JSIME *Journal of Simulation Engineering*, Volume 4 (2024/2025). Article URL: https://articles.jsime.org/4/1

Simulating and Validating Vehicular Cloud Computing Applications in MEC-enabled 5G Environments

Angelo Feraudo
angelo.feraudo@unibo.it

Alessandro Calvio
alessandro.calvio@unibo.it

Paolo Bellavista ✉
paolo.bellavista@unibo.it

University of Bologna, Bologna, Italy

ACM Subject Categories

- Human-centered computing~Ubiquitous and mobile computing
- Networks~Network performance modeling
- Networks~Network simulations

Keywords

Vehicular Cloud Computing, Multi-access Edge Computing, Vehicular Networks, 5G, OMNeT++

Abstract

The groundbreaking advancements in in-vehicle computing/communication resources and software are granting drivers' access to a diverse range of distributed applications and services. Edge Computing, alongside established frameworks like the European Telecommunications Standards Institute (ETSI) Multi-access Edge Computing (MEC), will play a vital role in these scenarios, by enabling the interoperable and standardized execution of these services at the edge of the network. In addition, Vehicular Cloud Computing (VCC) contributes to expanding computational capacity at the edge by leveraging computing/storage/communication resources offered by vehicles. This synergy holds the potential to forge robust computational infrastructures at the network edge, by favoring several benefits like real-time data processing and minimal latency. However, the research community lacks simulation tools for supporting the testing and validation of applications that exploit both the VCC paradigm and edge-enabled networks at the same time. In this paper, we present our novel simulation tool as a platform for researchers and engineers to design, test, and enhance next-generation distributed applications that exploit the concepts of vehicular, edge, and cloud computing. This simulation tool implements our novel ETSI MEC-compliant architecture, which, in a standard way, supports the leveraging of in-vehicle resources to increase edge computing ones. In addition, the paper reports performance results about the efficiency/scalability of our simulation platform and presents a practical use case where an original algorithm to distribute MEC application components on vehicular resources is validated.

1 Introduction

The rapid advancement of computing technologies and next-generation wireless communication networks has revolutionized in-vehicle hardware, transforming vehicles into powerful mobile computation nodes on wheels (Meng et al., 2023; Lu & Shi, 2023; Olariu, 2020). As a result, drivers can now tap into an extensive array of distributed applications and services, very often cloud-based, as in other vertical domains, including improved access to information and entertainment features. This creates a new ecosystem with unique real-time demands, presenting substantial challenges for the backbone network and cloud infrastructure (Liu et al., 2021). In such a scenario, edge computing has emerged as a solution by providing computational resources closer to the end-users. This provides a more efficient and effective way to handle the ever-increasing demand for computing and storage resources. Moreover, edge computing has the potential to replace traditional cloud solutions by reducing network stress, decreasing latency, improving efficiency, and enabling 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, 2022). This standard enables the execution of contextualized MEC-compliant applications near the data sources and/or users and within a virtualized and multi-tenant environment. Furthermore, the MEC standard facilitates the integration of cloud resources with those available at the edge, creating a complete cloud continuum of virtualized resources distributed within the network.

However, it has been predicted that a connected vehicle will transmit to cloud-based services 1 to 10

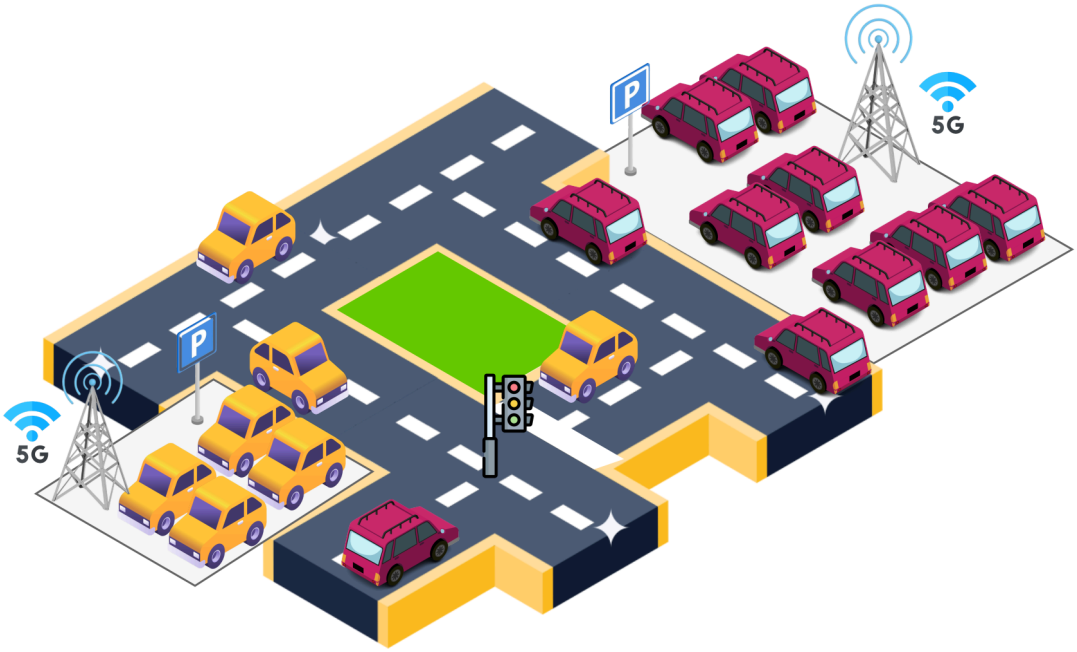


Figure 1. A smart city setting with unexploited vehicle resources.

exabytes of data traffic per month by 2025 (Automotive Edge Computing Consortium, 2021). Therefore, with the escalating number of connected vehicles, there is an anticipated strain on the network in the upcoming years. This surge in data transmission poses challenges in speed and bandwidth, impeding the ability to meet the latency-sensitive requirements of some applications in this emerging execution ecosystem, even when the infrastructure is supported by edge resources. To address this issue, Vehicular Cloud Computing (VCC), also known as Vehicular Computing, has emerged as a promising paradigm to support distributed applications designed according to the edge cloud approach (Lu & Shi, 2023; Bitam et al., 2015; Gerla, 2012; Olariu et al., 2011). VCC leverages the computing/storage/communication resources available on vehicles to create cost-effective mobile clouds at the far-edge. These dynamic clouds can be formed autonomously by vehicles, by exploiting Vehicular Ad-Hoc Networks (VANETs). These networks facilitate the exchange of information necessary for vehicles to share their resources among themselves and/or with nearby edge nodes to extend their virtualized resources for service execution.

To support the deployment of software components across different vehicle computer variants, the traditional embedded platform of vehicles needs to be replaced with a software-defined architecture (Lu & Shi, 2023). This architecture allows vehicles to support cloud-native technologies and receive over-the-air (OTA) updates throughout their life-cycle. Leading software companies are collaborating closely with automotive developers to expedite the widespread implementation of this advanced architecture. One notable solution in this direction is the ARM's Scalable Open Architecture for Embedded Edge (SOAFEE) project. Similarly, the challenge persists when discussing a standardized model for supporting the integration of these resources across the cloud continuum spectrum. Several frameworks and models (Lu & Shi, 2023; Olariu, 2020) proposed to incorporate/exploit resources included in software-defined vehicular networks into/with the cloud infrastructure, ensuring seamless connectivity and resource utilization. Furthermore, since practical experiments on vehicular network environments are expensive and challenging, some recent studies have focused on providing simulation frameworks (Ahmed et al., 2019). However, most of the existing models and frameworks proposed neglect and make no consideration of the challenges arising in multi-vendor and multi-domain environments (Lu & Shi, 2023; Bitam et al., 2015). Moreover, they mostly present algorithmic approaches for assessing the likelihood of vehicles completing task execution. These approaches typically do not address migration when a vehicle exits the cloud during task execution. Lastly, to the best of our knowledge, there is currently no simulation tool providing a unified platform for vehicular computing, enabling researchers to design and test their algorithms.

This paper originally provides an extensive description and reports about the extensions that we recently implemented to the simulation platform firstly presented in Feraudo et al. (2023). The platform relies

on the OMNeT++ network simulator as the underlying framework and incorporates the Simu5G library to model the 5G network and communications aspects. It provides an environment where researchers and engineers can explore, test, and design innovative solutions using the vehicular computing paradigm, by leveraging previously unused resources of stationary (e.g., parked) vehicles (Figure 1). The simulation platform implements an improved 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). In such a context, the resources provided by the vehicles are registered in the edge resource pool and can be accessed through standardized interfaces. The modeled architecture allows dealing with some of the primary challenges arising in vehicular computing environments, such as the integration with cloud resources and enabling the coexistence of heterogeneous technologies. Furthermore, it allows dealing with resource volatility issues (i.e., nodes that dynamically join/leave during service provisioning) via a standard-compliant migration mechanism that we have originally integrated.

In addition, the paper provides a comprehensive description of guidelines and technical insights about our implementation deployment, by highlighting the essential interactions for supporting dynamic resource management, vehicle mobility, and application relocation. Moreover, it carefully identifies the contributions of our solution with regards to the existing state-of-the-art in the related literature. Our simulation framework is openly accessible to researchers through the GitHub repository[1].

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 includes an overview of the related research. Section 4 provides a comprehensive overview of the architecture behind the simulation platform. Additionally, it 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 5. This section also identifies the contributions of the solution proposed with respect to the existing model proposals within the state-of-the-art. Lastly, Section 6 concludes the findings of this work and presents future directions.

2 Background

2.1 ETSI Multi-access Edge Computing

The European Telecommunications Standards Institute (ETSI) proposed the MEC standard to meet the need for a virtualized and multi-tenant environment at the edge of the network. It allows the execution of so-called MEC Applications (MEC-App) compliant with the specification. Figure 2 depicts all the main functional components encompassed within the reference architecture which can be divided into two main parts: the system and the host levels. Each component is linked to the others via reference points allowing the exchange of standardized information, i.e., management (Mm), external (Mx), and MEC Platform-related (Mp).

The host level of the architecture comprises the main functional elements in charge of managing the virtualized environment and its resources (i.e., storage, computing, and network resources). Furthermore, these components implement the real mechanism used to instantiate, delete, and control the actual MEC-Apps running on top of the infrastructure. Going more into the details of each element, the Virtualisation Infrastructure Manager (VIM) is responsible for managing the virtualized resources of the underlying Virtualisation Infrastructure (VI) of the MEC node. All the operations needed to prepare the infrastructure to run the new MEC-App occur in this component. The MEC-H also offers the possibility, for MEC-Apps, to interact with standard services, i.e. Location Service, Radio Network Information Service, Application Mobility Service (AMS), through the MEC Platform (MEC-P). The platform exposes a service registry that contains the information related to the different endpoints of the services. Finally, the MEC Platform Manager (MEC-PM) acts as an intermediate between the orchestrator and the MEC-H by communicating possible communication events that occur.

For the sake of clarity, we have reported all the standard-related acronyms in Table 1.

2.2 Vehicular Computing Paradigm

The early investigations into the Vehicular Cloud Computing (VCC) paradigm were led by Olariu et al. (2011) and Gerla (2012). Their definition relies on the idea that modern-day vehicles come equipped with powerful on-board computers, ample storage, and an array of sensing devices. Olariu et al. (2011) defined 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 Gerla (2012), Vehicular Computing keeps the information gathered by vehicle sensors locally and share it solely with other vehicles, as the sheer volume of in-vehicle generated data can pose serious technical challenges for the network infrastructure. Indeed, with the proliferation of sensors in vehicles (Meng et al.,2023), it is predicted an exponential growth in the data traffic generated by vehicles (Automotive Edge Computing

[1] MEC extension (https://github.com/aferaudo/Simu5G/tree/feat/vim-extension)

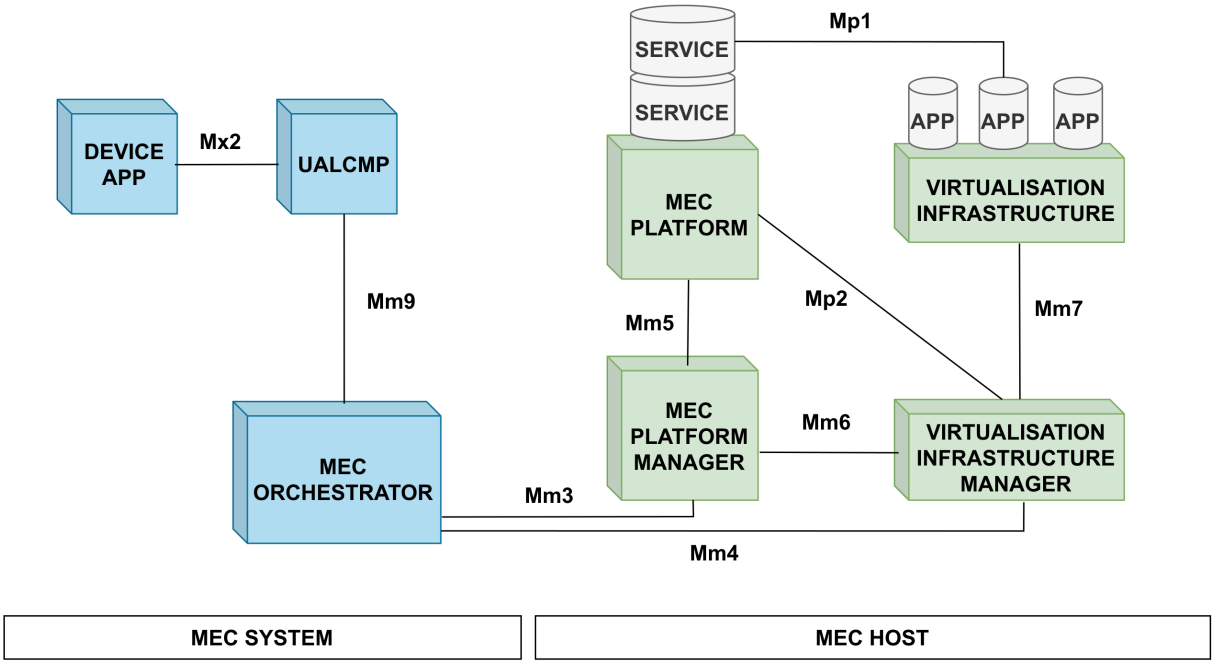


Figure 2. The standard MEC architecture provided by ETSI.

Table 1. Table of acronyms for MEC elements.

Abbreviations	Definition
AMS	Application Mobility Service
D-App	Device App
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	Virtualization Infrastructure
VIM	Virtualization Infrastructure Manager

Consortium ,2021). Therefore, Vehicular Computing proves to be crucial in mitigating network congestion by facilitating data pre-processing directly among groups of vehicles (Gerla, 2012).

As evolution of the VCC paradigm, the paradigm of Vehicular Edge Computing (VEC) has emerged as the integration of vehicular networks with the edge infrastructure. In this sense, VEC facilitates the bringing of computational resources closer to vehicle users allowing the latency times and improving availability of applications via Vehicle-to-Infrastructure (V2I) communications. In a tentative to contribute to the advancement of VEC state-of-the-art, we introduced a novel ETSI MEC-compliant architecture in our recent work (Feraudo et al., 2023), outlined in Section 4.1. The architecture expands the edge resource pool by leveraging vehicular computational resources and forms the foundation of the simulation tool presented in this paper.

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 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.

3 Related Work

To strengthen resource availability at the network edge, various studies have suggested harnessing the underutilized computational power of both stationary and mobile vehicles (Lu & Shi, 2023; Olariu, 2020; Ahmed et al., 2019). These opportunistic resources can be utilized for diverse purposes to handle the growing number of applications used in vehicular networks. For instance, vehicles can function as relay nodes (Liu et al., 2011; Qin et al., 2022) to enhance network connectivity or as computing nodes (Feng et al., 2017; Huang et al., 2018; Rahman et al., 2020; Dressler et al., 2014; Ma et al., 2021) to reduce the impact of these applications on edge node performance. Rajput et al. (2023) introduced the Vehicular Static Cloud-VANET framework, a unified infrastructure that integrates resources from parked vehicles—spanning communication and computing—into the cloud data center. Similarly, Li et al. (2019) formulated an energy-efficient model for parked vehicular computing, coupled with an incentive mechanism. This mechanism aims to motivate vehicle owners to actively participate in cloud formation by contributing with vehicle on-board resources. Dressler et al. (2014) suggested leveraging resources within parked vehicles as shared storage for storing and retrieving location-based data on a large scale.

Other works focused on dynamically forming micro-datacenters using in-vehicle resources without infrastructure requirements (Feng et al., 2017; Kamakshi & Shankar Sriram, 2020; Bute et al., 2022). Feng et al. (2017) presented a workflow for the autonomous formation of groups of vehicles, utilizing an algorithm based on ant colony optimization to schedule job distribution. Kamakshi & Shankar Sriram (2020) considered vehicles' relative mobility (i.e., relative speed and distance) to aggregate vehicles in communities, while Bute et al. (2022) designed an algorithm using the fuzzy logic for vehicular cluster formation.

From another perspective, 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 computing paradigm and the MEC standard deployment environment. This paper presents the implementation details of our simulation tool, which offers a platform where researchers and engineers can exploit the features of Vehicular Computing. The tool has been introduced in Feraudo et al. (2023). With respect to (Feraudo et al., 2023), this article reports a more comprehensive description of functionalities, including a detailed overview of the underlining architecture and interactions needed when migrating applications running on vehicles leaving the cluster. Furthermore, this paper highlights where this collocate within the state-of-the-art, by providing a detailed study of the existing solutions in this context.

4 Unveiling Our Cutting-Edge Vehicular Computing Platform

In this section, we start by presenting a comprehensive overview of the ETSI MEC-compliant Vehicular Computing architecture proposed in our previous work (Feraudo et al., 2023). Furthermore, we describe our novel and comprehensive simulation-oriented platform that facilitates the design and testing phases of modern vehicular computing applications in 5G-enabled environments. Our platform significantly extends the OMNeT++ tool, a discrete and event-based network simulator widely used in research communities, and employs the state-of-the-art Simu5G library (Nardini et al., 2020) for 5G communication modeling.

4.1 General Overview

As illustrated in Figure 3, our approach leverages the underutilized resources of vehicles, by integrating them into the MEC system resource pool. In contrast to the traditional ETSI MEC architecture (see Figure 2), the proposed model encompasses mechanisms capable of deploying and distributing applications on MEC-H (local) and vehicular (remote) resources, all while actively addressing concerns related to resource volatility.

The red dashed circle in Figure 3, defined by each MEC-H, represents the Area of Interest (AoI) within which vehicle resources are collected. The AoI may coincide with the coverage of a single base station or multiple base stations. This depends on where the MEC-H is located, i.e., at either the network edge (close to the base station) or the central data network (at aggregation points). In the scenario considered in this manuscript, the AoI coincides with the parking area where the MEC-H is located.

To model the resource acquisition procedure, our extended MEC architecture introduces an external entity at the MEC system level running a Broker, which

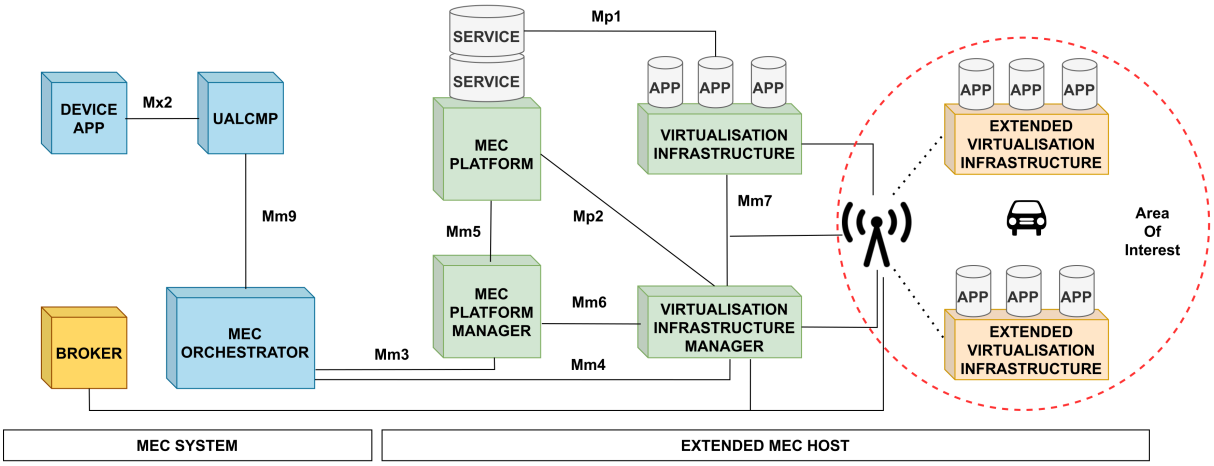


Figure 3. Our extended MEC architecture to leverage vehicle resources.

represents the message broker of a publish-subscribe system. It allows MEC-H subscriptions to the AoI and manages their notification whenever a new vehicle enters or leaves the area. The same entity runs a reward system encouraging vehicles to lease their local virtualized resources, e.g., computing power, and join the resource pool. It relies on a device-initiated scheme requiring mobile nodes to request available rewards contextualized to the AoI. Thus, whenever a new device accepts the MEC-H rewards related to that area, it publishes the amount of resources that it is willing to make available. In addition, once a device leaves the AoI, the MEC-H receives the notification, removes the concerned resources from those available in the pool, and starts the mobility procedure for the apps running on that device. The mobility procedure extends the Application Mobility Service (AMS) API provided by the ETSI standard, enabling seamless intra-host migration from a vehicle leaving the parking area to the local resources of the MEC host. It should be noted that if a device does not communicate its intention to leave the area, our MEC module cannot detect autnomously this new condition. In fact, it is the client of the application running on the leaving device that will initiate a new instantiation once it detects that the app is no longer responsive; this client detection process is automated in our platform and is in line with what specified by the ETSI standard specification (functionality of the Device App running on client devices.

By exploring slightly finer technical details, the VIM is the central MEC-H internal entity to be affected during resource acquisition, as it is in charge of administering the MEC-H resource pool and preparing the VI for the deployment of MEC applications. It operates to handle a heterogeneous pool of distributed resources. Specifically, once registered to the MEC-O, the VIM specifies the content of interest to the Broker corresponding to the AoI parameters (e.g., circle center and diameter) given during its configuration. Thus, when the Broker notifies it of new device resource acquisition, the VIM stores the endpoint information corresponding to an external VI address and the resource capacity of that device. This enables the VIM to monitor and manage the individual contributions of each host in terms of computational resources, effectively handling their volatility.

4.2 Simulation Platform

This section will go into the details of the implementation of our vehicular computing architecture introduced previously, by describing all the modules needed to support both the exploitation of external vehicular resources in MEC-compliant scenarios and the intra-host migration of the MEC applications to guarantee service availability. The scenario supported by our original simulation platform relates to smart city deployment environments with parked and mobile vehicles. The parked vehicles serve as resource providers to the ETSI-compliant infrastructure, while the moving cars act as requesters for the execution of supported applications.

4.2.1 Simulation Modules

In Figure 4 we present the structure and the deployment of our simulation model's main components. From a structural point of view, our models encompass a set of physical machines that host each of the main entities of our extended MEC architecture, developed as applications. In this way, each MEC actor can be abstracted from the machine running it. A set of hosts, the Resource Infrastructure Host, the MEC-PM, and the MEC-P, represent the core of the MEC Host, in charge of managing the life-cycle of MEC Apps and providing MEC services compliant with the specification.

The core of our tools leverages vehicle resources enabling the execution of applications on the far-edge layer. To support this kind of behavior, the simulation model defines the MEC car module, which extends the New Radio User Equipment (NRUE) defined in

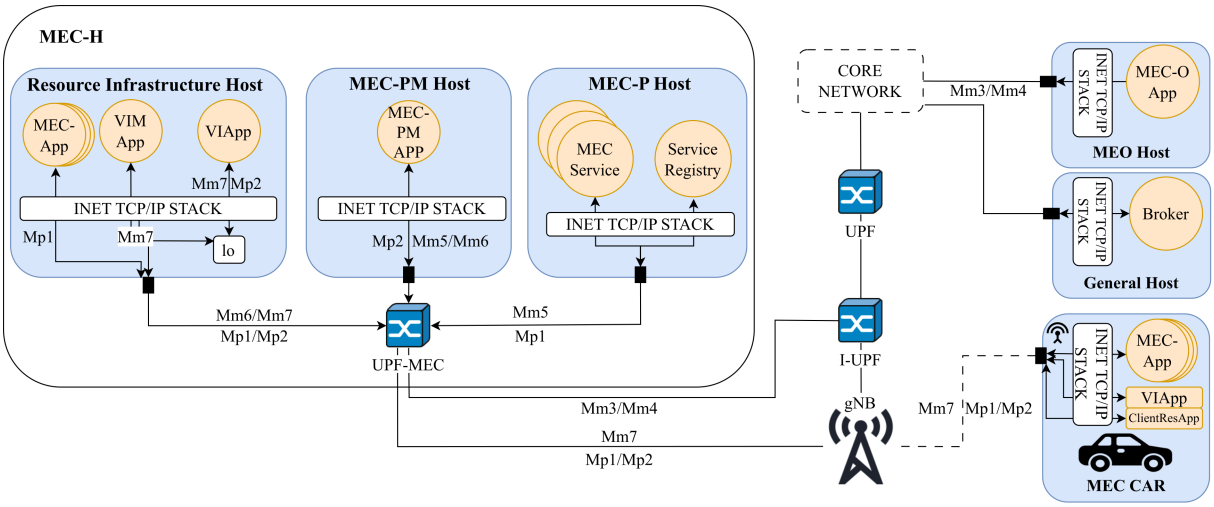


Figure 4. Simulation tool modules structure.

Simu5G. Such a module wraps any 5G-enabled device and provides computational resources (e.g., CPU, RAM, and storage) to run applications orchestrated from the MEC-H. The capabilities of this module are two-fold: on the one hand, the running ClientResApp allows the joining procedures to be initiated to the resource pool of a specific MEC-H, receives the list of rewards, and chooses one of them deciding whether the vehicle can join or not. At the same time, it is also in charge of the resource release procedures when the vehicle leaves the AoI (Section 4). On the other hand, it executes the VI application which manages all the local resources based on the central MEC-H instructions and is in charge of deploying or deleting any MEC App on top of it. The VI module is engineered to seamlessly execute its functions on any host with a resource infrastructure, being able to integrate any 5G-enabled device into the MEC-H resource pool.

In this new extended architecture, MEC-Hs are now able to support both local and remote resources. As a result, the VIM has to deal with scheduling, preparing, and releasing both local and remote resources. The scheduling part happens by means of an extensible system of scheduling algorithms that can be used to choose the best host on which to deploy applications based on some semantics that may favor certain behaviors. After the choice, the interaction with the remote hosts takes place with a set of VI deployed on top of them to handle remote commands for allocating, relocating, and terminating MEC applications.

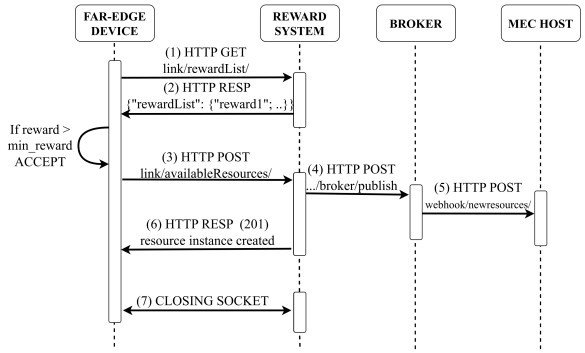
Based on the characteristics of our MEC extended architecture, a vehicle is allowed to leave the resource pool at any moment by causing eventual service disruption for the services it was hosting. To avoid this problem, a migration service is mandatory to guarantee the service availability and avoid delays. Our simulation model deals with this requirement by providing a custom implementation of the AMS service, extended to support MEC-assisted intra-host migrations and address volatility issues in a transparent way. In detail, each MEC App can be relocated from a remote host to the central infrastructure dealing also with the context synchronization between the two applications.

4.2.2 Simulation Interactions: Resource Management

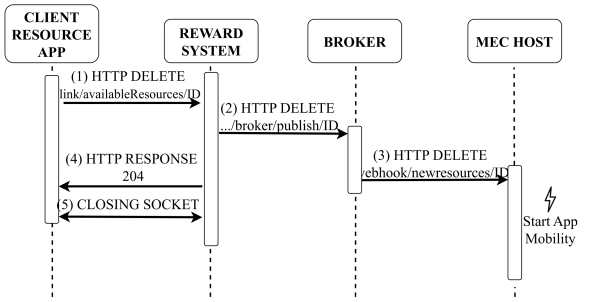
This subsection highlights all the sequence schema related to the allocation and release of dynamic resources by the MEC-H. On a general level, this is mainly done with the introduction of a new component, the Broker, briefly described in Section 4.1. The main role of this actor is to enable the registration and deallocation of vehicle resources present in the AoI by exploiting a system of subscription to ensure that the necessary resources are acquired, released, and allocated.

Resource Acquisition Figure 5(a) describes all the steps required for a vehicle that enters in the AoI to join the MEC-H resource pool. The first phase uses an incentive mechanism consisting of the selection of a reward for participation in the pool. The vehicle receives the list of possible rewards and selects the best one, thus making itself available to surrender its resources (steps (1) - (3)). In the second part, the vehicle's interaction shifts to the broker, who receives information on the availability of the new resources and communicates them to the MEC-H, also providing location details (steps (3) - (6)).

Resource Allocation An example of an interaction scheme for app instantiation on remote resources is reported in Figure 6. The procedure starts with the instantiation request of a new MEC App, done by the Device App and forwarded, through a service chain, from the UALCMP up to the VIM (steps (1) - (6)) where the actual allocation of the app is performed. The VIM manages all the information related to the vehicles participating in the pool and for each of them, it maintains a set of useful information for the schedul-



(a) Resource releasing sequence diagram.



(b) Resource releasing sequence diagram.

Figure 5. Sequence diagram device-initiated scheme.

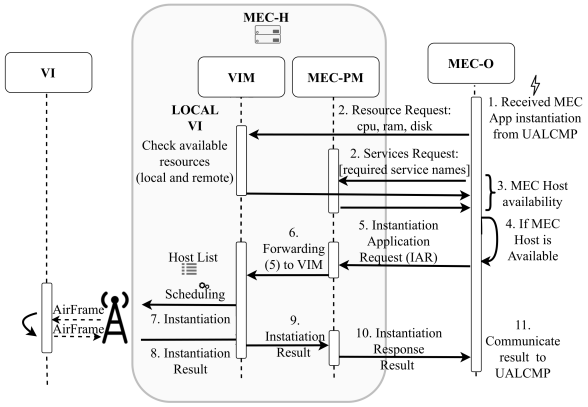


Figure 6. Resource allocation sequence diagram.

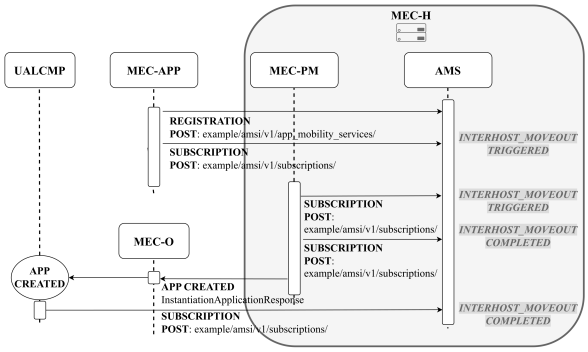


Figure 7. AMS interaction schema - Subscription phase.

ing phase. Depending on the chosen algorithm, the app requirements, and the resource contribution of each host, the VIM ends up with the selection of a single remote host where to deploy the application (step (7)).

Resource Releasing Figure 5(b) 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 mobility procedure for the apps running on that device. Occasionally, a leaving vehicle can have several applications running on top of it. In this case, the resource releasing is also the moment in which a migration procedure is triggered for each of them, as described in Section 4.2.3.

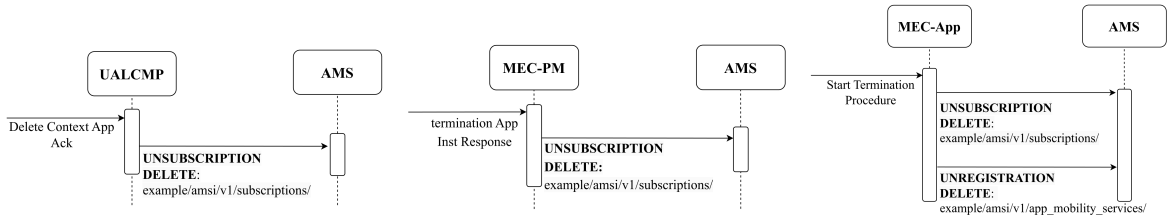
4.2.3 Simulation Interactions: Application Mobility Service

This subsection provides all the interaction schema related to our implementation of the AMS, compliant with the specification. This version of AMS supports the intra-host migration for applications running on a host leaving the resource pool, helping to preserve the service availability in a transparent way with respect to the client actors. This is achieved through the implementation of a subscriptions and notifications system that helps in involving all the modules interested in specific migration events. Each procedure handles a different aspect of the procedure highlighting also the operations that MEC Apps need to fulfill in order to support migration scenarios.

Subscription The initial prerequisite for facilitating migration is to establish the systemic recognition of each new MEC-App by the AMS. Figure 7 delineates the comprehensive operations executed by all involved modules upon the instantiation of a novel MEC-App.

At the startup time, the MEC-App registers to the AMS service receiving a registration ID. Afterward, it subscribes to the event of `INTERHOST_MOVEOUT_TRIGGERED` to be informed about the creation of migrated copies of the app and start the necessary context transfer procedure.

The instantiation of the MEC-App also has backward effects on the actors involved in the creation process. The MEC-PM subscribes to two different events which are respectively the `INTERHOST_MOVEOUT_TRIGGERED` and `INTER-`



(a) UALCMP unsubscription.

(b) MECPM unsubscription.

(c) MEC App unsubscription and unregistration.

Figure 8. AMS interaction schema - Unsubscription phase.

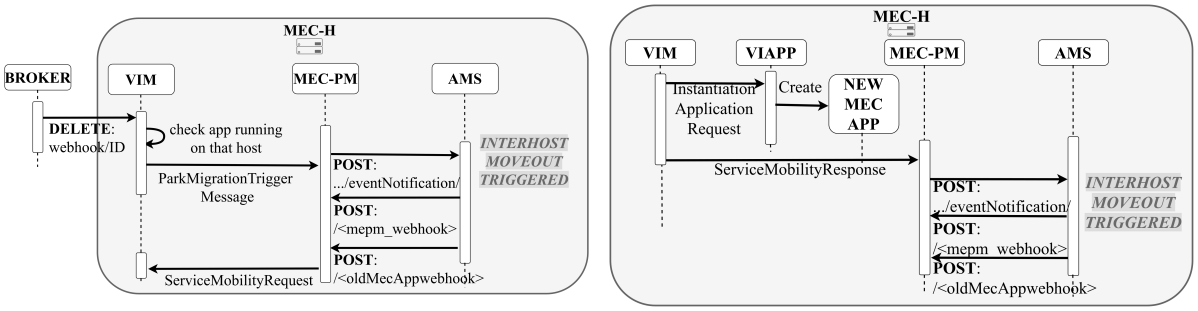
`HOST_MOVEOUT_COMPLETED` to be notified about the leaving of the host from the resource pool and the completion of the migration procedure. These notifications are necessary to start the migration procedure and update the app information. Finally, the UALCMP subscribes for the `INTERHOST_MOVEOUT_COMPLETED` event to get all the information about the new migrated app and give the information about the new location back to the User Application.

Unsubscription In opposition to the previous schema, the deletion or the migration of a MEC-App causes the unsubscription from the related events by all the actors involved (Figure 8). Thus, the UALCMP, the MEC-PM, and the interested MEC-App unsubscribe from all the `INTERHOST_MOVEOUT` events. Next, the app is removed from the system with the proper unregistration.

Context trigger Figure 9 (steps A and B) depicts all the interactions that occur when the migration process is initiated. This trigger is the starting point to enable the creation of a copy of the migrating MEC-App on the central MEC-H infrastructure. However, the procedure holds the potential for future enhancements. It enables the possibility of selecting an alternative host within the resource pool, according to the supported scheduling algorithms.

Whenever a host participating in the resource pool leaves the AoI, the broker notifies the information to the VIM to start the resource-releasing procedure (Figure 5(b)). At the same time, the notification represents also the instant in which the VIM checks if there is any MEC-App running on the leaving host. In such a case, the VIM triggers the migration for any of them. The `ParkMigrationTrigger` is sent to the MEC-PM which forwards the notification `INTERHOST_MOVEOUT_TRIGGERED` up to the AMS. Afterward, the AMS is in charge of notifying all the modules that have an active subscription to the TRIGGERED event. In this phase, the MEC-PM is the main actor that receives a notification and sends the `ServiceMobilityRequest` message to the VIM which leverages its local VI App to create a copy of the migrating app. This message serves to request the VIM to move MEC-Apps from one host to another. At the end (Figure 9(b)), the MEC-PM receives the information about the successful instantiation and emits a new notification for the `INTERHOST_MOVEOUT_TRIGGERED`. The main difference with the previous one is in the body of the notification, which contains the information about the location of the new app. Hence, this information can be available at the system level for all the interested actors.

Context transfer The context transfer procedure aims to synchronize the state of the two MEC apps — both the migrating and the migrated. This ensures that the new MEC is able to continue delivering the service to client apps with less disruptions. At the startup, the new MEC-App executes the operation of subscription and registration and listens on a socket until the proper operational state is injected. This step is triggered by the `INTERHOST_MOVEOUT_TRIGGERED` notification on the migrating app, which contains all the information related to the location of the new service (Figure 10). Using this data, the migrating MEC-App establishes a communication channel with the newly created MEC-App, through which it transmits the information related to the service's state. Once the new MEC-App has been initialized with the new state, it forwards this information to the AMS service, which in turn notifies the UALCMP about the new MEC-App location. Specifically, the last step consists of updating the information related to the serving service on the UE App side. This is achieved with a backward process where the new location is propagated through the Device App up to the UE App.



(a) Step A.

(b) Step B.

Figure 9. AMS interaction schema - Context trigger.

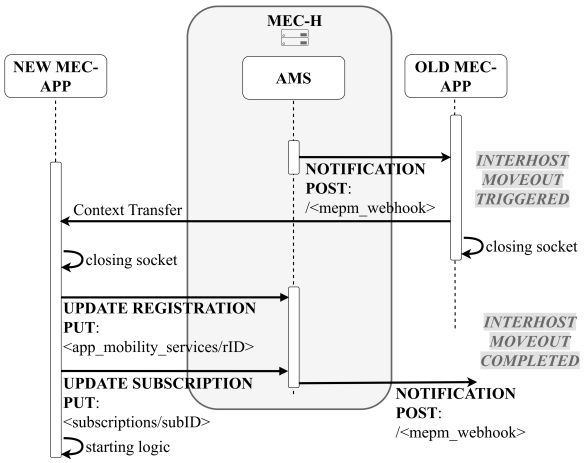


Figure 10. AMS interaction schema - Context transfer.

5 Performance Evaluation

In this section, we report on some relevant performance indicators measured on top of our simulation platform, described previously, 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 $\mu = 2$. As illustrated in Figure 11, the network consists of two MEC-Hs (*mechost1* and *mechost2*) each connected to a gNB, with the scenario taking place in two parking areas situated in close proximity to the gNBs. We also implemented a basic reward scheme for the resource acquisition procedure, which utilizes integer values accepted by all participating vehicles. For the sake of clarity, the evaluations outlined in this paper consider only performance related to one MEC-H. The experiments were carried out on a Linux Virtual Machine running OMNeT++ having 16 CPUs and 64Gb of RAM.

5.1 Resource Management

As described in Section 4, resource management involves the operations for acquiring, allocating, and releasing remote resources.

Figure 12(a) 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-Apps allocation (step (1) - (6) in Figure 5(a)). 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 5(b)). 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 requests 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 Parkeer data portal[2]. Figure 12(b)Figure depicts the average number of cars entering and leaving the most used garage during rush hours. The figure clearly shows

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

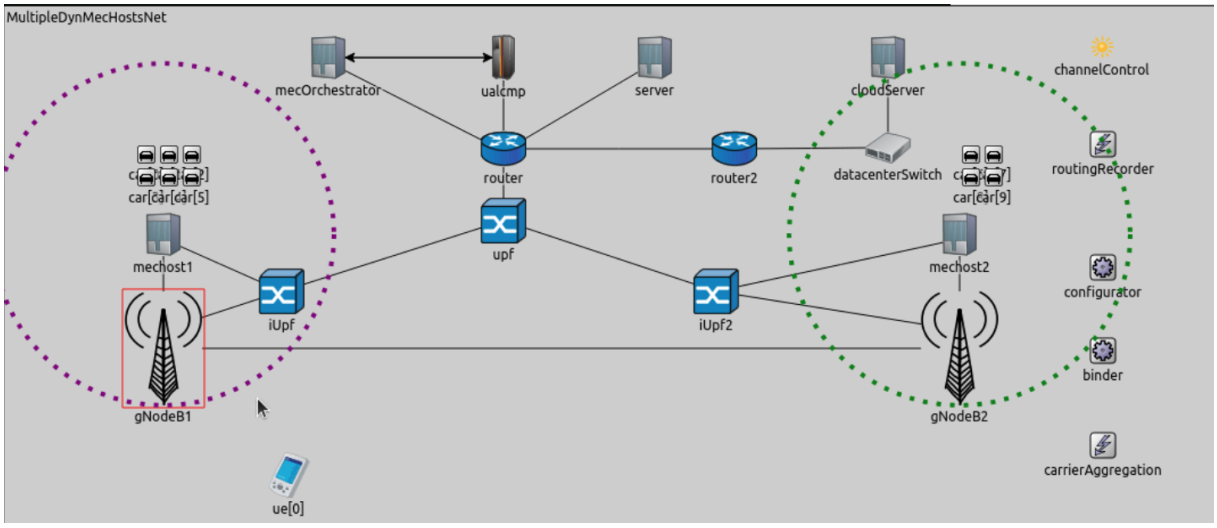
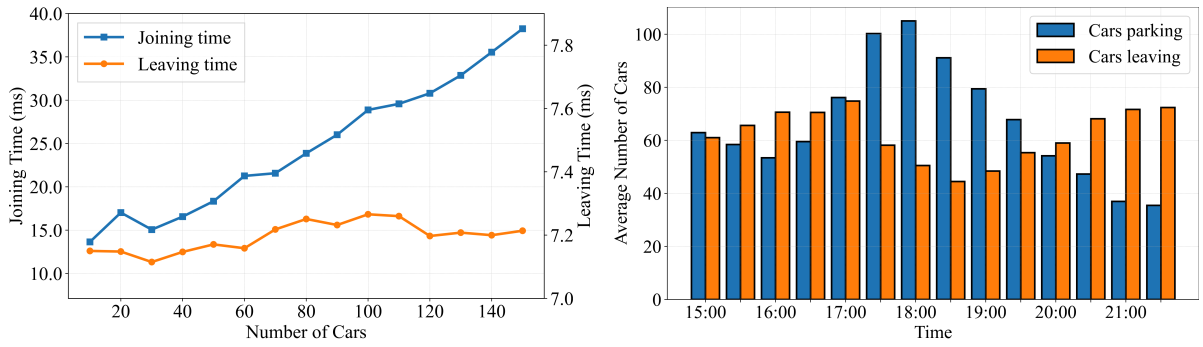


Figure 11. OMNeT++ simulation environment.



(a) Resource acquisition and releasing protocol times.

(b) Ceentral Garage City of Arnhem entering and leaving vehicles.

Figure 12. Resources management times and vehicle distribution in a parking lot.

that the number of parked vehicles reaches almost 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 Figure 6. The simulation involves multiple UEs requesting MEC app execution and several parked cars belonging to the MEC-H resource pool. MEC-Apps 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 13 illustrates that the delays associated with resource allocation follow an exponential growth that depends on the number of MEC applications deployed

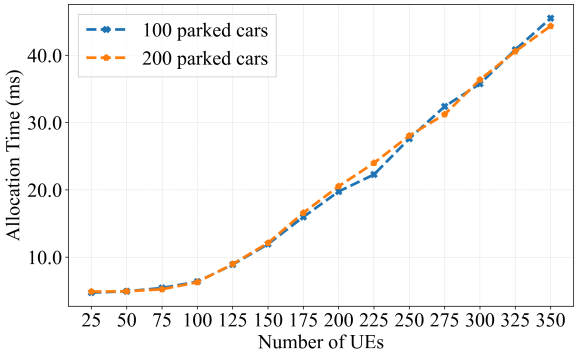


Figure 13. Resource allocation time.

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.

5.2 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[3] 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 request triggers the execution of a one-to-one MEC-Apps.

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 14 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.

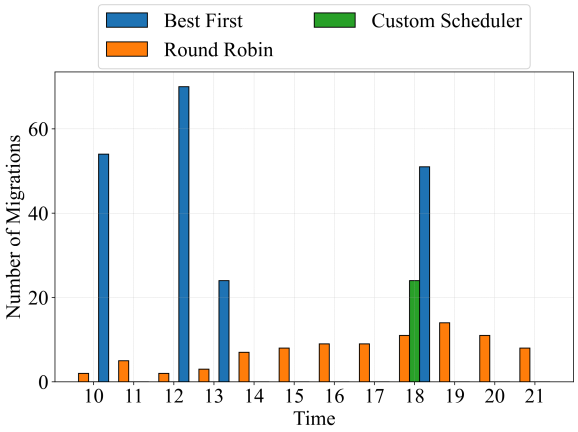


Figure 14. Comparison scheduling algorithms in terms of migrations.

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 aforementioned 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.

5.3 A Comparative Study of Existing Solutions

To provide a better understanding of the contributions made in this work, we summarized in Table 2 the key characteristics of existing solutions that provide a Vehicular Computing model. The table highlights the communication model under consideration during experimentation, how the solution handles the high dynamism inherent in the VANET environment, resources used to create the dynamic computing nodes, i.e., mobile or stationary, adherence to a standard for edge solutions, the provision of a open source platform to design and test other proposals, and lastly, whether infrastructure support is requisite for the vehicular cloud formation.

Rajput et al. (2023) proposed the Vehicular Static Cloud-VANET model, which aims to establish a cohesive infrastructure capable of hosting Intelligent

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

Table 2. Difference between existing solutions and our scheme.

Work	Comm. Model	Mobility Control	Resources	Standard Compliance	Open Source	Infrastructure Support
Our	5G SA and model 1	AMS API	Relatively Static	✓	✓	Needed
(Rajput et al., 2023)	802.11p	N.A.	Stationary	X	X	Needed
(Li et al., 2019)	N.A.	N.A.	Stationary	X	X	Needed
(Fan et al., 2022)	N.A.	N.A.	Stationary and Moving	X	X	Needed
(Dressler et al., 2014)	802.11p	N.A.	Stationary	X	X	Not Needed
(Cha et al., 2021)	802.11p	N.A.	Stationary and Moving	X	X	Not Needed
(Feng et al., 2017)	802.11p	N.A.	Stationary and Moving	X	X	Not Needed
(Bute et al., 2022)	LTE mode 4	N.A.	Moving	X	X	Not Needed

Transportation System (ITS) services. The model employs a centralized controller to integrate resources provided by parked vehicle, equipped with IEEE 802.11p enabled On-Board Unit (OBU), with those provided by the cloud, and receives request from on-road vehicles. To assign a job to a vehicle, the authors use the vehicle residency time, i.e., the time for which a vehicle remains part of the vehicular cloud. This allows them to avoid to worry about the migration of the task, which occurs when a vehicle leave the parking lot. Similarly, Li et al. (2019) considers the residency time for task allocation on parked vehicles, through the definition of predefined contracts. Using these contracts, the authors define the incentive used to motivate vehicle owners in providing in-vehicle resources. In the same direction, Fan et al. (2022) proposed a joint task offloading and resource allocation strategy for cloud of vehicles covered by a base station.

Dressler et al. (2014) extended an existing routing algorithm to allow the dynamic creation of a distributed storage and improvement in network connectivity without the need of RSUs, by exploiting resources provided by vehicle as they enter and leave a parking lot. The authors conducted extensive simulations using OMNet++ and SUMO (Lopez et al., 2018). Cha et al. (2021) presented an algorithm designed for the autonomous formation of vehicle clouds, leveraging a predictive metric termed "companion time". This metric delineates the duration for which vehicles maintain proximity to each other, facilitating the creation of a cohesive cloud of vehicles not only in parking scenarios but also when they are moving slowly. Similarly, Feng et al. (2017) proposed AVE a framework that exploits beaconing to dynamically form cloud of vehicles. The authors also provide a scheduling algorithm for job assignment based on the ant colony optimization problem. Bute et al. (2019) explore a scenario involving vehicles dynamically forming clusters on a highway to establish a collaborative computing platform with the cloud infrastructure. The selection of the cluster head is accomplished through the application of a fuzzy logic algorithm, utilizing metrics that ensure stable connectivity between nodes, thereby facilitating reliable communication.

The solutions proposed by the above authors do not refer to a standard for the integration of vehicular resources in the cloud-continuum spectrum. Indeed, they mostly define a new architecture or workflow to be used for in-vehicle resources acquisition. As outlined in Table 2, *Mobility Control* column, most of the works propose algorithmic strategies used to evaluate the probability of nodes to complete task execution, thus, they do not perform migration in case of a node leaving the cloud while executing tasks. Ge et al. (2020) formulated the service migration process for parked vehicle scenario, as vehicles leave the parking lot or refuse to continue providing the service. However, the solution focuses only on providing a mathematical formulation migration problem, by neglecting other challenges introduced in Vehicular Computing real-world scenarios. These encompass procedures for vehicle resources join and leave procedures, incentive

mechanism to motivate vehicle owners, and the aspect of standardization.

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 computing paradigm and the MEC standard deployment environment.

Furthermore, compared to the other existing solutions, our simulation tool incorporates a MEC-assisted procedure facilitating the migration of tasks allocated to vehicles departing from the parking lot or declining to continue providing services (see Section 4). At present, the tool supports quasi-mobile vehicles, specifically those exiting the confines of the parking lot. Additionally, as our platform adopts ETSI-compliant API for service migration, it becomes versatile enough to extend its support to acquiring resources from moving vehicles.

6 Conclusions and Future Work

This paper provides the readers with a comprehensive overview of our design for an ETSI MEC-compliant architecture, specifically tailored to support the dynamic acquisition of vehicular resources. In addition, it presents, with in-depth design and implementation details, our simulation/oriented platform (Feraudo et al., 2023) that faithfully replicates the proposed MEC/compliant architecture. We claim that our platform empowers researchers to model their solutions, by focusing on scheduling procedures at either edge level or vehicular level, and on the development of innovative applications that exploit the potential of Vehicular Computing. The paper also provides a practical deployment example that illustrates how our simulation framework can be utilized for designing, testing, and implementing vehicular computing applications. Finally, it highlights the novel contributions of our solution if compared with existing related proposals in the state-of-the-art literature. Despite the current version of our simulation framework enables researchers to design applications leveraging the vehicular computing paradigm, it is currently limited to supporting stationary vehicles or slowly-mobile ones, such as the vehicles leaving a parking lot; that priority in our implementation work depended on the envisioned VCC applications that we were willing to support. However, considering the versatility of the model and the implementation of our simulation platform, our future plans include expanding it to support the automated distribution of MEC applications on fully-mobile vehicles. Additionally, we intend to explore other innovative application scenarios, such as decentralized machine learning operations by quasi-autonomous mobile vehicles, based on pre-trained models and successively improved via local refinement learning.

Acknowledgements

A preliminary version of this article was presented at the INSTICC SIMULTECH 2023, Rome, Italy, 12–14 July 2023.

Bibliography

Ahmed, B., Malik, A. W., Hafeez, T., & Ahmed, N. (2019). Services and simulation frameworks for vehicular cloud computing: A contemporary survey. *Journal on Wireless Communications and Networking*, 4(2019).

Automotive Edge Computing Consortium (2021). *Break down the barriers to automotive edge adoption* [White paper].

Bitam, S., Mellouk, A., & Zeadally, S. (2015). Vanet-cloud: A generic cloud computing model for vehicular ad hoc networks. *IEEE Wireless Communications*, 22(1):96–102.

Bute, M. S., Fan, P., Liu, G., Abbas, F., & Ding, Z. (2022). A cluster-based cooperative computation offloading scheme for C-V2X networks. *Ad Hoc Networks*, 132:102862.

Cha, N., Wu, C., Yoshinaga, T., Ji, Y., & Yau, K.-L. A. (2021). Virtual edge: Exploring computation offloading in collaborative vehicular edge computing. *IEEE Access*, 9:37739–37751.

Dressler, F., Handle, P., & Sommer, C. (2014). Towards a vehicular cloud - using parked vehicles as a temporary network and storage infrastructure. In *Proceedings of the 2014 ACM International Workshop on Wireless and Mobile Technologies for Smart Cities* (pp. 11–18), New York, NY:ACM.

ETSI (2022). GS MEC 003 - V3.1.1.

Fan, W., Liu, J., Hua, M., Wu, F., & Liu, Y. (2022). Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles. *IEEE Transactions on Vehicular Technology*, 71(5):5314–5330.

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

Feraudo, A., Calvio, A., & Bellavista, P. (2023a). A novel OMNeT++-based simulation tool for vehicular cloud computing in ETSI MEC-compliant 5G environments. In *Proceedings of the 13th International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (pp. 448–455), Portugal:SciTePress.

Feraudo, A., Calvio, A., Bujari, A., & Bellavista, P. (2023b). A novel design for advanced 5G deployment environments with virtualized resources at vehicular and MEC nodes. *IEEE Vehicular Networking Conference (VNC)* (pp. 97–103), New

York, NY:IEEE.

Ge, S., Cheng, M., He, X., & Zhou, X. (2020). A two-stage service migration algorithm in parked vehicle edge computing for internet of things. *Sensors*, 20(10):2786.

Gerla, M. (2012). Vehicular cloud computing. In *The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)* (pp. 152–155), New York, NY:IEEE.

Huang, X., Yu, R., Liu, J., & Shu, L. (2018). Parked vehicle edge computing: Exploiting opportunistic resources for distributed mobile applications. *IEEE Access*, 6:66649–66663.

Kamakshi, S. & Shankar Sriram, V. (2020). Modularity based mobility aware community detection algorithm for broadcast storm mitigation in VANETs. *Ad Hoc Networks*, 104:102161.

Li, C., Wang, S., Huang, X., Li, X., Yu, R., & 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., & 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., & Cao, J. (2011). PVA in VANETs: Stopped cars are not silent. In *Proceedings IEEE INFOCOM* (pp. 431–435), New York, NY:IEEE.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., & Wiessner, E. (2018). Microscopic traffic simulation using SUMO. In *21st International Conference on Intelligent Transportation Systems (ITSC)* (pp. 2575–2582), Nwq York, NY:IEEE.

Lu, S. & Shi, W. (2023). Vehicle computing: Vision and challenges. *Journal of Information and Intelligence*, 1(1):23–35.

Ma, C., Zhu, J., Liu, M., Zhao, H., Liu, N., Zou, X. (2021). Parking edge computing: Parked-vehicle-assisted task offloading for urban VANETs. *IEEE Internet of Things Journal*, 8(11):9344–9358.

Meng, T., Huang, J., Chew, C.-M., Yang, D., and Zhong, Z. (2023). Configuration and design schemes of environmental sensing and vehicle computing systems for automated driving: A review. *IEEE Sensors Journal*, 23(14):15305–15320.

Meredith, J., Firmin, F., & Pope, M. (2022). *Release 16 description; Summary of Rel-16 Work Items*. HTML

Nardini, G., Stea, G., Virdis, A., and Sabella, D. (2020). Simu5G: A system-level simulator for 5G networks. In *Proceedings of the 10th International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (pp. 68–80), Portugal:SciTePress.

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., & Abuelela, M. (2011). Taking VANET to the clouds. *International Journal of Pervasive Computing and Communications*, 7(1):7–21.

Qin, P., Fu, Y., Feng, X., Zhao, X., Wang, S., and Zhou, Z. (2022). Energy-efficient resource allocation for parked-cars-based cellular-V2V heterogeneous networks. *IEEE Internet of Things Journal*, 9(4):3046–3061.

Rahman, F. H., Newaz, S. H. S., Au, T. W., Suhaili, W. S., Lee, G. M. (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.

Rajput, N. S., Singh, U., Dua, A., Kumar, N., Rodrigues, J. J. P. C., Sisodia, S., Elhoseny, M., and Lakys, Y. (2023). Amalgamating vehicular networks with vehicular clouds, AI, and big data for next-generation its services. *IEEE Transactions on Intelligent Transportation Systems*, 25(1):869–883.

Copyright Information