A Novel OMNeT++-Based Simulation Tool for Vehicular Cloud Computing in ETSI MEC-Compliant 5G Environments

Angelo Feraudo[a], Alessandro Calvio[b] and Paolo Bellavista[c]
Department of Computer Science and Engineering, University of Bologna, Viale Risorgimento 2, Bologna, Italy

Keywords: Simulation Tools, Vehicular Cloud Computing, 5G, ETSI MEC, Vehicular Cloud Applications.

Abstract: Vehicular cloud computing is gaining popularity thanks to the rapid advancements in next generation wireless communication networks. Similarly, Edge Computing, along with its standard proposals such as European Telecommunications Standards Institute (ETSI) Multi-access Edge Computing (MEC), will play a vital role in these scenarios, by enabling the execution of cloud-based services at the edge of the network. Together, these solutions have the potential to create real micro-datacenters at the network edge, favoring several benefits like minimal latency, real-time data processing, and data locality. However, the research community has not yet the opportunity to use integrated simulation frameworks for the easy testing of applications that exploit both the vehicular cloud paradigm and MEC-compliant 5G deployment environments. In this paper, we present our simulation tool as a platform for researchers and engineers to design, test, and enhance applications utilizing the concepts of vehicular and edge cloud. The tool implements our ETSI MEC-compliant architecture that leverages resources provided by vehicles. Moreover, the paper analyzes and reports performance results for our simulation platform, as well as provides a use case where our simulator is used to support the design, test, and validation of an algorithm to distribute MEC application components on vehicular cloud resources.

1 INTRODUCTION

Next-generation wireless communication networks are advancing at a rapid pace, leading to the development and prototyping of highly innovative application scenarios. In this context, as the amount of connected vehicles rapidly increases (Elektrobit, 2022), drivers can take advantage of a wide range of distributed applications and services, very often cloud-based, as in other vertical domains, including improved access to information and entertainment features. However, it starts to be widely recognized that traditional client-to-cloud architectures cannot meet the stringent requirements of many time-sensitive Internet of Things (IoT) applications. Furthermore, with the constant growth of intelligent cars, equipped with a large set of onboard sensors (Sabella et al., 2017), the amount of vehicle-generated data may pose a significant challenge to the backbone network.

Edge Computing has emerged as a promising solution for distributed environments, as it offers a more efficient and effective way to handle the ever-increasing demand for computing and storage resources. Moreover, edge computing has the capability to replace traditional cloud solutions due to its potential to reduce network stress, decrease latency, improve efficiency, and enable real-time data processing. To accelerate the adoption of this paradigm, the European Telecommunications Standards Institute (ETSI) has proposed the Multi-access Edge Computing (MEC) standard (ETSI, 2022a) to create a virtualized infrastructure at the network edge as the next key technology for radio networks. The MEC architecture orchestrates, distributes, and manages the life-cycle of MEC-compliant applications enabling their execution closer to the involved data sources and/or users. Nevertheless, dense application scenarios present a significant challenge due to the limited resources within the Multi-access Edge Computing (MEC) infrastructure. This requires the acquisition of additional capacity to address the issue effectively.

Recently, Vehicular Cloud Computing (or more simply Vehicular Computing) has been proposed as a promising paradigm to support distributed applications designed according to the edge cloud approach (Gerla, 2012; Olariu et al., 2011). Vehicular Comput-

[a] https://orcid.org/0000-0002-9727-0861
[b] https://orcid.org/0000-0002-2403-2969
[c] https://orcid.org/0000-0003-0992-7948

Feraudo, A., Calvio, A. and Bellavista, P.
A Novel OMNeT++-Based Simulation Tool for Vehicular Cloud Computing in ETSI MEC-Compliant 5G Environments.
DOI: 10.5220/0012140900003546
In *Proceedings of the 13th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2023)*, pages 448-455
ISBN: 978-989-758-668-2; ISSN: 2184-2841

ing exploits the computing power locally available on today's vehicles to create cost-effective mobile clouds at the far-edge layer. The formation of these dynamic clouds should be achieved autonomously by vehicles, which share their resources among them and/or with nearby edge nodes to extend their virtualized resources for service execution.

Numerous research efforts have been devoted to developing architectures that leverage the resources dynamically available on vehicles (Olariu, 2020). To enhance resource availability at the network edge, various works have proposed to exploit the underutilized computational power of both stationary (Liu et al., 2011; Li et al., 2019) and moving vehicles (Hou et al., 2016). These opportunistic resources can be utilized for diverse purposes to handle the growing number of applications used in vehicular networks. For instance, vehicles can serve as relay nodes (Liu et al., 2011) to improve network connectivity, or as computing nodes (Huang et al., 2018; Rahman et al., 2020; Ma et al., 2021) to reduce the impact of these applications on the performance of edge nodes.

Moreover, some recent studies have focused on providing simulation frameworks (Ahmed et al., 2019) since practical experiments on vehicular network environments are expensive and challenging. In fact, to validate their proposals, some of the works in the literature (Cha et al., 2021; Ma et al., 2021; Rahman et al., 2020; Feng et al., 2017) have utilized these simulation frameworks mainly for i) generating vehicle traces and ii) simulating the behavior of vehicular network protocols. To the best of our knowledge, there is currently no simulation tool that offers a single vehicular computing-based platform where researchers can design/test their algorithms and applications while exploiting at the same time the vehicular cloud computing paradigm and the MEC standard deployment environment.

In this paper, we propose a simulation tool that leverages the ETSI MEC standard to incorporate the resources available at the edge of the network, such as those provided by collaborating vehicles, into the cloud continuum spectrum. The simulation tool implements an extended version of the ETSI standard (Feraudo et al., 2023), which enhances MEC-compliant edge nodes with resources from vehicles within a designated Area of Interest (AoI). The resources provided by the nodes are registered in the edge resource pool and can be accessed through standardized interfaces. As an extension of the ETSI MEC standard, the modeled architecture allows dealing with some of the primary challenges arising in vehicular computing environments, such as integration with cloud resources and enabling coexistence among heterogeneous technologies. Furthermore, it allows dealing with resource volatility issues (i.e., nodes that dynamically join/leave during service provisioning) via a standardized migration mechanism. To build our simulation platform, we utilized the OMNeT++ network simulator as the underlying framework and incorporated the Simu5G library to model the 5G network and communications aspects.The platform is accessible to researchers publicly through the GitHub repository (Feraudo and Calvio, 2023).

The remainder of this paper is structured as follows. Section 2 provides an overview of the related background, which includes a description of the ETSI MEC reference model and vehicular computing paradigm. Section 3 presents the interactions and modules that we have originally implemented to create a MEC-compliant vehicular computing environment in a 5G network. Then, we evaluate the performance of our simulation platform, while performing resource management and by providing a practical proof of its usage in section 4. Section 5 concludes the findings of this work and presents future directions.

Table 1: Table of acronyms for MEC elements.

Abbreviation	Definition
AMS	Application Mobility Service
MEC	Multi-access Edge Computing
MEC-App	MEC Application
MEC-H	MEC Host
MEC-O	MEC Orchestrator
MEC-P	MEC Platform
MEC-PM	MEC Platform Manager
UALCMP	User Application LifeCycle Management Proxy
VI	Virtualisation Infrastructure
VIM	Virtualisation Infrastructure Manager

2 BACKGROUND

2.1 ETSI Multi-Access Edge Computing

Figure 1 includes the MEC reference architecture proposed by ETSI in (ETSI, 2022a). It offers the opportunity to run contextualized MEC-compliant applications within a virtualized and multi-tenant environment. The reference architecture consists of two layers: the system level and the host level, which together comprise all standard functional elements. The system level works as an entry point for authorized users to request the execution of MEC applications. It includes management-related functional elements, such as MEC Orchestrator (MEC-O) and User Application LifeCycle Management Proxy (UALCMP), as well as other components that are necessary for stan-

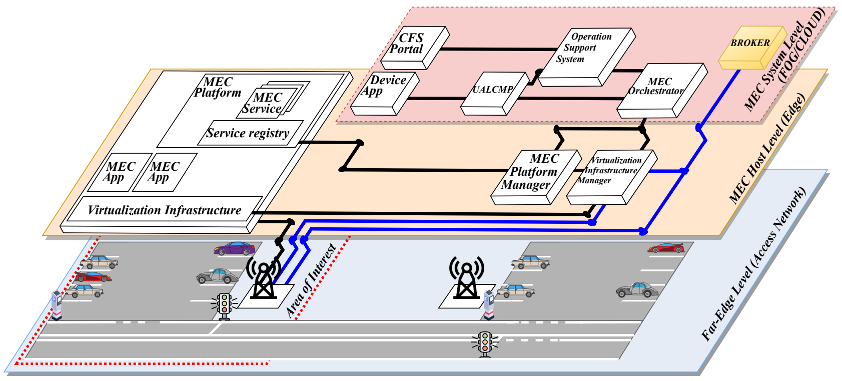


Figure 1: The extended MEC architecture, already presented in (Feraudo et al., 2023).

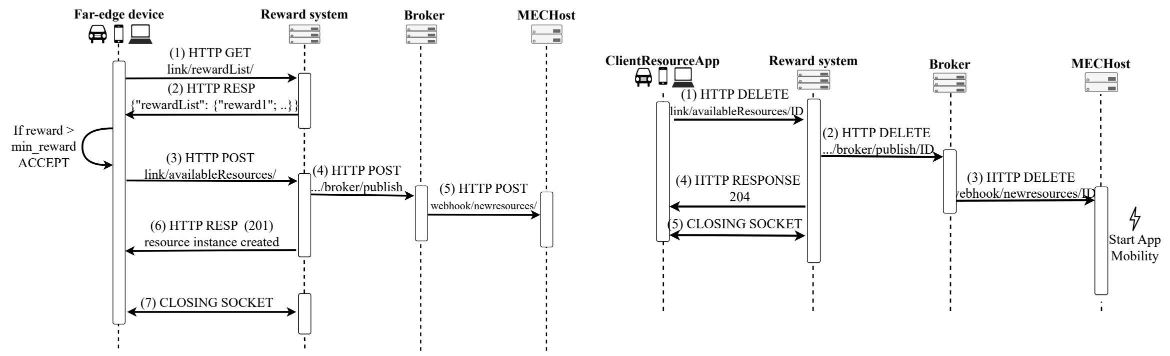
dard operation, e.g., Device Application. The MEC-O is the core functionality for layer management, as it has a global view of the MEC-compliant edge nodes (MEC Hosts) present in a particular area and serves as a controller for MEC application distributions. By selecting the best MEC Host (MEC-H) based on the requirements of the requested services, the MEC-O triggers the instantiation and termination of applications. The second management element, i.e., the UALCMP, acts as an intermediary node between the user and the application. It supports user requests for instantiation and termination, which are forwarded to the MEC-O.

The host level is situated at the edge of the network and is responsible for providing and managing the virtualization platform where MEC applications are deployed. It comprises the edge node that provides virtualized computational, network, and storage resources, i.e., MEC-H. The MEC-H enables MEC application deployment through the Virtualization Infrastructure (VI), which is also responsible for managing the data plane among the network interfaces through traffic policies. The MEC-H also includes a MEC Platform (MEC-P) that exposes a service registry, thus enabling the supported applications to discover, offer, and consume standard services. Moreover, at this level, the Virtualization Infrastructure Manager (VIM) and the MEC Platform Manager (MEC-PM) act as the management-related functional elements and serve as access points for the MEC-O to the lower layer. The VIM is responsible for managing and releasing the virtualized resources and configuring the VI to run software images appropriately. For the sake of clarity, we have reported all the standard-related acronyms in Table 1.

2.2 Vehicular Computing Paradigm

Many current and future applications for connected vehicles can take advantage of edge resources, including infotainment and time-sensitive applications related to traffic safety. The integration of vehicular networks with the MEC infrastructure can be referred as Vehicular Edge Computing (VEC) (Liu et al., 2021). VEC aims to bring computational capabilities to the proximity of vehicular users, allowing for services to be readily available via Vehicle-to-Infrastructure (V2I) communications. Nevertheless, the ever-increasing number of applications and businesses is starting to make the infrastructure unable to efficiently satisfy their demands and possibly stringent requirements, e.g., on latency. For this reason, the need has emerged to identify new resources that can support the more traditional edge infrastructure.

In (Olariu et al., 2011) and (Gerla, 2012), the authors introduced the Vehicular Cloud Computing concept relying on the idea that modern-day vehicles come equipped with powerful on-board computers, ample storage, and an array of sensing devices. In their work (Olariu et al., 2011), the authors define Vehicular Computing as a collaborative way to share resources among vehicles to solve problems that would otherwise require a significant amount of time with a more traditional centralized architecture, in particular for context-specific applications. In line with this, in (Gerla, 2012) the author states that the vehicular cloud paradigm allows keeping the information gathered by vehicle sensors locally and sharing it solely with other vehicles, as the sheer volume of in-vehicle generated data can pose serious technical challenges for the network infrastructure. According to these definitions, a cloud of vehicles can be formed anywhere on the road and their onboard resources can be dynamically allocated to authorized users. Despite its potential benefits, such a vehicular computing environment also poses several challenges that must be addressed. These challenges include distributed ownership, as each vehicle has a single owner responsible for deciding whether to share onboard resources; high node mobility, which makes it difficult to predict the vehicular residency times in the cloud even


Far-edge device
Reward system
Broker
MECHost
(1) HTTP GET link/rewardList/
(2) HTTP RESP {"rewardList": {"reward1"; ..}}
If reward > min_reward ACCEPT
(3) HTTP POST link/availableResources/
(4) HTTP POST .../broker/publish
(5) HTTP POST webhook/newresources/
(6) HTTP RESP (201) resource instance created
(7) CLOSING SOCKET
ClientResourceApp
Reward system
Broker
MECHost
(1) HTTP DELETE link/availableResources/ID
(2) HTTP DELETE .../broker/publish/ID
(3) HTTP DELETE webhook/newresources/ID
(4) HTTP RESPONSE 204
(5) CLOSING SOCKET
Start App Mobility

(a) Resource Acquisition Sequence Diagram. (b) Resource Releasing Sequence Diagram.

Figure 2: Sequence Diagram device-initiated scheme.

when clouds are formed using resources of stationary cars within a parking lot; device heterogeneity, as vehicles are manufactured by different companies; security and privacy. To deal with some of the aforementioned challenges, we introduced a novel ETSI MEC-compliant architecture in our recent work (Feraudo et al., 2023). This architecture expands the edge resource pool by leveraging vehicular computational resources and forms the foundation of the simulation platform described in this paper.

3 OUR SIMULATION PLATFORM FOR MEC-COMPLIANT VEHICULAR COMPUTING

The primary objective of our simulation tool is to provide a platform that facilitates the design, testing, and enhancement of applications by utilizing vehicular computing in 5G environments. The proposed framework works with the event-based simulator OMNet++ and exploits the well-known Simu5G communication library (Nardini et al., 2020) components as a basis to create our envisioned MEC architecture.

In the following sections, we will delve into the key components of our simulation platform and their interactions, enabling the utilization of external resource infrastructure in MEC-compliant environments.

3.1 Simulation Tool: Interactions

This subsection describes the procedures supported for MEC-enabled resource management for vehicular cloud computing, identified on the basis of the MEC extended architecture. Each procedure requires a specific set of interactions to be completed effectively. These interactions involve both MEC standard and non-standard components, which must be carefully considered to ensure that the necessary resources are acquired, released, and allocated.

Resource Acquisition. Figure 2a illustrates the steps required by parking vehicles to partake in the resource acquisition procedure. According to the sequence diagram, whenever a new vehicle accepts the rewards indicated by the MEC-H, it publishes the amount of resources that wants to make available. In addition to resource availability information, the POST request, step (3) in the figure, contains data related to its location, thus allowing the Broker to notify the appropriate MEC-H.

Resource Allocation. An example of interaction scheme for app instantiation on remote resources is reported in Fig. 3. The MEC-O receives an instantiation request from the UALCMP and starts resource/service discovery throughout all the MEC-Hs under its management. Once the MEC-H has been chosen, the VIM receives an instantiation request (step (7)) and starts the scheduling procedure. It should be noted that when the Broker notifies the VIM for new device resource acquisition, the VIM stores the endpoint information, corresponding to an external VI address (see Fig. 4), and the resource capacity of that device. Hence, the VIM is aware of the single contributions that each device brings to correctly allocate its resources when needed. The scheduling phase leads to the identification of a single resource infrastructure, either local or remote, where to deploy the requested MEC applications.

Resource Releasing. Figure 2b shows the interactions needed when devices leave the resource pool. In such a scenario, once the corresponding MEC-H receives the notification, it removes the concerned resources from those available in the pool and starts the

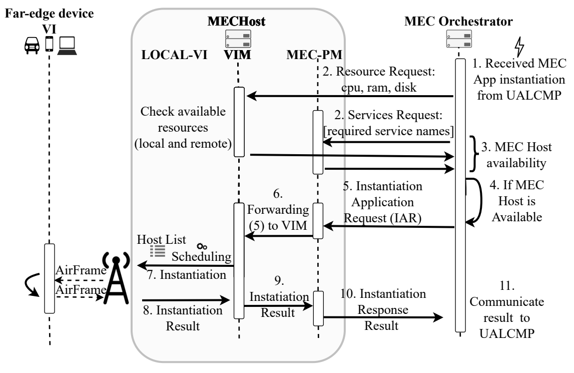

Far-edge device
VI
MECHost
LOCAL-VI
VIM
MEC-PM
MEC Orchestrator
1. Received MEC App instantiation from UALCMP
2. Resource Request: cpu, ram, disk
2. Services Request: [required service names]
Check available resources (local and remote)
3. MEC Host availability
4. If MEC Host is Available
5. Instantiation Application Request (IAR)
6. Forwarding (5) to VIM
Host List
Scheduling
7. Instantiation
AirFrame
AirFrame
8. Instantiation Result
9. Instatiation Result
10. Instantiation Response Result
11. Communicate result to UALCMP

Figure 3: Resource Allocation Sequence Diagram.

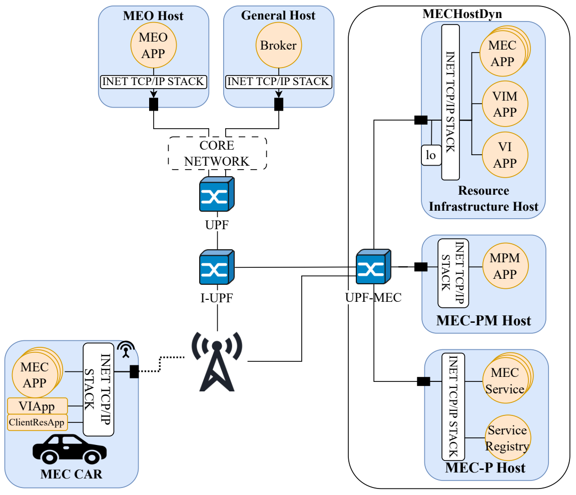


Figure 4: Simulation Tool Modules Structure.

mobility procedure for the apps running on that device. The migration event exploits the standard Application Mobility Service (AMS) API (ETSI, 2022b) defined by ETSI, which currently supports app migrations in environments encompassing multiple edge nodes. Nonetheless, the proposed design in (Feraudo et al., 2023) requires MEC components to support intra-host migration, e.g., MEC applications mobility from remote to local resource infrastructure. In our simulation platform, we decided to deploy a MEC-assisted application mobility for intra-host migrations, which implies MEC components to trigger the user-context transfer. Thus, once the VIM receives a notification involving nodes leaving the resource pool (step (3) Fig. 2b), it looks for MEC applications running on that host, and creates a migration event for each of them. This series of events serve as inputs for an extended version of the AMS, which creates an event chain capable of initiating the intra-host migration of the MEC applications.

3.2 Simulation Tool: Modules

To enable vehicular resources to host MEC applications in a simulated environment, we built each of the MEC entities as an application, abstracting their workload from the underlying host.

Figure 4 illustrates the deployment of simulation modules that replicate the scenario presented in our proposal (Fig. 1). Our simulation tool considers cars as devices capable of hosting MEC applications. To support this functionality, we introduce the car module, which extends the New Radio User Equipment (NRUE) defined in Simu5G. The car module acts as a wrapper for any 5G-enabled device, providing computational resources (e.g., CPU, RAM, and storage) and running applications that enable the car local resource infrastructure to host MEC-compliant applications. In addition, the car module forwards management messages to prepare and configure the local virtualization infrastructure received from the MEC-H management entities. Specifically, the module runs the ClientResApp, which requests and decides whether to accept rewards and handles resource registration and release (section 3.1). Moreover, it executes the VIApp that applies management instructions received from the MEC-H, to perform local resource allocation and release. It should be noted that the VIApp module can properly carry out its functionalities on any host having a resource infrastructure, thus it might be utilized to allow any 5G-enabled device to become part of the MEC-H resource pool.

With the MEC-H now responsible for managing both local and remote resources, the enhanced VIM takes on the critical role of scheduling, preparing, and releasing both local and remote resources. To support this functionality, it sets up the remote virtual infrastructure (VI) to handle remote commands for allocating, relocating, and terminating MEC applications. Moreover, the VIM supports interchangeable scheduling algorithms to optimize remote host selection for application deployment based on the environment.

According to our MEC extended architecture (Feraudo et al., 2023), vehicles may leave the resource pool while running MEC applications. When a vehicle hosting MEC applications exits a parking lot in our simulation scenario, it triggers the migration procedure to prevent service interruption and delay. This highlights the importance of designing a reliable migration mechanism for MEC applications in vehicular computing environments. Our framework supplies a module representing an extended version of the AMS that transparently addresses resource volatility issues. As mentioned in section 3.1, our tool supports MEC-assisted intra-host migration. Thus, when an application instance is relocated, e.g., from a remote

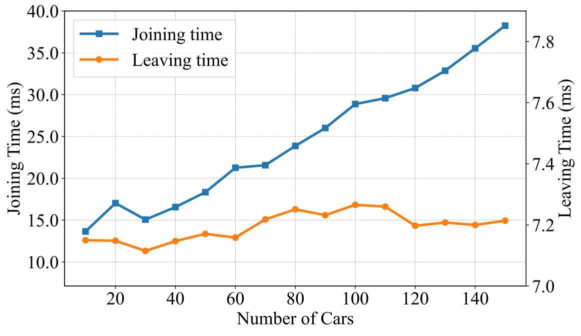


Figure 5: Resource acquisition and releasing protocol times.

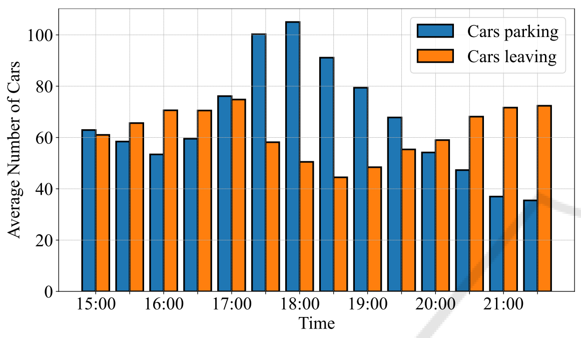


Figure 6: Ceentral Garage City of Arnhem entering and leaving vehicles.

to a local resource infrastructure, the module requires the MEC application to transfer the user context associated with the NRUE device it is serving.

4 PERFORMANCE EVALUATION

In this section, we report about some relevant performance indicators measured on top of our simulation platform for some examples of vehicular cloud computing applications. Additionally, we show how our simulation platform can be utilized to define and evaluate an algorithm that efficiently distributes MEC applications on stationary vehicular resources.

Our experiments were conducted in a 5G standalone network environment with a numerology index $\mu = 2$. The network consists of a single MEC-H connected to a gNB, with the scenario taking place in a parking area situated in close proximity to the gNB. We also implemented a basic reward scheme for the resource acquisition procedure, which utilizes integer values accepted by all participating vehicles. The experiments were carried out on a Linux Virtual Machine running OMNeT++ having 16 CPUs and 64Gb of RAM.

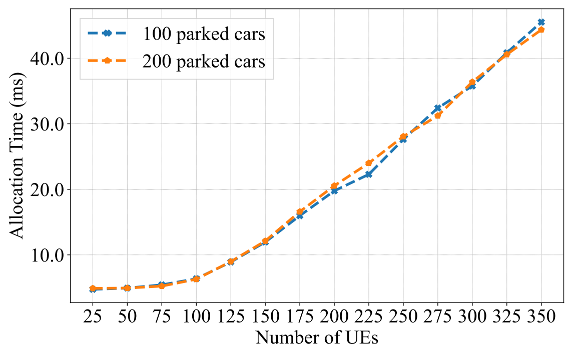


Figure 7: Resource Allocation Time.

4.1 Resource Management

As described in section 3.1, resource management involves the operations for acquiring, allocating, and releasing remote resources.

Figure 5 illustrates the time required by the protocols for collecting and releasing resources from vehicles as they enter or leave the parking lot within the MEC-H AoI. The join time shown in the figure indicates the time interval for the MEC-H to recognize the availability of a new vehicle for MEC application allocation (step (1)-(6) in Figure 2a). On the other hand, the release time is the interval required by the MEC-H to remove the vehicle from the resource pool (steps (1)-(4) in Figure 2b). The figure indicates that the join time follows a growing trend ranging from 13 to 40 ms as the number of cars participating in the resource acquisition procedure increases, whereas the release time remains relatively constant (around 7 ms). The difference in performance between the join and release times can be attributed to the varying number of request/response messages generated by the two protocols. On the one hand, the resource release process necessitates only a few messages to exclude a vehicle effectively from the pool. On the other hand, the resource acquisition process involves a series of request/response messages because the device-initiated reward scheme mandates that the vehicle request available rewards. However, it is unlikely for a large number of vehicles to enter a parking lot simultaneously. Such a scenario may only occur during special events like festivals or football matches. To support this claim, we analyzed the data of three parking garages in the city of Arnhem, which is available on the Open Parkeerdata portal[1]. Figure 6 depicts the average number of cars entering and leaving the most used garage during rush hours. The figure clearly shows that the number of parked vehicles reaches al-

[1]https://parkeerdata.nl/opendata/arnhem/parkeergarages/transactiedata-parkeergarages

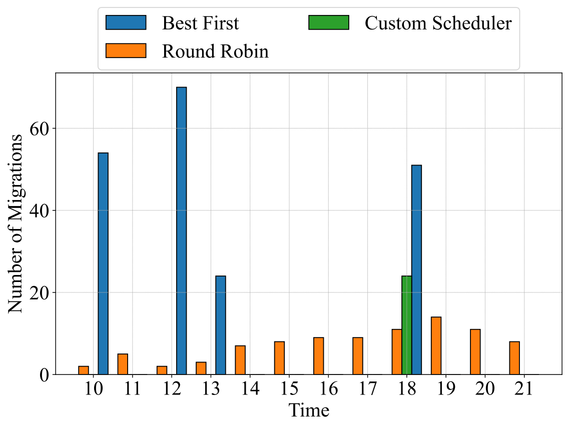


Figure 8: Comparison Scheduling Algorithms in terms of Migrations.

most the maximum considered in our test setup between 17:30 and 18:30. Moreover, it is important to note that the peak of participating vehicles does not necessarily occur simultaneously because data were sampled with 30-minute periodicity.

To analyze the time required to allocate MEC applications on remote resources, we measured the delay introduced by the interactions between the VI and VIM during steps (7)-(8) of the process in Fig. 3. The simulation involves multiple UEs requesting MEC app execution and several parked cars belonging to the MEC-H resource pool. MEC applications are evenly distributed on remote nodes using a Round Robin scheduler. We repeated the simulation 10 times with varying numbers of UEs and parked cars.

Figure 7 illustrates that the delays associated with resource allocation follow an exponential growth that depends on the number of MEC applications deployed on parked cars. This is confirmed by the overlapping curves, which indicate that the delay values remain relatively constant even when the number of parked cars varies. It is worth mentioning that we simulated the worst-case scenario, in which all UEs requested MEC app execution simultaneously, thus leading to a substantial increase in network traffic. Despite this, the delay caused by these interactions remained negligible, even when the number of requests exceeded 300, with a delay of approximately 40ms.

4.2 A Custom Scheduler for Stationary Vehicular Resources

We have developed and tested a custom scheduling algorithm using our simulation platform to demonstrate how it can aid researchers in designing, evaluating, and assessing new algorithms and protocols. It aims at minimizing the number of migrations generated by vehicles leaving the parking lot while running applications.

To generate vehicle and user behaviors, we recreated the scenario utilized in (Feraudo et al., 2023). This approach involves constructing a series of Poisson and Gaussian distributions using two real-world datasets. The Arnhem dataset, already presented in the previous section, was used to model the distributions describing vehicle entry and residency times in a parking lot. The Bologna WiFi dataset[2] provided information on user activities on Open WiFi networks within the city of Bologna, which enabled the creation of distributions mimicking the user behavior during each hour of the day. As in our previous work, we assume that each vehicle that enters the parking lot accepts the rewards proposed by the MEC-H, and each user requests triggers the execution of a one-to-one MEC application.

We simulated a 24-hour period of vehicle and user activities, taking into account a parking lot capacity of 150 vehicles. We run three simulations, one for each scheduling algorithm, namely best first, round robin, and our custom algorithm. The performance of these algorithms was evaluated based on the number of migrations they generated, as this directly impacts the reliability of MEC applications. In other words, a lower number of migrations is desirable for improved performance. For the sake of clarity, we reported the results associated with rush hours.

Figure 8 reports the associated performance results, by referring to the most challenging case of the day hours with highest levels of user and vehicle activity. The best first algorithm chooses the first available vehicle from the pool that has sufficient resources to execute the application. This approach can lead to a large number of migrations, as the selected vehicles may leave the parking lot while running all the applications they are capable of executing. In fact, the number of migrations exceeds 60 at the 12th hour of the simulation. Conversely, the round-robin algorithm maintains a steady number of migrations (around 7.42 in average) as the applications are equally distributed on the vehicles belonging to the MEC-H resource pool. In addition, by using our simulation platform, we have developed a custom scheduler that relies on multiple Gaussian distributions by using means and standard deviation produced after a pre-processing phase of the Arnhem dataset, which generated the average occupancy time based on a 10-minute interval sampling. Hence, for each vehicle that belongs to the resource pool, the custom scheduler utilizes the time at which it joined the pool and the aforemen-

[2]https://opendata.comune.bologna.it/explore/dataset/iperbole-wifi-affluenza/information/

tioned distributions to predict its residency time. It then assigns the MEC application to the vehicle with the highest remaining residency time. The results in the figure demonstrate how this algorithm can largely over-perform the others in the considered application scenario: even if it could be enhanced via more sophisticated machine learning techniques, already in its simple current version it generates only around 20 migrations at the 18th hour of the simulation.

5 CLOSING REMARKS AND FUTURE DIRECTIONS

This paper originally presents a simulation platform capable of assisting researchers in the development, evaluation, and assessment of vehicular application algorithms and protocols that exploit vehicular cloud resources accessed according to the standard ETSI MEC specifications. After presenting our extended MEC architecture for these scenarios, the paper reports about the design and implementation of our original simulation platform, with its resource management modules and procedure interactions. In addition, this paper originally describes a concrete example of how our simulation platform can be utilized to design, test, and implement applications that exploit the vehicular computing paradigm. Despite the simulation platform provided enables researchers to design applications leveraging the vehicular computing paradigm, the current version is limited to supporting stationary vehicles, specifically parking lots. Our future plans include expanding this platform to facilitate the distribution of MEC applications on mobile nodes. Additionally, we intend to explore innovative application scenarios that can harness the benefits of this unique infrastructure, utilizing the simulation platform described in this paper.

REFERENCES

Ahmed, B., Malik, A. W., Hafeez, T., and Ahmed, N. (2019). Services and simulation frameworks for vehicular cloud computing: a contemporary survey. *EURASIP Journal on Wireless Communications and Networking*, 2019:1–21.

Cha, N., Wu, C., Yoshinaga, T., Ji, Y., and Yau, K.-L. A. (2021). Virtual Edge: Exploring Computation Offloading in Collaborative Vehicular Edge Computing. *IEEE Access*, 9:37739–37751.

Elektrobit (2022). Connected vehicle software solutions. https://www.elektrobit.com/products/eb-connected-vehicle/. Last Accessed: 2022-10-03.

ETSI (2022a). GS MEC 003 - V3.1.1. https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_mec003v030101p.pdf.

ETSI (2022b). GS MEC 021 - V2.2.1. https://www.etsi.org/deliver/etsi_gs/MEC/001_099/021/02.02.01_60/gs_mec021v020201p.pdf.

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

Feraudo, A. and Calvio, A. (2023). MEC extension. https://github.com/aferaudo/Simu5G/tree/feat/vim-extension.

Feraudo, A., Calvio, A., Bujari, A., and Bellavista, P. (2023). A novel design for advanced 5g deployment environments with virtualized resources at vehicular and mec nodes. *arXiv preprint arXiv:2303.15836*.

Gerla, M. (2012). Vehicular Cloud Computing. In *Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pages 152–155.

Hou, X., Li, Y., Chen, M., Wu, D., Jin, D., and Chen, S. (2016). Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE Transactions on Vehicular Technology*, 65(6):3860–3873.

Huang, X., Yu, R., Liu, J., and Shu, L. (2018). Parked Vehicle Edge Computing: Exploiting Opportunistic Resources for Distributed Mobile Applications. *IEEE Access*, 6:66649–66663.

Li, C., Wang, S., Huang, X., Li, X., Yu, R., and Zhao, F. (2019). Parked vehicular computing for energy-efficient internet of vehicles: A contract theoretic approach. *IEEE Internet of Things Journal*, 6(4):6079–6088.

Liu, L., Chen, C., Pei, Q., Maharjan, S., and Zhang, Y. (2021). Vehicular edge computing and networking: A survey. *Mobile networks and applications*, 26:1145–1168.

Liu, N., Liu, M., Lou, W., Chen, G., and Cao, J. (2011). Pva in vanets: Stopped cars are not silent. In *2011 Proceedings IEEE INFOCOM*, pages 431–435.

Ma, C. et al. (2021). Parking Edge Computing: Parked-Vehicle-Assisted Task Offloading for Urban VANETs. *IEEE Internet of Things Journal*, 8(11):9344–9358.

Nardini, G., Stea, G., Virdis, A., and Sabella, D. (2020). Simu5g: A system-level simulator for 5g networks. In *Simultech*, pages 68–80.

Olariu, S. (2020). A survey of vehicular cloud research: Trends, applications and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2648–2663.

Olariu, S., Khalil, I., and Abuelela, M. (2011). Taking VANET to the Clouds. *International Journal of Pervasive Computing and Communications*, 7(1):7–21.

Rahman, F. H. et al. (2020). Off-Street Vehicular Fog for Catering Applications in 5G/B5G: A Trust-Based Task Mapping Solution and Open Research Issues. *IEEE Access*, 8:117218–117235.

Sabella, D., Moustafa, H., Kuure, P., Kekki, S., Zhou, Z., Li, A., Thein, C., Fischer, E., Vukovic, I., Cardillo, J., et al. (2017). Toward fully connected vehicles: Edge computing for advanced automotive communications; 5gaa: Munich.