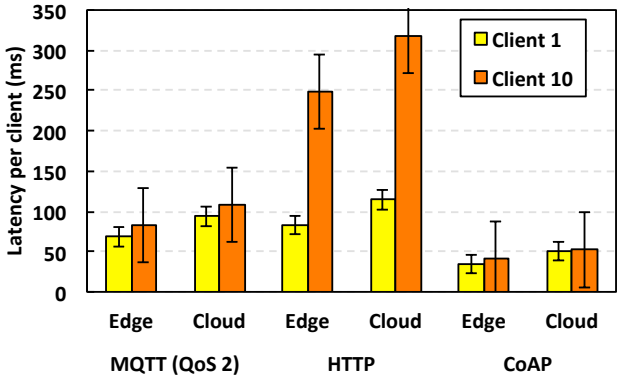

Fig. 6: Latency variation as a function of the number of clients connected with the In-Car OBU.

Impact of QoS in MQTT – the last part of our empirical analysis seeks to analyze how a different value of the MQTT QoS can affect the performance in terms of both throughput and latency. Before discussing the outcome of such evaluation, we briefly describe the main differences between the two values of QoS that we considered in our work. The lowest possible QoS that can be set (i.e. 0) was not considered as it only guarantees a best effort delivery. That is, a message is not acknowledged by the receiver or stored and redelivered by the sender. The alternative values of QoS (i.e. 1 and 2) ensure more reliability in terms of message delivery, somehow in a similar way to how HTTP and CoAP ensure it – although it must also be made clear that the three protocols are designed in a very different way. The main difference among the aforementioned setup is that a QoS set to 1 guarantees that the transmitted message will be delivered once or more than once to the receiver. Differently, with QoS 2, each message is received only once by the receiver [7].

That being said, we previously mentioned how a higher reliability is paid at the expenses of lower performance. We want to quantify how much this performance trade-off is and accordingly draw conclusions. There are two main empirical insights we are interested in observing. First, we want to estimate how throughput and latency are affected by different values of QoS on a base scenario with only one client. Second, we want to observe whether a higher number of clients (i.e. 10) further influences these performance metrics. In the following, we only report the results related to the cloud-based service provisioning. In the case of edge-based provisioning, the same empirical conclusions are valid.

Fig. 7 shows the comparison of the average throughput per client for the two QoS values under analysis. The usage of a higher QoS produces a throughput reduction in the order of 40%. This result remains comparable regardless of the number of connected clients. With regard to the latency performance (Fig. 8), higher QoS slows down the message transmission by approximately 75%. Such analysis may prove useful especially to services providers to determine a tradeoff between reliability and performance.

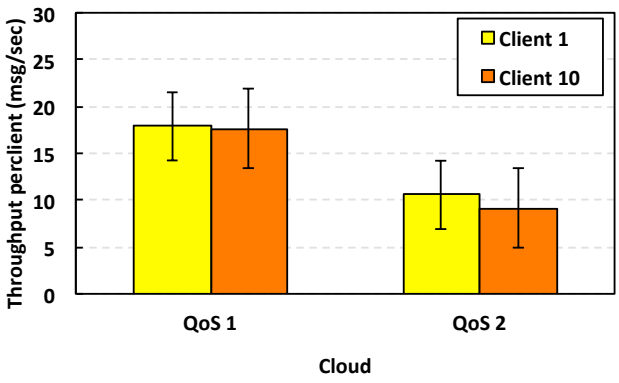

Fig. 7: Throughput variation as a function of two different values for the MQTT QoS.

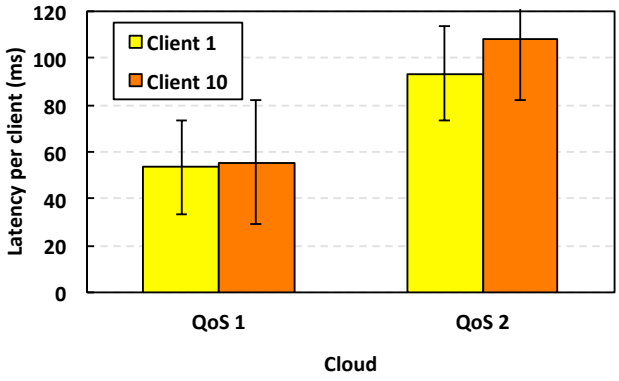

Fig. 8: Latency variation as a function of two different values for the MQTT QoS.

V. Related work

The related work falling within the scope of this work can be categorized into two main areas: (i) application layer protocols comparison and (ii) multi-access edge computing in vehicular networks. Regarding the former, several studies have been focusing on evaluating the performance of different application layer protocols. The authors in [14] deploy a common gateway for exploiting the features of both CoAP and MQTT. The main outcome of the empirical investigation carried out in this paper shows how the performance of CoAP and MQTT – in terms of delay and packet loss – depends on the size of the transmitted message. In [9], Markel et al. perform an empirical study in order to quantify overhead, latency and scalability both for MQTT and CoAP by also considering the different levels of QoS. In [10], CoAP, MQTT and WebSocket efficiency, together with average RTT, are investigated by considering the performance impact generated by the use of different wireless radio technologies. A review of the previous literature indicates no existing empirical study that compares several application layer protocols.

Regarding MEC in vehicular scenarios, and accounting for the remarkable benefits introduced by VEC, an increasing number of works are exploiting such new ecosystems in order to define new systems' optimization and consequently empower the IoV. Jingyun et al. [5] propose a framework named Autonomous Vehicular Edge (AVE). The framework aims to increase the computational capabilities of vehicles at the network edge. This framework includes many functionalities such as job caching, job offloading and sharing of computation resources. In [15], the authors introduced new task offloading mechanisms based on predictions in order to increase the resource computing of vehicles in MEC-IoV scenarios. In [12], the authors offer a description of how MEC can boost the efficiency of vehicular networks, particularly by facilitating the execution of demanding computing tasks at the network edge.

VI. Conclusion

In this paper, we empirically investigated the effects of edge-based and cloud-based service provisioning in vehicular networks. In particular, by harnessing design and implementation of a testbed, we conducted an extensive performance evaluation that aimed at providing experimental insights useful for an optimized design of current vehicular networks. We focused on scenarios in which vehicles interact with the network infrastructure for exchanging small-sized data, comparable for example to the ones produced by the vehicle's sensors. Furthermore, by considering that service providers may exploit a wide set of application layer protocols for connecting vehicles and other network facilities, we have evaluated the efficiency of three of the most widespread protocols for the purpose of better understanding the behavior of such protocols when exploited in such a context.

Acknowledgement

This work is partially funded by the ITEA3 APPSTACLE project (grant no. 15017) and the Academy of Finland's Flagship programme 6Genesis (grant no. 318927).

References

[1] ETSI – Automotive Intelligent Transport Systems. http://www.etsi.org/technologies-clusters/technologies/automotive-intelligent-transport. Accessed: 2018-05-15.

[2] K. Dar, M. Bakhouya, J. Gaber, M. Wack, and P. Lorenz. Wireless communication technologies for its applications [topics in automotive networking]. *IEEE Communications Magazine*, 48(5):156–162, May 2010.

[3] ETSI. Intelligent transport systems (its); vehicular communications; basic set of applications; definitions. Technical report, Tech. Rep. ETSI TR 102 638, 2009.

[4] ETSI. *LTE – Service requirements for V2X services (3GPP TS 22.185 version 14.3.0 Release 14)*, March 2017.

[5] J. Feng, Z. Liu, C. Wu, and Y. Ji. Ave: Autonomous vehicular edge computing framework with aco-based scheduling. *IEEE Transactions on Vehicular Technology*, 66(12):10660–10675, Dec 2017.

[6] M. Gerla, E. Lee, G. Pau, and U. Lee. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 241–246, March 2014.

[7] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. Mqtt-s a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pages 791–798, Jan 2008.

[8] S. Husain, A. Kunz, A. Prasad, K. Samdanis, and J. Song. An overview of standardization efforts for enabling vehicular-to-everything services. In *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 109–114, Sept 2017.

[9] M. Iglesias-Urkia, A. Orive, M. Barcelo, A. Moran, J. Bilbao, and A. Urbieta. Towards a lightweight protocol for industry 4.0: An implementation based benchmark. In *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pages 1–6, May 2017.

[10] S. Mijovic, E. Shehu, and C. Buratti. Comparing application layer protocols for the internet of things via experimentation. In *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 1–5, Sept 2016.

[11] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott. Consolidate iot edge computing with lightweight virtualization. *IEEE Network*, 32(1):102–111, Jan 2018.

[12] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab. Edge computing enabling the internet of things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 603–608, Dec 2015.

[13] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys Tutorials*, 19(3):1657–1681, thirdquarter 2017.

[14] D. Thangavel, X. Ma, A. Valera, H. Tan, and C. K. Tan. Performance evaluation of mqtt and coap via a common middleware. In *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, April 2014.

[15] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. ZHANG. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. *IEEE Vehicular Technology Magazine*, 12(2):36–44, June 2017.