IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 February 2024, accepted 20 March 2024, date of publication 27 March 2024, date of current version 2 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3382001

RESEARCH ARTICLE

An Edge-Based Digital Twin Framework for Connected and Autonomous Vehicles: Design and Evaluation

CLAUDIA CAMPOLO[1,2], (Senior Member, IEEE), GIACOMO GENOVESE[1,2], ANTONELLA MOLINARO[1,2,3], (Senior Member, IEEE), BRUNO PIZZIMENTI[1,2], (Student Member, IEEE), GIUSEPPE RUGGERI[1,2], (Member, IEEE), AND DOMENICO MARIO ZAPPALÀ[1,2], (Student Member, IEEE)

[1]DIIES Department, Mediterranean University of Reggio Calabria, 89124 Reggio Calabria, Italy
[2]Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), 43124 Parma, Italy
[3]Laboratoire des Signaux et Systémes, CentraleSupélec, Université Paris-Saclay, 91190 Gif-sur-Yvette, France

Corresponding author: Claudia Campolo (claudia.campolo@unirc.it)

This work was supported in part by the MOST—Sustainable Mobility National Research Center; in part by European Union (EU) Next-GenerationEU through Piano Nazionale di Ripresa e Resilienza (PNRR)—MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4—D.D. 1033 17/06/2022 under Grant CN00000023; in part by the "Mobility for Passengers as a Service" (MyPasS) Project funded by Italian Government [through the Programma Operativo Nazionale (PON) (2014–2020) Initiative]; and in part by European Union Next-GenerationEU under Grant PNRR DM n. 351.

ABSTRACT Connected and Autonomous Vehicles (CAVs) will be provided with multiple sensing and connectivity options as well as embedded computing and decision-making capabilities. The resulting technological landscape paves the way for the deployment of a plethora of innovative applications involving different stakeholders, such as insurance companies, car repairs, car manufacturers and public authorities. In such a context it is crucial to collect data in an efficient manner, not to burden the vehicle itself and the network infrastructure, while also providing an interoperable data sharing among all the involved players. The Digital Twin (DT) concept can play a key role to properly retrieve, store and share data as well as to exploit them to monitor, predict and improve the vehicle safety and driving experience. This work proposes a comprehensive framework which encompasses the presence of an edge-based DT interacting with the vehicle and the remote applications. It leverages properly specified interfaces and semantic models for different types of data provided by on-board sensing and learning capabilities. A Proof-of-Concept (PoC) has been developed to assess the practicality of the proposal and its performance in terms of communication and computation footprint under a variety of settings.

INDEX TERMS Connected and autonomous vehicles, digital twin, multi-access edge computing, MQTT, OMA-LwM2M.

I. INTRODUCTION

The impressive recent progresses in the Information and Communication Technology (ICT) domain are significantly changing our daily life. Take the concept of vehicle, for instance, which is rapidly progressing towards the idea of a Connected and Autonomous Vehicle (CAV) to enable a safer, more sustainable, comfortable and efficient mobility. This would be possible by retrieving data (e.g., speed, coolant temperature) through the vehicle's Electronic Control Unit (ECU) or acquired via other on-board sensors (e.g., accelerometer, gyroscope, cameras, Light Detection and Ranging (LiDAR)) and by using data as inputs for communication and control decisions. This continuous data stream by feeding the driving assistance system, on the one hand will enable applications of different autonomy levels, according to the SAE J3016 standard [1]. On the other hand, performing autonomous driving tasks by solely relying

The associate editor coordinating the review of this manuscript and approving it for publication was Jonathan Rodriguez.

on on-board sensors has limitations on coverage and/or detection accuracy. Indeed, vehicles can promptly share the aforementioned data with nearby vehicles, pedestrians and infrastructure elements in real-time to improve the perception of the environment and safely coordinate their maneuvers. Different Vehicle-to-Everything (V2X) communication technologies can be used to such a purpose, such as IEEE 802.11p/bd [2], and Fifth Generation (5G)-related connectivity solutions like 5G New Radio (NR) V2X [3] and Sixth Generation (6G) in the near future.

The ambitious CAV's idea can be enabled at a large scale by the synergies of communication, computing, and automation technologies, also encompassing embedded Machine Learning (ML) algorithms, that work together to promptly and reliably share and process the big amount of massively collected data. Indeed, the capability to perceive the environment and perform learning tasks coupled with novel radio interfaces pave the way to a plethora of innovative applications in urban areas and smart cities, ranging from cooperative perception and maneuvering to environmental sensing, from air pollution monitoring to remote diagnostics [4], [5]. Notwithstanding, to achieve the objectives of such ambitious applications, the on-board computing vehicle capabilities need to be complemented by resources available at edge and cloud facilities which can enable the cooperative implementation of computation-heavy tasks, by hiding the complexity of the CAV environment to the applications.

The emerging Digital Twin (DT) paradigm [6] can embody the intermediate entity between applications and the vehicle. While being continuously synchronized with the vehicle, the DT can feed applications built upon collected real-time and historical vehicle data and augment the vehicle's capabilities. Moreover, it can model the vehicle and help simulating and predicting its behaviour.

While the DT concept has been around since many years, its application to the automotive domain is still in its infancy. With the exception of a few works, e.g., [7], solutions in the literature mainly target the definition of a DT for a specific vehicle-related task, e.g., trajectory prediction scheme for platoons [8], support at non-signalized intersections [9], vehicle stability monitoring [10].

In the aforementioned works, concrete implementation details about data collection and delivery from the vehicle as well as about their storage and processing at the DT paired with the vehicle, are also typically not provided. Moreover, an holistic evaluation of the most relevant metrics to characterize the DT performance in terms of communication and computing footprint, from a deployment perspective, is also missing.

Moreover, most of them miss workflows and data models to enable an efficient and interoperable data sharing, as well as their storage. Indeed, the data generated by the vehicle are considered as the new gold by several stakeholders in the ecosystem, e.g., insurers, car repairs and manufacturers, public authorities [11]. Hence, access to them by third parties should be easily enabled.

In this work, in order to fill these gaps, we aim to complement the existing literature by providing the following main original contributions:

- A *comprehensive general-purpose architectural framework* is proposed of an *edge-based Vehicular Digital Twin (VDT)*, where the different physical and virtual components are identified and their role explained.
- Interfaces among the physical and the digital counterparts, as well as among the VDT and third-party applications exploiting vehicle data, are specifically designed, with relevant workflows. Such *interfaces are defined both in terms of communication protocols as well as data models*.
- A *Proof of Concept (PoC) is designed to practically implement the envisioned framework* by leveraging off-the-shelf hardware and software components. In particular, in the PoC, the VDT is deployed as a *containerized microservice* at the edge of the network.
- Experimental results are reported in terms of *communication and computation footprint*, under different workloads to mimic different representative CAVs applications, in order to provide guidelines for the actual deployment of the envisioned framework.

The rest of the paper is organized as follows. In Section II, related works are scanned. In Section III, the proposed architecture is presented. In Section IV, the experimental setup and the main findings of the conducted evaluation are discussed. Section V reports the final remarks and hints on future works.

II. RELATED WORK

Originally developed to improve manufacturing processes [12] and even before by the National Aeronautics and Space Administration (NASA) to simulate aerospace vehicles [13], the DT concept can be also leveraged in the automotive domain [14], [15]. By creating digital models that simulate the behavior and interactions of physical entities (e.g., drivers and vehicles) in a transportation network, it is possible to optimize routes, schedules, and other factors to improve efficiency, reduce congestion, and enhance user experience.

Applications that can benefit from DTs associated to CAVs are extensively discussed in [16]. They range for instance from onboard diagnostics to the creation of High Definition (HD) digital maps. Similar topics are addressed in the work in [17].

In [18] the DT is leveraged to create the virtual model of a 5G CAV in order to analyze and emulate its performance under different road and connectivity scenarios in a safe, reliable and secure manner.

The work in [8] proposes a DT-based real-time trajectory prediction scheme for platoons. In particular, the platoon

leader collects the sensing data in real time and processes the trajectory prediction with neural networks distributed among platoon members. It also maintains a DT to optimize the update of the model, to ensure the prediction accuracy and minimize the consumption of communication and computing resources. In [9] a DT is developed to support a cooperative driving system at non-signalized intersections, whereas a DT vehicle stability monitoring system based on the side slip angle is designed in [10]. An overview of different DT platforms that can be used in Electric Vehicle (EV) applications is presented in [19].

Initially, the DTs of vehicles and of other physical entities were deployed in the cloud [20], but then deployment at the edge became an asset. The work in [21] proposes to model the behaviour of DTs of humans (commuters) and public transport vehicles in the context of Mobility as a Service (MaaS). There, the Constrained Application Protocol (CoAP) and the Message Queue Telemetry Transport (MQTT) protocol are considered as potential candidates for interactions between physical entities and DTs hosted at edge facilities.

Due to the mobility of the vehicle, the DT at the edge may need to be migrated so that the digital model of the vehicle and its augmentation is in the closest edge computing node to the vehicle to ensure low-latency interactions. This issue is tackled in [22] and [23].

The work in [24] proposes a Driver Digital Twin (DDT) for the online prediction of personalized lane change behavior, allowing CAVs to predict surrounding vehicles' behaviors with the help of DT technology. A hierarchical cloud-edge architecture is considered, enabling both real-time and bulk-batch collection, processing and analytics of personal data.

A comprehensive Mobility Digital Twin (MDT) framework is developed in [7], which is defined as an Artificial Intelligence (AI)-based data-driven cloud-edge-device framework for mobility services. The proposed MDT consists of three entities in the physical space (namely, human, vehicle, and traffic), and their associated DT in the digital space. The authors scan different off-the-shelf solutions to be concretely deployed for data collection and processing. However, they provide early experimental results.

The work in [25] proposes an end-to-end framework to efficiently perform data collection, offloading, and processing aimed at the construction of a high-fidelity and real-time DT model for CAVs. Edge facilities are considered to provide the capabilities of real-time data processing and pre-built DT model storage. However, the work provides too high-level details about the proposed framework, by mentioning the main functions only.

Whatever the deployment option (edge/cloud), the choice of the application-layer messaging protocols and of data models plays a crucial role in the interactions between the physical entities and their digital counterparts and between the latter ones and applications.

Moreover, the computing footprint associated to the DT needs to be adequately understood to provide guidelines for their actual deployment at available edge/cloud facilities.

To the best of our knowledge, such issues are poorly investigated in the literature and motivate our work.

III. THE PROPOSED FRAMEWORK

A. TARGETED OBJECTIVES AND REFERENCE ARCHITECTURE

To enable innovative applications for smarter mobility, proper solutions need to be devised to complement the capabilities of next-generation vehicles. In this work we propose to leverage the DT concept to serve this purpose. The actual DT deployment entails the design of a comprehensive architecture where all the interacting components are specifically detailed, along with interfaces among them, for a holistic understanding of all the required workflows.

The main components of the envisioned architecture (see Fig. 1) are: *(i)* the vehicle representing the physical entity with embedded OBU and on board sensors; *(ii)* the digital twin of the vehicle, i.e., the VDT, with the related data models and repositories; *(iii)* the southbound interfaces, which allow the interactions between the vehicle and its digital representation, a.k.a. intra-twin communications; *(iv)* the northbound interfaces, which allow the interactions between the VDT and applications which may need to consume the services it provides.

The following objectives are specifically targeted and motivate the design choices for each envisioned architectural component:

- *Low latency*. Both southbound and northbound interfaces should envision communication protocols which ensure the retrieval of data, regardless of the specific network connectivity option, in near real-time whenever needed,[1] e.g., for edge-assisted cooperative perception. The placement at the edge or cloud of the VDT should also account for such a requirement.
- *Low communication footprint*. The burden on the network for interactions over the southbound and northbound interfaces should be kept low, given the expected amount of massively transmitted data to feed the VDT. Hence, compact and efficient data models and low-overhead communication protocols are needed. This is especially true for the southbound interfaces which may involve data exchange over a radio interface, expected to be increasingly loaded.
- *Interoperable data sharing*. Data retrieved from vehicles are heterogeneous in nature and may require to be consumed by different applications for different purposes. Hence, semantic-rich data models should be leveraged to ensure interoperability.
- *Efficient data storage*. The huge amount of data retrieved by the VDT need to be stored with low hardware and software costs.
- *Flexible and future-proof design*. The hardware and software components as well as protocols should be

[1]Please notice that the real-time requirement may not hold for all applications.

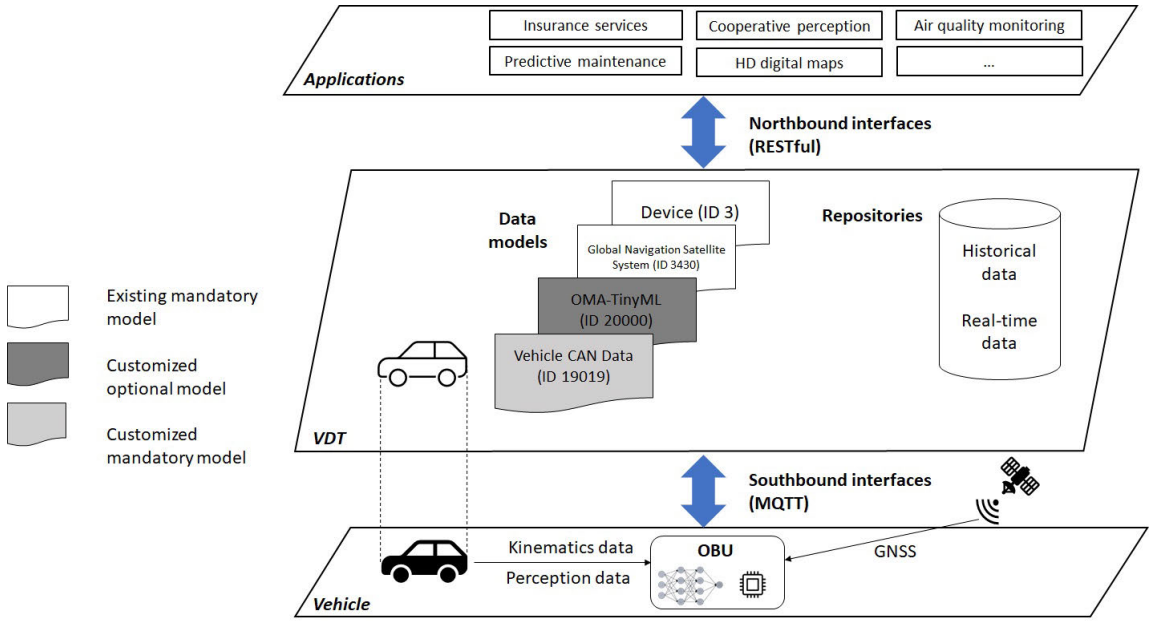


FIGURE 1. The reference architecture.

designed and selected to ensure their simple extensions with additional capabilities/functionalities and to support novel applications. Selected application protocols should be also selected to be *radio technology-agnostic* so to account for the quick advancements in the V2X connectivity realm [4].

In the following, each component of the reference architecture is described in detail.

B. THE VEHICLE

The physical entity is a next-generation vehicle which is equipped with a plethora of sensing devices and communication interfaces in order to retrieve different kinds of data, ranging from, but not limited to, kinematics to environmental and context data.

1) CAN BUS

Kinematics-related data are retrieved from the vehicle's CAN bus. The standards used on CAN bus communications, such as the SAE J1979 [26], are crucial for reliable and efficient data exchange. By establishing communication with the vehicle's on-board diagnostics (OBD-II) system, it is possible to retrieve parameters such as engine speed, coolant temperature, and more. The acquired data are presented in a standardized format, including unique *Parameter Identifiers* (PIDs), data length indicating the size of the data payload, and the corresponding data value. The data collection process follows a request/response method using standard PIDs to obtain their value. CAN bus data can be leveraged by the vehicle itself but also shared with other vehicles in proximity, e.g., to implement a Cooperative Adaptive Cruise Control (CACC) system that enables vehicles to travel in a platoon while minimizing the inter-vehicle spacing [27].

2) ON BOARD SENSORS

The embedded localization module and perception sensors provide additional data. Specifically, information such as latitude, longitude, and timestamp are recorded through Global Navigation Satellite System (GNSS), enabling georeferencing of vehicular data and real-time vehicle position tracking. This way, data gained from other installed on board sensors become more informative. By merging data acquired by vehicle cameras with GNSS data, our framework provides insights into the objects surrounding the vehicle and their locations. If shared with nearby vehicles such data enable cooperative driving, and applications focused on road safety and efficiency (such as forward collision avoidance) [28].

The vehicle can also be utilized as a *probe vehicle*, as described in [29] and [30], for application like environmental monitoring using various sensors. For instance, "Lambda probe" or "NOx sensor" data acquired from the CAN bus can be merged with data retrieved from additional modules installed on board, such as particulate CO2 sensor, or duster sensor. This integration enables real-time georeferencing of emission monitoring in the view of enabling a more sustainable mobility.

3) THE ON BOARD UNIT

A device is available on board to collect data from the CAN bus and the available on board sensors before passing them to the VDT through the available communication interfaces. Next-generation vehicles are expected to be equipped with several V2X radio interfaces, e.g.,

5G Vehicle-to-Infrastructure (V2I) and sidelink Vehicle-to-Vehicle (V2V) connectivity. Hence, instead of using a single network, the interaction between the vehicle and the VDT can occur through the 5G V2I connectivity and/or other options available for medium/long-range connectivity (cellular Fourth Generation (4G), Wi-Fi). Exploiting heterogeneous interfaces would ensure a seamless and low-latency connectivity so to enable an efficient data synchronization between the vehicle and the VDT [31]. Starting from the retrieved raw data, preliminary processing is executed at the On Board Unit (OBU). This processing mainly concerns data aggregation and formatting in a JavaScript Object Notation (JSON) string with a specific structure, in agreement with the data models and semantics of the VDT. Specifically, this string has two main components:

- The *timestamp*, ''tmstp'', retrieved from the GNSS module.
- An *array*, ''e'', containing a collection of the retrieved data reported in *key-value* format. The key (i.e., the *"n"* field) is the resource identifier (i.e., an integer number), while the value (i.e., the *"v"* field) is the actual resource value.

For example, the OBU produces strings such as:

$$\{\text{``tmstp''} : \text{``2023} - 06 - 19\text{T9} : 32 : 34 + 02 : 00'', \text{``e''} : [\{\text{``n''} : \text{``0''}, \text{``v''} : 10\}]\}$$

Besides data collection, the OBU can be equipped with cognitive capabilities. In particular, similarly to [32], we assume that object detection capabilities are available on board to promptly identify road signs, vehicles, pedestrians and their relative position, as streamed from the camera, with greater accuracy than human drivers. The task can be also (partially) offloaded to the edge whenever more powerful computing capabilities are needed, as suggested for instance in [33] and [34]. In our architecture, the VDT can assist the vehicle in performing the cognitive task, if endowed with ML models.

C. THE VDT

Each vehicle is paired with a digital counterpart called the VDT. The VDT is deployed as a containerized microservice at the edge of the network and it is responsible for describing the physical counterpart, modeling/adapting its behavior, and providing for it additional storage, computing, and analytics capabilities. In particular, the VDT is deployed as a container to ensure the digital component to be lightweight and occupy a few computing and storage resources, by also facilitating migration procedures [23].

The chosen placement at the edge is meant to ensure low-latency interactions with the physical counterpart [35]. Moreover, by running at the edge, the VDT can more easily interact with other VDTs and also with natively provided context-aware edge services, as those provided by the European Telecommunications Standards Institute (ETSI) Multi-access Edge Computing (MEC) architecture [21]. The migration of the VDT from an edge server to another one is needed to ensure proximity to the paired vehicle.

1) VDT DATA AND REPOSITORIES

The VDT is constantly synchronized with the physical counterpart from which it gets kinematics, position, perception and environmental-related data which are transmitted in a standardized format, as detailed in the following.

The VDT processes, parses, and stores the received data. Both real-time and historical data are stored to serve a wide set of monitoring applications and also encompassing those which can predict the vehicle's future behavior. For example, by leveraging kinematics data, it is possible to manage traffic flow, monitor vehicle mechanical parts status and schedule maintenance routine before severe issues, thus increasing road safety and efficiency.

As vehicles' OBUs might also run local processing (e.g., to aggregate data, to run inference upon collected data), the VDT does not only store the output of those operations making them available for further processing, but also has a blueprint of the configuration of those tasks and their performance figures in terms of consumed resources.

Raw and/or pre-processed perception data retrieved by the VDT from the vehicle are made available to other microservices (at the edge).

They can be merged with those acquired by other VDTs hosted either in the same or nearby edge servers, so enabling cooperative perception applications based on retrieved features map (e.g., maps describing context around the vehicle) that expands the vehicle's field of view [36].

In our design, the VDT storage capability relies on a different container running a high-performance non-relational database that can handle high-level concurrency Create, Read, Update, and Delete (CRUD) operations and frequent updates.

Having a non-relational database allows to easily and quickly store and manage large amounts of data due to its intrinsic properties such as high data throughput, scalability, flexibility, and support for multiple operation transactions [37]. Hence, this design choice complies with the need for high data freshness and a minimum 10 Hz update frequency required by CAV applications to track vehicle status and position in real-time [32].

Deploying the database as a separate container enables more flexible migration procedures [38]. For instance, considering the mobility of a vehicle and of the corresponding VDT, from edge node *A* to edge node *B*, a new copy of the stateless VDT can be easily created at edge node *B* to ensure proximity, while the original VDT is still working at edge node *A*. As the new VDT at edge node *B* takes over, the received data will be stored in a new instance node of the database cluster created at node *B*. Clusterization will ensure synchronization between the two database nodes, the newly created and the former one.

2) VDT DATA MODELS

To ensure that data retrieved by the VDT can be easily understood by the VDT itself and other parties (such as mobility and insurance providers), other VDTs in case of cooperative (driving) applications, and guarantee interoperable data sharing, they need to be presented in a standardized manner. To this aim, we rely on the Open Mobile Alliance (OMA) Lightweight Machine-to-Machine (LwM2M) protocol [39], designed for Internet of Things (IoT) devices. The OMA-LwM2M standard defines a lightweight client-server protocol leveraging descriptive semantic models of involved devices written as eXtensible Markup Language (XML) schema.[2] The models define objects (e.g., a vehicle, a traffic light, etc.), each representing a specific hardware or software component, and associated object's resources (e.g., latitude, longitude, sensors, network's parameters, etc.) that include attributes like value, unit, maximum and minimum values.

To ensure unique identification of resources, the standard mandates the use of Uniform Resource Identifier (URI) paths composed of *ObjectID/InstanceID/ResourceID*. Each field in the above mentioned URI is allowed to assume numerical values. In certain cases, multiple instances of the same object may exist on a device, like a temperature sensor, and the *InstanceID* component is used to distinguish them.

The OMA maintains a publicly accessible registry of standard objects and resources, facilitating contributions from developers to create new standard objects or utilize customized objects within their respective environments [40], [41]. By employing identifiers to label and encode information, the OMA-LwM2M offers a compact and efficient data model, well suited to minimize the data transfer requirements over the southbound interfaces.

In our framework, a vehicle is represented as a set of OMA-LwM2M objects, some of them already existing in the registry, other defined on purpose. Furthermore, part of those objects are mandatory as they represent basic data common to all vehicles, while others are optional as representing additional information specific to a group of vehicles or even to a single vehicle. An example of those additional resources are pollution monitoring sensors that can be added to some probe vehicles performing environmental monitoring. Among the optional objects we can also list those objects containing the blueprint and the performance figures of the applications running on a given vehicle.

Specifically, we considered the following mandatory objects:

- *Device* (ID 3): This is a standard OMA object containing a description of a generic device (the vehicle itself in our case). It contains 23 resources conveying information such as the manufacturer, the model number, the serial number.
- *Global Navigation Satellite System* (ID 3430): This is a standard OMA object providing all the information required to calculate the position/location of the vehicle [42]: e.g., latitude, longitude, elevation and others.
- *Vehicle CAN Data* (ID 19019): This is a customized OMA object to describe data related to the vehicle's kinematics, extracted from the CAN bus. Creating a custom OMA object model implies the definition of the identifiers necessary to pinpoint the object and its resources; referring to the OMA-LwM2M standard, we chose this model ID (i.e., 19019), in the range allocated for models registered by individuals or companies. A short description of the model is reported in Table 1. Without loss of generality, a subset of kinematics-related data, as extracted by the CAN bus and transmitted by vehicular OBU, are represented as resources. Others can be flexibly added.

To give a practical example of our design, the URI *19019/0/0* refers to the vehicle resource Revolutions per Minute (RPM) identified by the OMA-LwM2M path *ObjectID/InstanceID/ResourceID*. In such example: the *ObjectID* is the Vehicle CAN Data ID (19019); the *InstanceID* is 0; and the *ResourceID* is the RPM resource ID equal to 0 as described in the OMA-LwM2M *XML* semantic meta model of the Vehicle CAN Data custom Object previously introduced and reported in the following table.

Besides the mandatory objects, a vehicle might be described by further optional objects. The inclusion of optional objects makes our data model easily extensible at any time to describe the specific features of a single vehicle as well as the possible aftermarket add-ons installed during the vehicle's lifetime. As an example, if a vehicle is equipped with an external $CO2$ sensor, the standard OMA object 6047 should be included in the description of that vehicle.

In the same manner, our proposed data model can be extended to take into account all the specific tasks performed in the OBU. Particularly relevant to this work are object detection ML inference tasks running on the vehicles which can be easily represented through an OMA-LwM2M object using the *OMA-Tiny ML* object model we devised in [43]. The VDT could ask a vehicle to perform object detection to serve, for instance, an application responsible for the cooperative creation of HD digital maps [34]. A short description of the aforementioned object is reported in Table 2.

Specifically, the resource called ''AI Application'' indicates the executed inference task (e.g., object detection), the resource ''Model'' denotes the utilized ML model (e.g., Convolutional Neural Network (CNN)), the resource ''CPU'' indicates in GHz the computing capabilities available on the device where the inference task is running (e.g., 2.5 GHz), the resource ''Start Inference'' allows to trigger on-demand the task execution, and the resource ''Output'' describes the inference output using a JSON-formatted string. In our context, the latter resource has values such as ''[[264, 250, 544, 377]]['car'][0.92]'', describing where the object has been detected inside the analyzed frame (i.e., the first four numbers represent the box center offset along x and y axis, the

[2]http://openmobilealliance.org/tech/profiles/LWM2M.xsd

TABLE 1. OMA-LwM2M model for object *Vehicle CAN Data* (ID 19019).

Resource ID	Resource Name	Type	Unit
0	Engine RPM	Integer	1/min
1	Fuel Level	Float	%
2	Vehicle Speed	Integer	km/h
3	Engine Coolant Temperature	Float	°C
4	Vehicle Accelerator Pedal Pos.	Integer	%
5	Ambient Air Temperature	Float	°C
6	Fuel Rate	Float	l/h

TABLE 2. OMA-LwM2M model for object *OMA-Tiny ML* (ID 20000).

Resource ID	Resource Name	Type	Unit
0	AI Application	String	
1	Model	String	
2	CPU	Float	GHz
3	Start Inference	Executab.	
4	Output	String	

box width and height in pixels), class (i.e., car), and related detection accuracy (i.e., 0.92).

D. THE SOUTHBOUND INTERFACES

The southbound interfaces allow data exchange between the real vehicle and its virtual counterpart, the VDT.

Whatever the radio interface available on board, according to the protocol implemented by the vehicular OBU, the VDT can expose multiple southbound interfaces protocols, e.g., MQTT [44], CoAP [45], which provide a reliable and efficient two-way communication, and more protocols can be flexibly added. In the following, without loss of generality and similarly to [20] and [46], we rely on MQTT.

MQTT is a communication protocol designed for the IoT that adopts a lightweight approach and employs the publish/subscribe paradigm [47]. MQTT clients can operate as both publishers and subscribers. The latter ones subscribe to a particular *topic* and receive notifications when a new message related to that topic is published by a publisher through a broker, which acts as a server with the main purpose of forwarding packets between publishers and subscribers. MQTT topics are organized in a hierarchical structure with forward slashes (/) used as delimiters. Each level of the hierarchy must have at least one character to be considered valid. The protocol offers low overhead interactions and supports real-time messaging. These features make it suitable for time-sensitive applications in the CAVs context, where a vehicle can be at the same time a data publisher or subscriber, for example, to publish telemetry data and to receive external commands to/from the VDT. In the envisioned framework all MQTT publish messages include a payload formatted in JSON, adhering to the OMA-LwM2M semantics.

Our framework supports the bidirectional communication between the actual vehicle and its VDT by leveraging three primitives defined using three different topic prefixes, namely:

- *cmnd*: this primitive is employed by the VDT to trigger command execution on the vehicle and to force/query the publication of the device resource status. For example, the VDT operates it to trigger actuation on the vehicle's settings or to query the actual status of the vehicle. Moreover, this primitive can be used to enable the VDT to observe some telemetry data that the vehicle publishes over time without the need to issue multiple requests.
- *stat*: this primitive is utilized by the vehicle to answer a query (i.e., a *cmnd* publish) formerly issued by the VDT as well to notify the VDT about a punctual update on a specific vehicle's parameter (e.g., an alarm indicator lighting on) using this interface.
- *tele*: this primitive is utilized by the vehicle to exchange telemetry data with the VDT at fixed time intervals. It is utilized to periodically notify the VDT about the value of the data describing its status (i.e., RPM, Position, $CO2$, etc.) in the observation process. Optionally, telemetry can be set to forward messages only when resource data change value.

Those primitives are supported through MQTT messaging as follows. Either entities, the vehicle and the VDT, subscribe topics such as *prefix/VIN/ObjectID/InstanceID/ResourceID* where *prefix* represents the specific communication primitive to be implemented, *Vehicle Identification Number (VIN)* identifies the vehicle to be monitored, and *ObjectID/InstanceID/ResourceID* have the semantic URI formerly described. To cumulatively subscribe to a block of topics, the MQTT wildcard # can be used to indicate all the topics hierarchically below it and wildcard + can be used to indicate all the topics regardless of what may be contained in the topic level where the special plus character is used. As an example, the string *tele/VIN_x /#* indicates a subscription to all the telemetry topics concerning the vehicle identified by *VIN_x*. Instead, the string *+/VIN_x/#* indicates a subscription to all the topics concerning the vehicle identified by *VIN_x* regardless the prefix. In the same way, a subscription finishing at the *InstanceID* level (e.g., tele/VIN_x/19019/0) is utilized for a single OMA Object, and a subscription terminating at the *ResourceID* level (e.g., tele/VIN_x/19019/0/0) refers to a single OMA Resource belonging to an OMA Object. The actual communication is carried out by posting/receiving messages on the most appropriate topic. As an example, a vehicle, identified by *VIN_x*, wishing to communicate to its VDT telemetry information, such as an update that its coolant temperature has reached 98 $^{\circ}C$, should post this value on the topic *tele/VIN_x/19019/0/3*.

Furthermore, in alignment with the evolutions proposed in the OMA standard [48], the southbound interfaces support OMA-like "Read" and "Read-Composite" operations. The Read operation allows to request reading for: *(i)* single object, specifying the *ObjectID* in the path, which includes all the instances and, therefore, all the resources of the instances; *(ii)* single object instance, specifying *ObjectID/InstanceID* path, which includes all resources of that instance; and *(iii)* single resource, specifying the entire

ObjectID/InstanceID/ResourceID path, which includes only the resource specified in the declared path.

Using the newly added Read-Composite request instead, received packets can be composed by resources belonging to different objects as described in the example below:

$$\{\text{“tmstp”} : \text{“}2023-06-19T9:32:34+02:00\text{”},$$
$$\text{“e”} : [\{\text{“n”} : \text{“}3430/0/1\text{”}, \text{“v”} : 38.168755\},$$
$$\{\text{“n”} : \text{“}3430/0/2\text{”}, \text{“v”} : 15.644197\},$$
$$\{\text{“n”} : \text{“}19019/0/0\text{”}, \text{“v”} : 779\}]\}$$

where the Latitude and Longitude resources of the GNSS object (ID 3430) are conveyed along with the Engine RPM resource of the Vehicle CAN Data object (ID 19019). Such operation allows to read any combination of objects, object instance and resources of different objects in a single communication (i.e., MQTT publish packet) reducing the number of packets exchanged for the same number of transferred resource's values.

Then, southbound interfaces are devoted to vehicle management which includes data collection for monitoring purpose such as vehicle control. The control can be exercised both through the parametric setting of resource values and through the sending of commands (i.e., using the OMA-LwM2M Execute operation). To this end, a topic starting with the *cmnd* prefix is used to indicate that the published data are commands. Furthermore, the actuation topic has to continue up to the *ResourceID* level to perform the action by a specific OMA-LwM2M resource. Thus, a topic used to initiate actions is as follows:

$$cmnd/VIN/ObjectID/InstanceID/ResourceID$$

For instance, to control the *RPM* of the vehicle identified by *VIN_x*, the VDT should post the value of RPM it wishes to set under the the topic *cmnd/VIN_x/19019/0/1*. In the same way, the VDT can trigger an inference process on board the vehicle by publishing a message under the topic *cmnd/VIN_x/20000-/0/3*.

E. THE NORTHBOUND INTERFACES

These specific interfaces are exposed by the VDT in favor of several applications and/or services interested in the data retrieved/processed by the VDT.

Applications and services may request access through web-oriented protocols and based on the Representational State Transfer (RESTful) approaches. In fact, the northbound interfaces of the VDT use the HyperText Transfer Protocol (HTTP) enabling not only reading operations but also management operations on vehicle. The expected primitives, based on HTTP methods, are: READ, READ realtime, WRITE, EXECUTE, OBSERVE, DELETE Observation.

In order to maintain semantic interoperability, the message payload is based on the OMA-LwM2M semantics in JSON format. Moreover, interfaces to request aggregated and/or filtered data based on the keys declared in the body of the HTTP request are added to the aforementioned primitives.

IV. PERFORMANCE EVALUATION

The evaluation study aims at *(i)* providing a preliminary but realistic PoC of the proposed framework by leveraging off-the-shelf hardware and software components, properly combined and overhauled to meet the targeted objectives and *(ii)* assessing the communication and computation footprints of the envisioned components and workflows, under different settings.

A. EXPERIMENTAL SETUP

In the following the main hardware and software components are described in detail and graphically sketched in Fig. 2(a), whereas the corresponding experimental set-up is shown in Fig. 2(b).

1) HARDWARE COMPONENTS

An ELM327 module is used for translating CAN bus messages coming from ECU. On the one hand, it is connected to the vehicle's CAN bus through the OBD-II port, and on the other hand to the ESP32 microcontroller (i.e., one of the OBU components) via Bluetooth.

The purpose of this system is to extract data from the CAN bus and parse it using specific code running on the microcontroller.

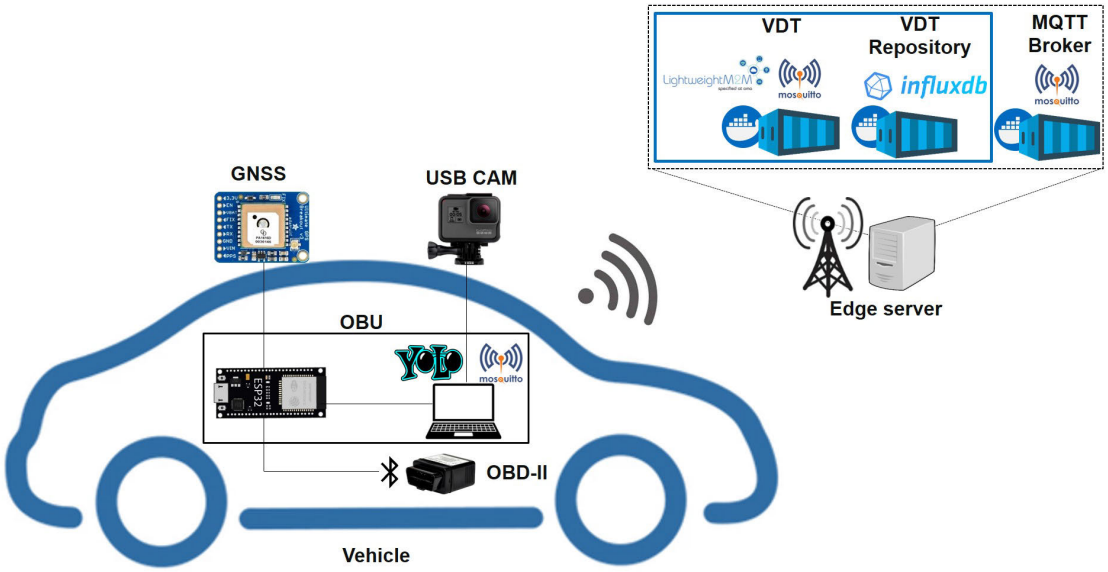
The system retrieves various data from the CAN bus, such as kinematics parameters, sensor readings, vehicle diagnostics. Such raw data are then processed and merged with GNSS coordinates obtained from an Adafruit Ultimate GNSS Breakout receiver [49], a high-quality and energy-efficient GNSS module that can track up to 22 satellites on 66 channels, with an excellent high-sensitivity receiver, and a built-in antenna. This GNSS module is connected to the ESP32 microcontroller. The latter one is responsible for sending the merged data to the VDT hosted on an edge server. For the purpose of experimentation, the edge server facility is implemented by a laptop with CPU Intel Core i7-6500U, 12 GB RAM and 512 GB SSD.

For the ease of PoC implementation, the cognitive tasks concerning the on-board object detection are performed through a laptop, to which the USB camera is connected.

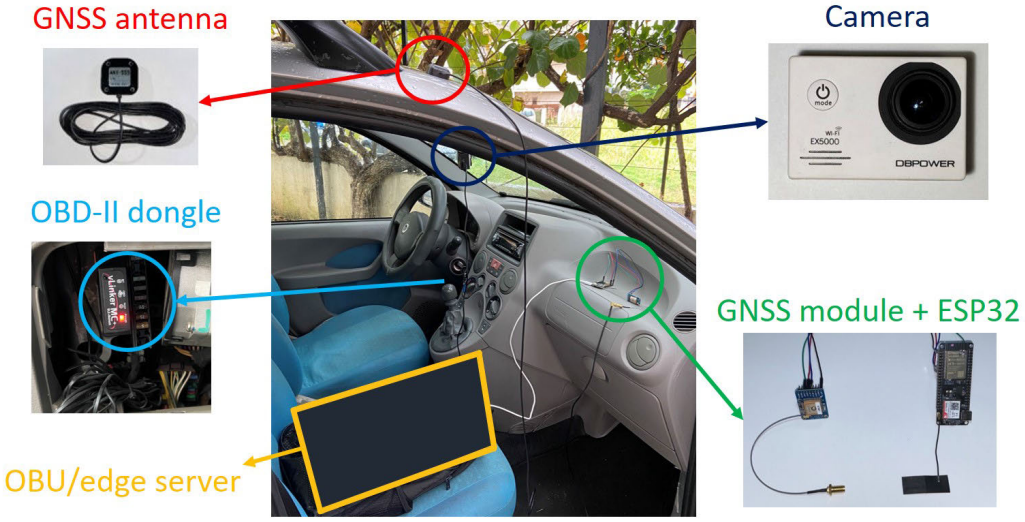
2) SOFTWARE MODULES

Different software modules complement the hardware setup. The following three lightweight and easy to deploy libraries are hosted at the OBU side:

- The ELMDuino [50] is the library employed to perform queries through the ELM327 module. In particular, this library is based on the OBD-II standard and enables the reading of the data reported in Table 1, without knowing OBD-II specifics (e.g., PIDs).



(a) The designed framework.



(b) The in-vehicle experimental setup.

FIGURE 2. **The developed PoC: main hardware and software components.**

- The library chosen to convert GNSS signals acquired into location data, such as latitude, longitude, timestamp, is TinyGPS++ [51].
- The library used to perform on-board object detection tasks is cvlib [52]. Indeed, our system relies on a Python script where cvlib functions are employed to acquire a video streaming through a USB camera and then process it. Specifically, the leveraged ML model is Yolov4-Tiny [53], a tiny version of the well-known Yolov4 model, based on a CNN, designed to run using fewer hardware resources.

All the aforementioned libraries are open source.

An MQTT client is also hosted at the OBU. The Mosquitto client implementation [54] has been chosen since it is considered one of the most popular MQTT implementations due to its simplicity in installation, operating system portability, and a few lines of code [55], well matching the OBU constraints.

The envisioned framework also relies on three Docker containers running on the device acting as an edge server. Specifically, we chose Docker [56] as a lightweight virtualization platform due to our requirements for scalability at the

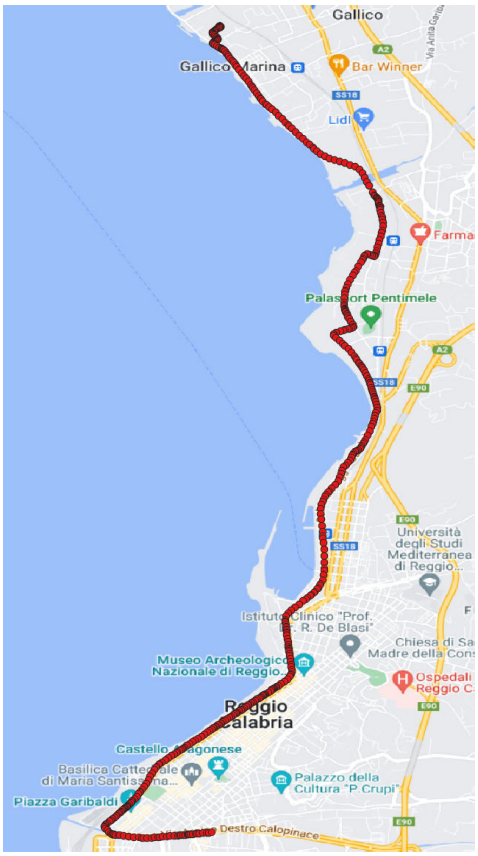


FIGURE 3. Trip of the experimental test-bed in the city of Reggio Calabria.

edge. In particular, our Docker ecosystem is composed of the following containers:

- The Mosquitto container: it runs the namesake Mosquitto Docker image and serves as the MQTT broker, responsible for facilitating communication between the VDT and the vehicle. Its scalability compared to other MQTT open-source implementations has been recently proven in [57].
- The VDT container: it runs our custom image for the VDT. It is specifically designed to include an MQTT client, to parse received data using the OMA-LwM2M data models and to interact with the repository.
- The InfluxDB2 container: it runs the InfluxDB2 Docker image and acts as the repository for the VDT. InfluxDB2 is a high-performance time series database that allows efficient storage, querying, and retrieval of time-stamped data [58]. Its usage to create a DT platform is also foreseen in [59], where an Industry 4.0 context is considered. The InfluxDB2 container integrates seamlessly with the VDT container, providing a robust and scalable solution for storing and managing the VDT data; interactions occur through HTTP primitives.

B. METRICS

The following metrics have been measured.

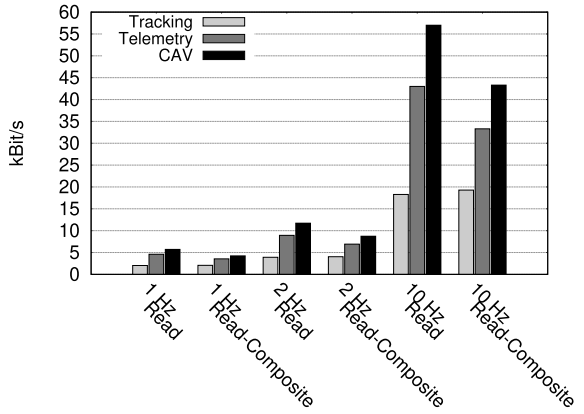


FIGURE 4. Intra-twin communication footprint for different sets of data and data transmission frequencies using Read and Read-Composite operations.

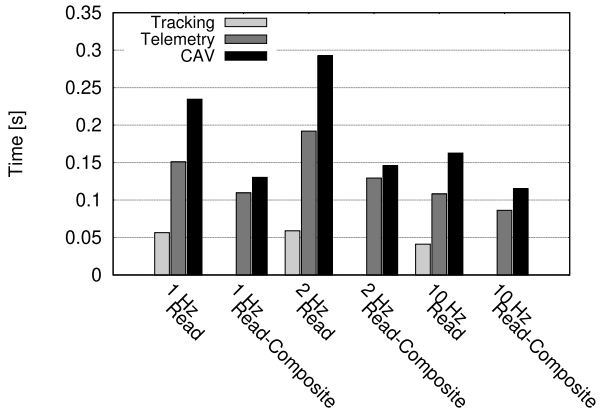
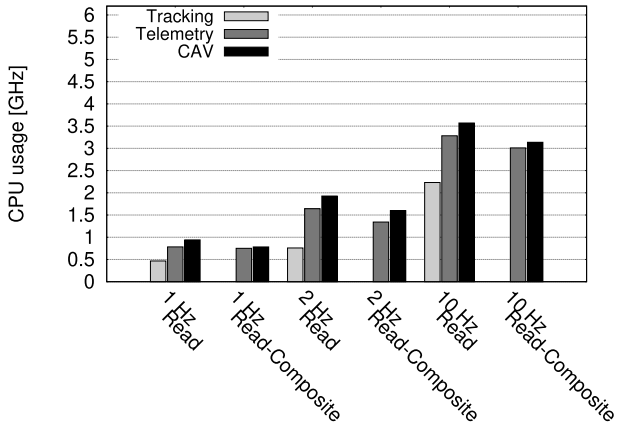


FIGURE 5. VDT processing time for different sets of data and data transmission frequencies using Read and Read-Composite operations.

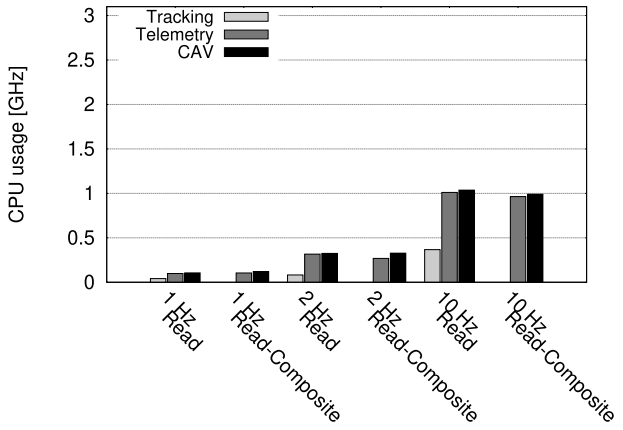
- *Intra-twin communication footprint*: it refers to the amount of kbit/s transmitted over the southbound interfaces from the OBU to the VDT through MQTT publish messages with the *tele* primitive as well as the relevant signaling messages (e.g., TCP ACK). It is measured through the *tcpdump* tool [60]. Such a metric allows to assess how the presence of the VDT impacts the network load over the radio segment.
- *VDT processing time*: it is computed as the time taken by the VDT to parse data received (in JSON format) from the OBU and store them in the InfluxDB2 container. It provides a measure of when data are actually usable after being retrieved.
- *Edge resource utilization*: it refers to the amount of computational resources spent at the edge to run the VDT and InfluxDB2 containers, during their operations. It is measured through the Linux utility *top* and allows to infer the scalability of the proposal.

C. RESULTS

Results have been derived for different sets of data being transmitted by the OBU to the VDT, during

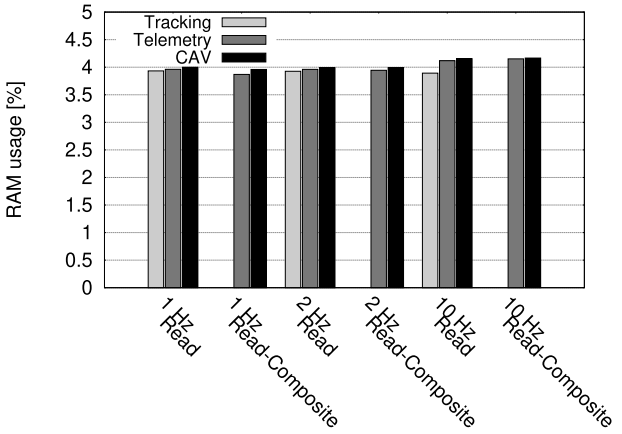


(a) VDT container.

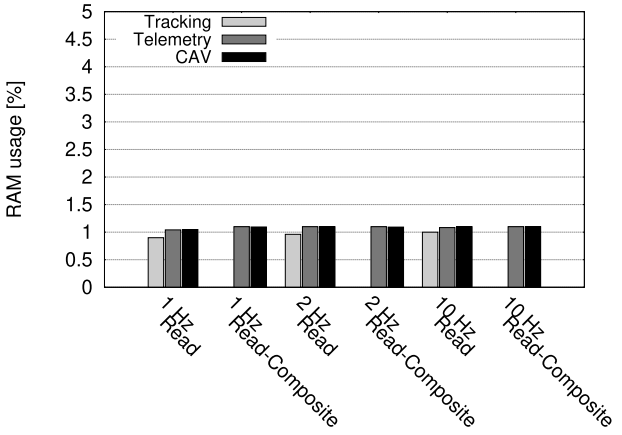


(b) InfluxDB2 container.

FIGURE 6. CPU usage for different sets of data and data transmission frequency values using Read and Read-Composite operations.



(a) VDT container.



(b) InfluxDB2 container.

FIGURE 7. RAM usage for different sets of data and data transmission frequency values using Read and Read-Composite operations.

experiments conducted with a vehicle moving in a urban scenario:

- The set denoted as *Tracking* includes timing and GNSS data, specifically latitude and longitude information, from Object 3430.
- The set of data denoted as *Telemetry*, along with the latitude and longitude, conveys additional kinematics data, corresponding to resources of the Object 19019, as per Table 1.
- The set denoted as *CAV* encompasses the previous sets of data plus those carrying inference results in terms of detected objects, corresponding to the Object 20000. During the conducted journey, the average number of detected objects per frame is equal to 3.14.

The 10 km-long followed path is shown in the map reported in Fig. 3.

Experiments have been performed when varying the update frequency of data transmission from the OBU to the VDT, from 1 Hz up to 10 Hz. The lowest frequency resembles Floating Car Data (FCD) scenarios [29], whereas the highest frequency value may resemble the sampling frequency of data from cameras and LiDARs [61]. Such a value is considered to conduct a stress test both over the radio interface and on containers to be hosted at the edge server.

For all reported results, the OMA LwM2M Read-Composite operation which selectively reads several Resource Instances of different Objects in a single request is benchmarked against the conventional Read operation. For the Tracking data, carrying a single Object, the Read and Read-Composite operations are the same. Hence, in the following Figures, the corresponding bar is only reported for the Read operation, to avoid cluttering the plots.

The first set of results in Fig. 4 reports the intra-twin communication footprint. As expected the larger the amount of data transmitted, the higher the communication footprint, with a maximum amount of data in the order of 60 kbit/s in the worst case, i.e., CAV setting, 10 Hz and Read operation. Despite the small footprint, as the number of involved vehicles (it could be hundreds in large cells) increases, it could heavily burden the uplink channel of the cellular

network, thereby threatening the delivery of traditional traffic or hindering the timely data exchange between the vehicle and the corresponding VDT. Interestingly, thanks to the Read-Composite operation, the total amount of data transmitted can be reduced up to 24% for the CAV data transmitted with a 10 Hz frequency. This is because it allows the transmission of a single (larger) MQTT publish message containing information regarding different OMA objects, instead of a packet per each object and incurs a lower transport signaling (i.e., 50% and 66% less in Telemetry and CAV scenarios, respectively).

The VDT processing time is shown in Fig. 5. The metric reasonably increases as more data are published, i.e., when passing from Tracking to CAV data. More time is needed to parse the received message(s) and store retrieved resources in the InfluxDB2 container, making them available to other parties and/or processing modules. Whatever the data transmission frequency value, the metric is higher for the Read operation, because multiple packets need to be processed to retrieve the same set of data. When the Read-Composite operation is enabled, instead, packets even received at 10 Hz do not experience bottlenecks while being processed (the processing time does not significantly exceed 100 ms, even when CAV data are considered).

The lower processing time values at 10 Hz compared to the lower frequency values can be explained by a sort of granularity effect. In fact, as shown in Fig. 6(a), up to 10 Hz the CPU consumed by the VDT container remains under the capacity of a single core, i.e., 3.1 GHz in our case, then it is reasonable to assume that the VDT container is using a single core. At 10 Hz the computational load of the VDT increases so that the VDT container gets more processing resources and starts using more than a single core. This intuition is again confirmed by Fig. 6(a) which shows that the overall CPU consumed by the VDT container exceeds the resources of a single core. As a consequence, having now two cores at its disposal, the VDT can perform much better by experiencing a lower processing time compared to the case with a lower data transmission frequency.

In both Fig. 6(a) and Fig. 6(b), the CPU utilization respectively, at the VDT and InfluxDB2 containers, increases as the amount of data and the frequency with which they are transmitted increase. Interestingly, processing a large MQTT message (as foreseen by the Read-Composite operation) requires fewer resources at the VDT compared to processing multiple small size MQTT messages (as foreseen by the Read operation), conveying the same information. The same trend holds for the CPU usage at the InfluxDB2 container which overall is very low. Again, utilizing the Read-Composite operation ensures higher scalability. Indeed, reducing the amount of resources needed per VDT, means that more VDTs can be deployed at the same edge server.

RAM usage is affected by the frequency and the amount of exchanged data, usually rising as these parameters increase. Furthermore, the values are either below or do not exceed significantly 4% and 1% for VDT and InfluxDB2, respectively, as shown in Fig. 7(a) and Fig. 7(b).

V. CONCLUSION AND FUTURE WORKS

In this paper, we have presented the design of an edge-based DT framework for CAV. All the components of the envisioned framework (i.e., the physical entity, the VDT, and southbound and northbound interfaces) have been described in detail along with the rationale behind the relevant design choices and selected deployment options.

A comprehensive PoC has been developed to showcase the viability of the proposal. Moreover, metrics have been measured to understand the performance in terms of communication footprint over the vehicle-VDT radio interface and computation footprint of the VDT at the edge infrastructure.

The analysis has been conducted also from a networking perspective to provide, especially to network operators, helpful insights on the requirements of a VDT framework to be practically deployed at a large-scale.

Results show that the Read-Composite operation is efficient in terms of used communication and computing resources. We can expect that larger benefits can be achieved when further OMA LwM2M objects are subscribed at once and, hence, multiple data are retrieved.

Future works will address the extension of the VDT with cognitive components and the assessment of their computing and communication burden.

REFERENCES

[1] SAE. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Accessed: Feb. 5, 2024. [Online]. Available: https://www.sae.org/standards/content/j3016_202104/

[2] G. Naik, B. Choudhury, and J.-M. Park, "IEEE 802.11bd & 5G NR V2X: Evolution of radio access technologies for V2X communications," *IEEE Access*, vol. 7, pp. 70169–70184, 2019.

[3] A. Bazzi, A. O. Berthet, C. Campolo, B. M. Masini, A. Molinaro, and A. Zanella, "On the design of sidelink for cellular V2X: A literature review and outlook for future," *IEEE Access*, vol. 9, pp. 97953–97980, 2021.

[4] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the Internet of Vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[5] S. Lu, N. Ammar, A. Ganlath, H. Wang, and W. Shi, "A comparison of end-to-end architectures for connected vehicles," in *Proc. 5th Int. Conf. Connected Auto. Driving (MetroCAD)*, Apr. 2022, pp. 72–80.

[6] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167653–167671, 2019.

[7] Z. Wang, R. Gupta, K. Han, H. Wang, A. Ganlath, N. Ammar, and P. Tiwari, "Mobility digital twin: Concept, architecture, case study, and future challenges," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17452–17467, Sep. 2022.

[8] H. Du, S. Leng, J. He, and L. Zhou, "Digital twin based trajectory prediction for platoons of connected intelligent vehicles," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2021, pp. 1–6.

[9] Z. Wang, K. Han, and P. Tiwari, "Digital twin-assisted cooperative driving at non-signalized intersections," *IEEE Trans. Intell. Vehicles*, vol. 7, no. 2, pp. 198–209, Jun. 2022.

[10] J. Wang, C. Zhang, Z. Yang, M. Dang, P. Gao, and Y. Feng, "Research on digital twin vehicle stability monitoring system based on side slip angle," *IEEE Trans. Intell. Transp. Syst.*, 2024.

[11] D. A. Kountche, F. Raissi, M. R. Rakotondravelona, E. Bonetto, D. Brevi, A. Martin, O. Otaegui, and G. Velez, "Monetisation of and access to in-vehicle data and resources: The 5GMETA approach," 2022, *arXiv:2208.11335*.

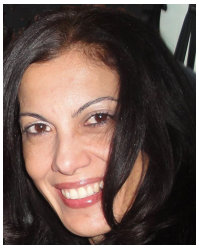
[12] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives On Complex Systems: New Findings and Approaches*, 2017, pp. 85–113.
[13] E. Glaessgen and D. Stargel, "The digital twin paradigm for future NASA and U.S. Air Force vehicles," in *Proc. 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dyn. Mater. Conf., 20th AIAA/ASME/AHS Adapt. Struct. Conf.*, 2012, p. 1818.
[14] W. A. Ali, M. Roccotelli, and M. P. Fanti, "Digital twin in intelligent transportation systems: A review," in *Proc. 8th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, vol. 1, May 2022, pp. 576–581.
[15] G. Bhatti, H. Mohan, and R. Raja Singh, "Towards the future of smart electric vehicles: Digital twin technology," *Renew. Sustain. Energy Rev.*, vol. 141, May 2021, Art. no. 110801.
[16] C. Schwarz and Z. Wang, "The role of digital twins in connected and automated vehicles," *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 6, pp. 41–51, Nov. 2022.
[17] S. M. M. Hossain, S. K. Saha, S. Banik, and T. Banik, "A new era of mobility: Exploring digital twin applications in autonomous vehicular systems," in *Proc. IEEE World AI IoT Congr. (AIIoT)*, Jun. 2023, pp. 493–499.
[18] H. X. Nguyen, R. Trestian, D. To, and M. Tatipamula, "Digital twin for 5G and beyond," *IEEE Commun. Mag.*, vol. 59, no. 2, pp. 10–15, Feb. 2021.
[19] M. Ibrahim, V. Rjabtšikov, and R. Gilbert, "Overview of digital twin platforms for EV applications," *Sensors*, vol. 23, no. 3, p. 1414, Jan. 2023.
[20] D. P. Proos and N. Carlsson, "Performance comparison of messaging protocols and serialization formats for digital twins in IoV," in *Proc. IFIP Netw. Conf. (Networking)*, Jun. 2020, pp. 10–18.
[21] C. Campolo, G. Genovese, A. Molinaro, and B. Pizzimenti, "Digital twins at the edge to track mobility for MaaS applications," in *Proc. IEEE/ACM 24th Int. Symp. Distrib. Simul. Real Time Appl. (DS-RT)*, Sep. 2020, pp. 1–6.
[22] Y. Zhou, A. K. Bashir, J. Wu, Y. D. Al-Otaibi, X. Lin, and H. Xu, "Secure digital twin migration in edge-based autonomous driving system," *IEEE Consum. Electron. Mag.*, vol. 12, no. 6, pp. 56–65, 2023.
[23] C. Campolo, A. Iera, A. Molinaro, and G. Ruggeri, "MEC support for 5G-V2X use cases through Docker containers," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–6.
[24] X. Liao, X. Zhao, Z. Wang, Z. Zhao, K. Han, R. Gupta, M. J. Barth, and G. Wu, "Driver digital twin for online prediction of personalized lane change behavior," *IEEE Internet Things J.*, 2023.
[25] Y. Liu, H. Wang, Z. Cai, D. Chen, and K. Han, "Poster: Enabling high-fidelity and real-time mobility digital twin with edge computing," in *Proc. IEEE/ACM 7th Symp. Edge Comput. (SEC)*, Dec. 2022, pp. 281–283.
[26] SAE. *J1979 E/E Diagnostic Test Modes; J1979_201408*. Accessed: Feb. 5, 2024. [Online]. Available: https://www.sae.org/standards/content/j1979_201408/
[27] G. Nardini, A. Virdis, C. Campolo, A. Molinaro, and G. Stea, "Cellular-V2X communications for platooning: Design and evaluation," *Sensors*, vol. 18, no. 5, p. 1527, May 2018.
[28] J. He, Z. Tang, X. Fu, S. Leng, F. Wu, K. Huang, J. Huang, J. Zhang, Y. Zhang, A. Radford, L. Li, and Z. Xiong, "Cooperative connected autonomous vehicles (CAV): Research, applications and challenges," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2019, pp. 1–6.
[29] O. Briante, C. Campolo, A. Iera, A. Molinaro, S. Y. Paratore, and G. Ruggeri, "Supporting augmented floating car data through smartphone-based crowd-sensing," *Veh. Commun.*, vol. 1, no. 4, pp. 181–196, Oct. 2014.
[30] D. Budimir, N. Jelušić, and M. Perić, "Floating car data technology," *Pomorstvo*, vol. 33, no. 1, pp. 22–32, Jun. 2019.
[31] J. Zheng, T. H. Luan, Y. Zhang, R. Li, Y. Hui, L. Gao, and M. Dong, "Data synchronization in vehicular digital twin network: A game theoretic approach," *IEEE Trans. Wireless Commun.*, vol. 10, no. 15, pp. 13235–13246, 2023.
[32] J. Hochstetler, R. Padidela, Q. Chen, Q. Yang, and S. Fu, "Embedded deep learning for vehicular edge computing," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Oct. 2018, pp. 341–343.
[33] S.-W. Kim, K. Ko, H. Ko, and V. C. M. Leung, "Edge-network-assisted real-time object detection framework for autonomous driving," *IEEE Netw.*, vol. 35, no. 1, pp. 177–183, Jan. 2021.
[34] Y. Xue, Y. Zhang, Q. Liu, D. Chen, and K. Han, "CoMap: Proactive provision for crowdsourcing map in automotive edge computing," 2023, *arXiv:2302.03204*.
[35] M. Picone, M. Mamei, and F. Zambonelli, "A flexible and modular architecture for edge digital twin: Implementation and evaluation," *ACM Trans. Internet Things*, vol. 4, no. 1, pp. 1–32, Feb. 2023.
[36] Q. Yang, S. Fu, H. Wang, and H. Fang, "Machine-Learning-Enabled cooperative perception for connected autonomous vehicles: Challenges and opportunities," *IEEE Netw.*, vol. 35, no. 3, pp. 96–101, May 2021.
[37] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain, "A survey and comparison of relational and non-relational database," *Int. J. Eng. Res. Technol.*, vol. 1, no. 6, pp. 1–5, 2012.
[38] C. Puliafito, A. Virdis, and E. Mingozzi, "The impact of container migration on fog services as perceived by mobile things," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Sep. 2020, pp. 9–16.
[39] *Lightweight Machine to Machine Technical Specification Core; V1_1-20180612-C*, Open Mobile Alliance, San Diego, CA, USA, 2018.
[40] Open Mobile Alliance. *OMA-LwM2M Object Editor*. Accessed: Feb. 5, 2024. [Online]. Available: https://devtoolkit.openmobilealliance.org/OEditor/default.aspx
[41] Open Mobile Alliance. *OMA-LwM2M Object and Resource Registry*. Accessed: Feb. 5, 2024. [Online]. Available: https://technical.openmobilealliance.org/OMNA/LwM2M/LwM2MRegistry.html
[42] Open Mobile Alliance. *OMA Global Navigation Satellite System Model*. [Online]. Available: https://raw.githubusercontent.com/openmobilealliance/lwm2m-registry/prod/3430.xml
[43] C. Campolo, G. Genovese, A. Iera, and A. Molinaro, "Virtualizing AI at the distributed edge towards intelligent IoT applications," *J. Sensor Actuator Netw.*, vol. 10, no. 1, p. 13, Feb. 2021.
[44] A. Banks and R. Gupta, *MQTT Version 3.1.1*, OASIS Standard 29, 2014, p. 89.
[45] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)(RFC 7252)," Jun. 2014. Accessed: Feb. 5, 2024. [Online]. Available: http://www.rfc-editor.org/info/rfc7252
[46] F. Flamigni, P. Pileggi, O. Barrowclough, and J. Haenisch, "First report on standards relevant for digital twins," in *Proc. Change2Twin Consortium*, 2020.
[47] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the Internet of Things," *Trans. IoT Cloud Comput.*, vol. 3, no. 1, pp. 11–17, 2015.
[48] *Lightweight Machine to Machine Technical Specification: Core Approved Version: 1.2.1*, Open Mobile Alliance, San Diego, CA, USA, Dec. 2022.
[49] Adafruit. *ADAFRUIT Ultimate GPS Breakout*. Accessed: Feb. 5, 2024. [Online]. Available: http://www.adafruit.com/product/746#description-anchor
[50] PowerBroker2. *ELMDuino*. Accessed: Feb. 5, 2024. [Online]. Available: https://github.com/PowerBroker2/ELMduino
[51] M. Hart. *TinyGPS++*. Accessed: Feb. 5, 2024. [Online]. Available: http://arduiniana.org/libraries/tinygpsplus/
[52] A. Ponnusamy. (2018). *Cvlib—High Level Computer Vision Library for Python*. [Online]. Available: https://github.com/arunponnusamy/cvlib
[53] C.-Y. Wang, A. Bochkovskiy, and H. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13024–13033.
[54] Mosquitto. *MQTT Open-Source Implementation*. Accessed: Feb. 5, 2024. [Online]. Available: https://mosquitto.org/
[55] E. Bertrand-Martínez, P. D. Feio, V. de Brito Nascimento, B. Pinheiro, and A. Abelém, "A methodology for classification and evaluation of IoT brokers," in *Proc. LANOMS*, 2019.
[56] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, p. 2, 2014.
[57] M. Bender, E. Kirdan, M.-O. Pahl, and G. Carle, "Open-source MQTT evaluation," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–4.
[58] InfluxData. *InfluxDB*. Accessed: Feb. 5, 2024. [Online]. Available: https://www.influxdata.com/
[59] A. Costantini, G. Di Modica, J. C. Ahouangonou, D. C. Duma, B. Martelli, M. Galletti, M. Antonacci, D. Nehls, P. Bellavista, C. Delamarre, and D. Cesini, "IoTwins: Toward implementation of distributed digital twins in Industry 4.0 settings," *Computers*, vol. 11, no. 5, p. 67, Apr. 2022.
[60] V. Jacobson, C. Leres, and S. McCanne. (1989). *TCPDUMP*. [Online]. Available: https://ee.lbl.gov/ftp.html
[61] M. Buchholz, J. Muller, M. Herrmann, J. Strohbeck, B. Volz, M. Maier, J. Paczia, O. Stein, H. Rehborn, and R.-W. Henn, "Handling occlusions in automated driving using a multiaccess edge computing server-based environment model from infrastructure sensors," *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 3, pp. 106–120, May 2022.

CLAUDIA CAMPOLO (Senior Member, IEEE) is currently an Associate Professor in telecommunications with the Mediterranean University of Reggio Calabria, Italy. Her main research interests include vehicular networking and future internet architectures.

BRUNO PIZZIMENTI (Student Member, IEEE) is currently pursuing the Ph.D. degree with the Mediterranean University of Reggio Calabria, Italy. His research interests include connected and autonomous vehicles, the IoT, and device virtualization.

GIACOMO GENOVESE is currently a Researcher with CNIT Consortium, ARTS Laboratory, Mediterranean University of Reggio Calabria, Italy. His research interests include the IoT heterogeneous gateways design, the IoT device virtualization, digital twin, and edge computing technologies.

GIUSEPPE RUGGERI (Member, IEEE) is currently an Associate Professor in telecommunications with the Mediterranean University of Reggio Calabria, Italy. His current research interests include self-organizing networks, the IoT, and the social IoT.



ANTONELLA MOLINARO (Senior Member, IEEE) is currently a Full Professor in telecommunications with the University Mediterranean of Reggio Calabria, Italy, and Université Paris-Saclay, France. Her current research interests include 5G/6G networks and connected vehicles.

DOMENICO MARIO ZAPPALÀ (Student Member, IEEE) is currently pursuing the Ph.D. degree with the Mediterranean University of Reggio Calabria, Italy. His research interests include connected and autonomous vehicles, platooning techniques, and edge solutions.

Open Access funding provided by 'Univ Mediterranea di Reggio Calabria' within the CRUI CARE Agreement