2025 36th International Teletraffic Congress (ITC 36)

Mobility Aware Task Migration in Vehicular Edge Computing Networks

Rim Sayegh*†, Hela Marouane*, Sahar Hoteit†§, Abdulhalim Dandoush‡
* ESME Research Lab, Ivry sur Seine, France
†Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des Signaux et Systèmes (L2S), Gif-sur-Yvette, France
‡University of Doha for Science and Technology (UDST), Doha, Qatar
§Institut Universitaire de France (IUF), France
rim.sayegh@centralesupelec.fr, hela.marouane@esme.fr, sahar.hoteit@centralesupelec.fr, abdulhalim.dandoush@udst.edu.qa

***Abstract*—With the emergence of autonomous vehicles and the ever-increasing data reported, providing the required latency and computational capabilities is becoming challenging. To address this issue, multi-access edge computing (MEC) for 3GPP 5G Cellular vehicles to everything (C-V2X) has been proposed recently. In this paper, we propose the Mobility Aware Task Migration (MATM) algorithm that strives the limitations of the benchmark algorithms that the MEC orchestrator utilizes for offloading tasks from vehicles. We provide a detailed simulation-based study for End-to-End (E2E) delay by serving the safety application as a function of different network densities. Firstly, we propose an extension to the location service defined in the ETSI MEC reference architecture; this extension enables the MEC service to send a migration notification to the MEC orchestrator to start the migration. Secondly, we introduce an enhancement to the MEC orchestrator module to enable task migration during vehicle mobility by selecting the most suitable MEC host with the closest proximity to the vehicle. Additionally, we use the Simu5G simulator to implement our scenario and conduct an evaluation of task offloading algorithms, and a detailed E2E delay analysis. Simulation results highlight the ability of our proposed algorithm to minimize E2E delay while ensuring fairness in resource allocation.**

***Index Terms*—Multi-Access Edge Computing (MEC); Automated Vehicles (AV); Task Migration, Task offloading, End-to-End (E2E) Delay; 5G Simulation**

I. Introduction

Intelligent Transport System (ITS) represents a complex system that includes infrastructure (e.g., roads, traffic lights), dynamic environmental conditions (e.g., weather and road conditions), and connected vehicles interacting with these elements. Different intelligent transport materials and software are deployed onboard or at a central cloud to enable a new generation of applications and services such as collaborative auto-driving or safety applications (e.g., Intersection Collision Warning Service). Thus, ITS applications are feasible thanks to exchanging predefined data messages via wireless communications and processing them by small computing nodes embedded in the vehicles or by a remote robust server in a distant cloud when a large computing capacity is required. However, certain critical services require relatively large computing and storage resources, which are larger than the capacity of vehicles. Moreover, these services are sensitive to delays, and forwarding the information to a remote central cloud for processing may be inappropriate. To solve this issue, Multi-Access Edge Computing (MEC) [1] provides cloud computing capabilities and an Intelligent Transport (IT) service environment, which is based on Network Function Virtualization (NFV) and Software Defined Networks (SDN) located at the edge of the 5G and beyond (5GB) networks near the end users. 5GB network introduces several innovations, from access to the core network side. The 5G New Radio (NR) improves latency compared to 4G. This aspect makes MEC adoption suitable for supporting vehicular services such as in-vehicle information technology and cooperative driving. The deployment of MEC close to the terminal allows low latency and high service efficiency [2]. In addition, MEC also has excellent computing and storage capabilities [3]. A key research issue of 5GB is how to efficiently use the MEC paradigm to optimize the offered Quality of Services (QoS). The recent development of autonomous driving technologies has increased the demand for services and resources at the network edge. Inappropriate service of autonomous vehicles on the network edge, due to resource allocation and scheduling limitations, can lead to incorrect decisions and potentially serious accidents [4]. Therefore, evaluating MEC performance in 5GB networks is crucial. Moreover, optimizing MEC resource allocation and communication segments for various services is essential to ensure safety and efficiency.

The MEC architecture is structured into two levels: the MEC system level and the MEC host level [5], as shown in Fig. 1. The MEC system level provides a comprehensive overview and management of MEC hosts, enabling efficient lifecycle management of MEC applications, such as instantiation, relocation, and termination, through the MEC orchestrator. This orchestrator interacts with the User Application Lifecycle Management Proxy (UALCMP) to handle application requests and selects appropriate MEC hosts based on specific requirements like latency and resource availability. At the MEC host level, resources, including CPU, RAM, and Disk, are allocated to MEC applications within the virtualization infrastructure. MEC services deployed on the MEC platform are accessed via standard APIs and can be discovered through a service registry. The MEC platform manager and virtualization infrastructure manager (VIM) ensure efficient resource monitoring and management. However, with autonomous vehicles, a new challenge arises because of unpredicted user mobility in the wireless network. With the presence of user mobility [6], enhancing low latency and smoothing user experience is far more than simply pushing the cloud capabilities to the network edge. To enhance the vehicle experience in MEC, the application profiles of vehicles should be dynamically migrated across edge nodes to keep up with their mobility. The challenge of dynamically placing application profiles is not straightforward. User-perceived latency depends on both communication and computing delays. Simply placing each vehicle's application

978-3-948377-03-8/19/$31.00

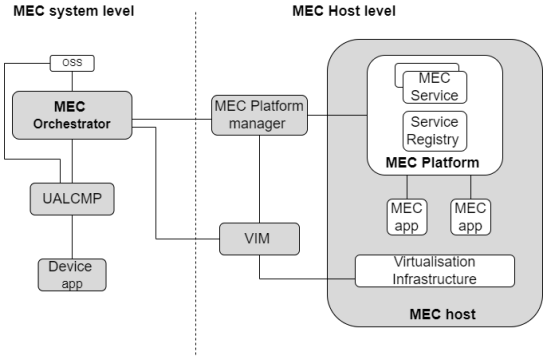


Fig. 1: MEC Architecture

profile at the closest MEC host might seem like a good idea, but it can lead to some nodes becoming overloaded, increasing computing delays. To address this issue, balancing the load among MEC hosts and reducing task response time while keeping the vehicle application close to the vehicle during mobility is crucial. This paper highlights the challenges of achieving efficient task migration while maintaining load balancing between MEC hosts in vehicular edge computing (VEC) environments. The contributions of our paper are:

1) We introduce the Proximity-Based Migration Notification Extension for the MEC Location Service defined in ETSI GS MEC 013 Location API [7] within the Simu5G simulator [8]. The latter enables real MEC applications to interact with 5G transport and services via ETSI-compliant interfaces [5]. Where MEC host includes a UPF module, enabling its placement anywhere in the 5G core network. This extension allows migration to be handled based on vehicle proximity and sends these notifications to the orchestrator. This mechanism allows for faster migration decisions, eliminating the delay caused by vehicle-initiated migration requests and their acknowledgment process.
2) We propose the Mobility-aware Task Migration (MATM) algorithm employed by the MEC orchestrator to identify the best MEC hosts. This algorithm considers the distance between the vehicle and the MEC hosts then selects the nearest one while maintaining fairness in resource allocation.
3) We evaluate the performance of the proposed MATM algorithm through simulation experiments. The results demonstrate the significant potential of our algorithm in reducing the E2E delay compared to benchmark algorithms.

The remainder of the paper is structured as follows. Section II provides an overview of the related works. Section III presents our system model and the details of the simulation scenarios, respectively. Section IV presents our proposed application migration algorithm. Section V discusses the obtained results. Finally, Section VI concludes our work and highlights future directions.

II. Related Work

Mobile Edge Computing (MEC) reduces latency and resource constraints in vehicular networks, but rising data and computational demands make fast service challenging. Optimizing offloading and task migration strategies is essential for maintaining low latency and efficient resource use, especially in dynamic environments. Prior work has primarily focused on optimizing task migration strategies and addressing mobility challenges in dynamic environments. Gkatzikis et al. [9] lay the foundation for task migration in Mobile Cloud Computing (MCC). Their work explores centralized, server-initiated, and task-initiated migration mechanisms, emphasizing trade-offs in complexity, scalability, and system performance. In addition, they highlight critical challenges, including user mobility, workload uncertainty, and multi-tenancy effects, while advocating for mobility-aware migration strategies. The MCC relies on distant cloud servers, leading to higher latency but offering greater computational resources. In contrast, our work focuses on the MEC paradigm, where MEC decentralizes computing resources by placing them at the network edge closer to users, thus reducing latency. Addressing mobility-specific challenges, Taleb et al. [11] introduce the Follow-Me Cloud (FMC) framework, which ensures service continuity via mobility-aware service migration decisions by applying Markov-decision-process based algorithm for cost-effective performance-optimized service migration decisions. These studies mainly focus on offloading tasks to cloud data centers to reduce the computation time. In contrast, offloading tasks to the cloud increases the communication delays which are unsuitable for sensitive applications such as autonomous vehicles. Extending this, Machen et al. [10] propose a layered migration framework tailored for MEC environments, significantly reducing service downtime through container-based solutions. They propose a Generic Layered Migration framework using incremental file synchronization that decreases the migration time compared to the virtual machine technologies. They focus on optimizing the migration process for applications running in MEC hosts; unlike our work, it does not address when migration decisions should be made. By using mobility-aware algorithms and proximity-based triggers, we enhance the decision-making process.
Although edge servers reduce communication delays, their limited resources compared to cloud data centers highlight the growing need for efficient load-balancing techniques to optimize resource utilization. In this context, Zhu et al. [12] focus on load balancing, presenting the Load-aware Task Migration (LATM) algorithm that dynamically redistributes tasks to optimize resource utilization. They work on general edge computing environments with limited mobility considerations while considering resource similarity and migration costs to balance the load across edge nodes. In contrast, our work focuses on vehicular MEC environments, addressing high mobility and latency-sensitive applications while considering vehicle proximity and fairness in MEC host selection, reducing downtime, and ensuring low E2E delay. Ouyang et al. [13] propose a mobility-aware dynamic service placement framework for MEC, which leverages Lyapunov optimization to minimize long-term user-perceived latency while adhering to a predefined migration cost budget, addressing challenges related to resource utilization and service continuity. While their paper focuses on the general mobility-aware service placement in various MEC scenarios, our work provides proximity-based solutions tailored to vehicular mobility. Our MATM algorithm also embeds location awareness and fairness in host selection directly, making it more practical for real-time vehicle applications. Ma et al. [14] focus on container migration in static IoT edge networks, triggered by load thresholds to optimize resource use. In contrast, our approach uses real-time Proximity-Based Migration Notifications and the MATM algorithm to dynamically trigger migration and ensure fair load balancing. De Vita et al. [16] propose a deep reinforcement learning-based approach for optimizing application relocation in MEC-enabled LTE-A networks, leveraging SimuLTE [19] to demonstrate significant latency and resource utilization improvements compared to traditional

TABLE I: Contrastive analysis of different works

Feature	Ours	[9]	[10]	[11]	[12]	[13]	[14]	[15]	[16]	[17]	[18]
Detailed E2E delay analysis	✓	–	–	–	–	–	–	–	–	–	–
Fairness consideration	✓	–	–	–	–	–	–	–	–	–	–
Focus on downtime (interruption time)	✓	✓	✓	✓	–	–	✓	✓	–	✓	–
Integration of migration	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
More than two MEC hosts scenario	✓	–	✓	–	✓	✓	✓	–	✓	✓	–
Use of real-world safety applications	✓	–	–	–	–	–	–	–	–	–	–
Simu5G as simulator	✓	–	–	–	–	–	–	–	–	–	✓

heuristic policies. They focus on reducing delays but do not explicitly optimize downtime during migration. Hathibelagal et al. [15] experimentally compare three migration strategies for MEC-assisted 5G-V2X applications. The three strategies are discussed regarding their viability, the service downtime recorded, and the amount of state preserved after migration, with the user mobility emulated using the ETSI MEC Sandbox [20]. However, their study does not discuss the reasons for migration, focusing solely on the migration strategy instead of the application/task offloading decision-making algorithm. Also, they do not measure the E2E latency. In contrast, our work puts emphasis on all the steps involved, from migration detection to the pre-relocation strategy for seamless application migration, and includes an application offloading decision algorithm. Rezazadeh et al. [17] introduce MiGrror, a synchronized mirroring approach for live migration in MEC environments. Comparing the results obtained by traditional methods, their proposal achieves a reduction in service downtime, migration time, and packet loss. They use MobFogSim [21] and SUMO [22] for performance evaluation. However, their work does not take load balancing or resource usage at the target MEC host into account during the migration decision process. In contrast, our work ensures fairness in MEC host selection; it considers resource availability, reduces E2E delay, and enhances downtime compared to their work. Araujo et al. [18] introduce a mobility-aware orchestration framework supported by the RAVENS architecture for real-time mobility data acquisition, leveraging Simu5G to demonstrate reductions in Round-Trip Time (RTT) and resource consumption during task offloading and service migration in dynamic vehicular MEC scenarios. Their study involves only two MEC hosts, simplifying the target MEC selection process and limiting its evaluation to complex, real-world scenarios. This setup does not account for diverse MEC infrastructures with varying resource loads and network conditions. In contrast, our work targets dynamic and high-mobility environments with multiple MEC hosts, optimizing proximity and resource availability. We evaluate performance across varying vehicle densities and algorithms, integrating them into our migration workflow. Unlike [18], we implement a realistic safety application with high data rates to reflect real autonomous vehicular scenarios. Table I shows a comparative analysis of the works related to our proposed work.

III. System Model

Consider a VEC environment with n vehicles denoted by $V = \{v_1, v_2, \ldots, v_n\}$, and a set of h MEC hosts denoted by $M = \{m_1, m_2, \ldots, m_h\}$. The workflow of our scenario is illustrated in Fig. 2. We implement the application called "Safety app", where the vehicles request the offloading of sensing application from a centralized controller known as the "Orchestrator" [5]. Once the best MEC host is selected, it offloads and processes the vehicle's data and subsequently sends the necessary responses, such as navigation instructions, back to the vehicle. We define $R = \{r_1, r_2, r_3\}$ as the set of resources: CPU, RAM, and Disk, respectively; k is the index of the resource type r_k. Each MEC ($m_j \in M$) is characterized by a maximum CPU, RAM, and Disk capacity, denoted by $C^{max}_{j,k}$. Similarly, Rq_{i,r_k} represents the resource requirements of the resource type r_k for a given vehicle v_i. Table II shows the notations used throughout our paper.

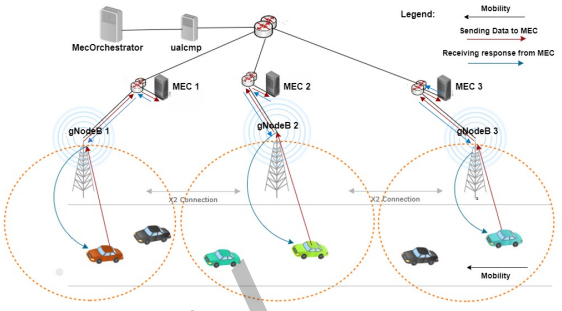


Fig. 2: Architecture of the *"Safety App"* in autonomous driving scenario.

TABLE II: Important notations

Notation	*Definition*
V	Vehicle set, $V = \{v_1, v_2, \ldots, v_n\}, v_i \in V$
M	MEC host set, $M = \{m_1, m_2, \ldots, m_h\}, m_j \in M$
R	set of resources, $R = \{r_1, r_2, r_3\}, r_k \in R$
W	Weights set, $W = \{w_1, w_2, w_3\}, w_k \in W$
t_s	Time Slot $s \in \mathbb{Z}_{\geq 0}$
$x^{t_s}_{ij}$	Binary variable indicating whether v_i is allocated to m_j at t_s
I_i	Required Instruction Per Request (IPR) of v_i
$C^{max}_{j,k}$	Maximum capacity of m_j in terms of resource type r_k
Rq_{i,r_k}	Requirement of v_i in terms of resource r_k
$U^{t_s}_{j,r_k}$	Utilisation rate of resource r_k in m_j at t_s
$Load^{t_s}_j$	Load of MEC host m_j at t_s
F^{t_s}	Jain's Fairness Index in the system at t_s

A. Autonomous Driving Service for Safety control

We implement an autonomous vehicle application, referred to as the *"Safety application"* within a MEC environment, as illustrated in Fig. 2. In our scenario, the MEC host offloads the vehicle's data (*e.g.* sensed data) and processes these data, then sends necessary responses back to the vehicle, such as alerts, navigation instructions, and acknowledgments. This real-time processing is crucial for ensuring autonomous driving systems' safety and smooth operation. We consider a highway environment where autonomous vehicles are served by the MEC hosts who are responsible for assisting them. Network coverage is provided by base stations (i.e., gNodeB) at the roadside. In the case of multiple MECs in the simulation, we assume that each base station owns its MEC system and complies with the European Telecommunications Standards Institute (ETSI) MEC framework [1]. Our safety application

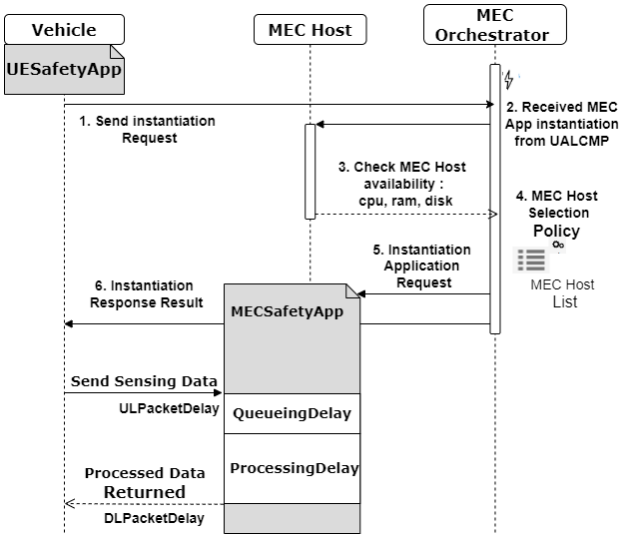


Fig. 3: Data handling sequence between vehicle and MEC Host.

is realized to be run on both the vehicle (UESafetyApp) and MEC (MECSafetyApp) sides. Each User Equipment (UE) application is installed on the vehicle, whereas the MEC application will be on-boarded to all MEC systems. Vehicle applications have CPU, RAM, and Disk requirements that must be allocated to the MEC host. The number of instructions required to execute each vehicle's packet is determined using the Instruction Per Request (IPR) parameter I_i [23], [24] for a vehicle v_i, which follows an exponential distribution with a rate parameter λ. To simulate a real-time use case, we generate packets following the Poisson process where the inter-arrival rate follows an exponential distribution as considered in [25]. The Poisson process accurately represents independent random events over time, a common feature in network traffic. This approach ensures unpredictable packet arrival times based on a statistically sound distribution, providing a realistic view of network load. Also, the payload size of each packet on the uplink direction follows an exponential random distribution [23]. For the payload size on the downlink, we fix the data size to a given value compatible with most remote sensing data in autonomous driving [23]. To ensure a good Quality-of-Service (QoS) characterized by low E2E delay for vehicles with unpredictable movement patterns, each vehicle's application profile must be dynamically migrated across various edge nodes to align with their mobility. We define the total simulation time as T, discretized by a fixed time interval of $\Delta t_s = 11\mu s$, corresponding to the subscription interval used by the location service to track the vehicle's location. Hence for each time slot t_s where $s \in \mathbb{Z}_{\geq 0}$, a binary variable, $x_{ij}^{t_s}$, is used to represent the dynamic placement decision. Specifically, $x_{ij}^{t_s} = 1$ indicates that the application of vehicle $v_i \in V$ is hosted at MEC host $m_j \in M$ during time slot t_s, and $x_{ij}^{t_s} = 0$ otherwise. Importantly, since each vehicle is served by only one MEC host at any given time slot t_s, the following constraints govern the application placement decisions $x_{ij}^{t_s}$:

$$\sum_{j=1}^{h} x_{ij}^{t_s} = 1, \quad \forall i, t_s. \tag{1}$$

$$x_{ij}^{t_s} \in \{0, 1\}, \quad \forall i, j, t_s. \tag{2}$$

B. Evaluation Metrics

1) ***End-to-End Delay****:* We define the E2E delay as the sum of the *"initDelay"* and the Total Response Time (TRT), as shown in Eq.(1). Fig. 3 describes the communication sequence diagram between a vehicle and a MEC host. We define the *"initDelay"* as the sum of 3 sub delays as follows: (i) the time spent to deliver the allocation request from the vehicle to the orchestrator, (ii) the time to allocate the request to the best MEC host according to the task offloading policy, and (iii) the time spent receiving the acknowledgment with the address of the best allocated MEC host to the vehicle. The *"initDelay"* corresponds to the decision-making time, it is represented by the steps one to six in Fig. 3. We define in Eq. (2) the *"Total Response Time"* (TRT) as the time between the transmission of sensed data as a data packet from the vehicle to MEC and the reception of response data at the vehicle after data executed and processed by the MEC.

$$E2EDelay = initDelay + TRT \tag{1}$$

$$\begin{aligned} TRT =& ULPacketDelay + QueueingDelay \\ &+ ProcessingDelay + DLPacketDelay \\ &+ InterruptionTime \end{aligned} \tag{2}$$

The uplink delay denoted by *"ULPacketDelay"* is the time interval between generating a packet at the vehicle and its reception by the MEC Host. The queueing delay, *"QueueingDelay"*, refers to the duration a data packet waits in a queue before being processed by the MEC host, where the queueing system is M/M/1 queue. The time taken by each packet within a MEC application follows an exponential distribution. The processing delay, *"ProcessingDelay"*, refers to the duration the MEC host takes to execute or process a task or data packet after being dequeued. The resources required by each vehicle's request are allocated on the MEC host in fair sharing mode [26], whereby active MEC applications proportionally share all available computing resources based on their requested rates, potentially receiving more capacity than specified when contention is low [26]. The downlink packet delay, *"DLPacketDelay"*, is the duration from when the MEC sends the response to its reception at the vehicle. We add the interruption time to the TRT in case of migration. It is the downtime or the period during which the vehicle is not connected to the MEC application on the initial MEC host, and the connection with the new MEC host has not yet been established. As the migration starts before the handover starts, the orchestrator, as a centralized agent, starts the relocation after being notified by the location service. This means that the migration is doing in a mode "make-before-break". Therefore, the migration time does not affect the TRT.

2) ***Utilization Rate****:* To evaluate the load balance of task offloading strategies, the utilization rate of a specific resource type r_k [12] for m_j is defined by:

$$U_{j,r_k}^{t_s} = \frac{\sum_{i=1}^{n} x_{ij}^{t_s} \times Rq_{i,r_k}}{C_{j,k}^{max}} \tag{3}$$

where j is the index of m_j, i is the index of v_i, r_k is the type of resource with index k and t_s is the time slot.

3) ***Load of MEC host****:* The Load of the MEC host m_j depends on accumulating the resource requirements of vehicle applications executing on it [27], it is defined by:

$$\text{Load}_j^{t_s} = \sum_{k=1}^{3} \omega_k \times U_{j,r_k}^{t_s} \tag{4}$$

where j is the index of m_j, t_s is the time slot, ω_k is the weight of resource type r_k.

*4) **Jain's Fairness Index**:* It is used to measure the fairness of resource allocation among multiple entities [28]:

$$F^{t_s} = \frac{\left(\sum_{j=1}^{h} Load_j^{t_s}\right)^2}{h\sum_{j=1}^{h}(Load_j^{t_s})^2} \tag{5}$$

where j is the index for m_j, t_s is the time slot, h is the number of MEC hosts in the environment. It ranges from 0 to 1, where a value of 1 indicates perfect fairness (equal distribution), and lower values signify increased disparity.

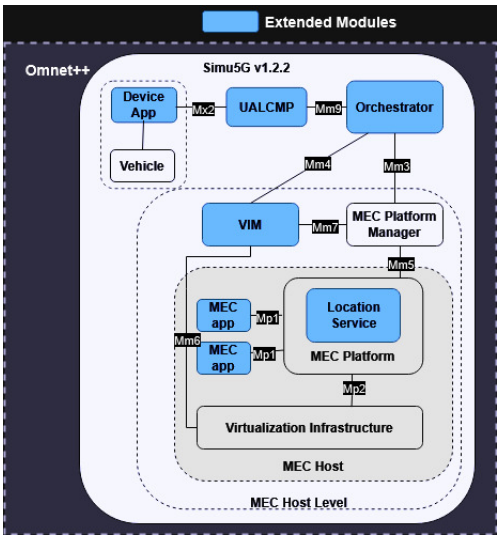


Fig. 4: Simulation Architecture.

IV. The Proposed Algorithm

In this section, we introduce the Mobility-Aware Task Migration (MATM) algorithm, which comprises the Proximity-Based Migration Notification and the Distance-Load Trade-Off (DLTO) algorithm.

A. Proximity-Based Migration Notification (PMN)

Migration in Vehicle Edge Computing (VEC) is essential to ensure seamless service delivery during vehicular mobility. We propose this dynamic approach after detecting the limitations of static benchmark algorithms [29], which increase latency and affect service continuity during mobility. This section describes our proposed migration workflow, emphasizing minimal downtime and effective resource management.

Our proposed migration workflow has been implemented in Simu5G [8], a 5G network simulator based on OMNeT++ [30], as an extension for the Orchestrator mechanism and the MEC Location Service defined in ETSI GS MEC 013 Location API [7]. These extensions enable the orchestrator to manage application migration dynamically and proactively in a pre-relocation mode, leveraging real-time location updates provided by the Location Service at each subscription interval time. The extended modules in the MEC architecture are mentioned in Fig. 4. The migration workflow is described in Fig. 5. The migration workflow involves four key components: the MEC App, MEC Service, Orchestrator, and Vehicle. Each component plays a distinct role in ensuring efficient pre-copy application migration across MEC hosts. The MEC application (app) handles the application logic and communicates with the vehicle. MEC Service tracks vehicle locations and makes migration notifications. The orchestrator coordinates the migration process and selects the target MEC host.

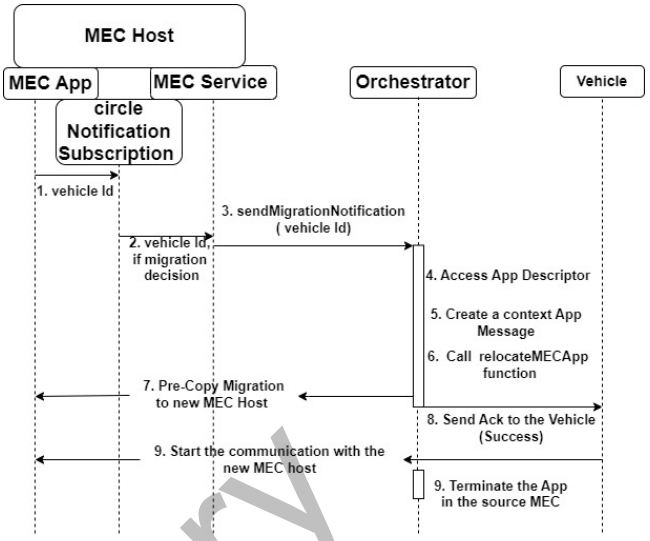


Fig. 5: Application migration procedure in ETSI-MEC in Simu5G.

*1) **Location Service Rules**:* The Location Service is a MEC service implemented in Simu5G, aligned with the ETSI GS MEC 013 Location API specifications [7], [8]. Its primary functions include obtaining the current locations of User Equipment (UE) and Base Stations (BSs) and enabling a circular notification subscription. Our extension enhances the Location Service by making migration notifications subscriptions and notifying the orchestrator to initiate migration. This decision-making process leverages the notification subscription to track vehicle locations at each subscription interval. Based on the location and the vehicle's direction, the subscription informs the Location Service about the corresponding vehicle ID. The Location Service then proactively sends a migration notification to the orchestrator, which employs the Distance-Load Trade-Off (DLTO) migration algorithm to handle relocation. This extension enhances the functionality of the Location Service and the orchestrator by eliminating the need for vehicle migration requests. In addition, throughout the pre-relocation phase, the source MEC host continues to serve the vehicle without any interruptions. Instead, the proactive monitoring mechanism ensures that migration is triggered automatically when a vehicle should be migrated, reducing latency and improving system responsiveness.

*2) **Orchestrator Rules**:* When the orchestrator receives a migration notification from the Location Service, it retrieves the vehicle's application description and related information from an internal map. This approach eliminates the need of additional information from the vehicle, ensuring efficient handling. Next, the orchestrator generates a context application message containing essential details, including the application descriptor ID, the initial request ID, and the device application ID (which corresponds to the app requested by the vehicle during the initial simulation phase). This information is used to carry out the application migration. The orchestrator possesses the historical information about all vehicles like application ID, the MEC hostname, the reference to the MEC host where the MEC application has been deployed, the connection to

VIM which handles the availability of virtualized resources on the MEC host, and vehicle address and port for downstream using UDP sockets, etc. Subsequently, the orchestrator invokes the relocation function, where it accesses the application descriptor and employs the Distance-Load Trade-Off (DLTO) migration algorithm to select a suitable new MEC host. During this process, the orchestrator initiates a pre-copy migration, transferring a copy of the application to the target MEC host while ensuring the vehicle remains connected to the original MEC host. Communication with the original host continues until the vehicle receives an acknowledgment from the orchestrator confirming the migration's success. This acknowledgment includes the address and port details of the target MEC Host, allowing the vehicle to transition seamlessly. This approach minimizes downtime and ensures service continuity. After the vehicle successfully switches to the target MEC host, the orchestrator terminates the application instance on the initial MEC host, completing the migration process.

Algorithm 1: DLTO migration algorithm

Data: $mecList$, $appDesc$, $vehicle$
Result: $bestHost$
$bestHost \leftarrow nullptr$;
$maxScore \leftarrow -1$;
$minDistance \leftarrow \infty$;
for <mec in mecList> **do**
  Check Resources availability(CPU,RAM,$Disk$)
  if *isAllocable* **then**
    Check Service availability
    if *Service available* **then**
      Score = $0.5 \times CPU + 0.3 \times RAM + 0.2 \times Disk$
Sort $mecList$ by $Score$ in descending order
L = length($mecList$)
$bestMecList$ = $mecList[:\frac{L}{2}]$
for <mec in bestmecList> **do**
  Calculate Distance between $vehicle$ and mec
  if $Distance < minDistance$ **then**
    $bestHost = mec$
if *bestHost* = *nullptr* **then**
  Add vehicle to a queue
return $bestHost$

B. Distance-Load Trade-Off (DLTO) algorithm

Our DLTO Algorithm is designed to optimize the selection of MEC hosts. The algorithm considers the computational load of MEC hosts and their physical proximity to the vehicle, ensuring a balance between resource availability and reducing communication latency. The algorithm starts by checking all the MEC hosts to find those that can allocate enough resources for the application. The evaluation reviews CPU, RAM, and disk needs as mentioned in the application descriptor. Then, it checks the availability of the required location service. If the location Service is available, it calculates a composite score for each valid candidate based on resource availability. The composite score will consider CPU, RAM, and disk capacity with weights of 0.5, 0.3, and 0.2, respectively. Based on the score calculated for each MEC host, the MEC host list is sorted by score in descending order. Then, the algorithm filters the MEC hosts to keep only half of the MEC hosts with the highest composite score on the list. The algorithm selects the top candidates as potential winners and then checks their physical proximity by calculating the Euclidean distance between the vehicle's position and the MEC hosts. This step identifies the nearest MEC host to the vehicle and minimizes the communication delay.
The DLTO Algorithm optimizes resource utilization by assigning a score based on the availability of resources to handle the application workload on potential MEC hosts. It prioritizes MEC hosts with greater scores before selecting the one closest to the vehicle. In addition, proximity as a decision criterion minimizes the delay between the vehicle and the MEC host, improving the quality of service for latency-sensitive applications. It enables proactive decision-making, dynamically evaluating available MEC hosts and vehicle locations in real-time, enabling seamless and adaptive migration. Overall, the DLTO migration algorithm combines computational efficiency and spatial optimization, making it well-suited for dynamic VEC environments where mobility and low-latency requirements are critical. It not only ensures reliable application performance but also optimizes MEC infrastructure usage, paving the way for scalable and adaptive edge computing systems. In case the allocation fails or there is no MEC host available, the vehicle request is queued until it is allocated by an available MEC host. For reproducibility, we make the code of our proposed algorithm publicly available in [31].

TABLE III: Simulation Parameters

Parameter Name	*Value*
h	3 MEC hosts
n	[10, 50] vehicles
W	$\{0.5, 0.3, 0.2\}$
Carrier Frequency	2 GHz
Numerology index	2
Resource Blocks	50
Handover Latency	0.005s
Vehicle Speed	uniform(80, 130) km/h
Simulation Time	200 s
Uplink rate (msg/s)	Pois(5), Pois(0.5) [23]
Uplink payload (Bytes)	Exp(40000) [23]
Downlink payload (Bytes)	313 [23]
Downlink Bandwidth (MBit/s)	0.05
Instruction Per Request (IPR)	Exp(500) [23]
Min CPU Required (MIPS)	165130 [23]
Processing Speed (MIPS)	4712460
Road Map Length	4 Km
gNodeBs inter distance	2 Km
Downlink rate (msg/s)	20
Subscription interval	$11\mu s$

V. Simulation and Performance Evaluation

In this section, to validate the effectiveness of our proposed method, we evaluate the performance of the MATM methodology in a simulation environment and compare it with benchmark algorithms implemented in simu5G [8] and analyzed in [29].

A. Simulation environment and Parameters

In this study, we use the Simu5G [8] open-source simulator, which is based on the OMNet++ [30] framework for the packet level simulation and integrates the INET [32] framework for the network protocol stack. The edge environment considered in this experiment comprises h MEC hosts and n vehicles, the number of MEC hosts is set in the simulation to $h = 3$, and the number of vehicles n varies between 10 to 50 vehicles. Each vehicle has a speed selected uniformly between 80 to 130 km

per hour, and the simulation time is set to $200s$. Each MEC host has a CPU, RAM, and Disk capacity. The weights ω_k of the different resource types are set to 0.5, 0.3, and 0.2 for CPU, RAM, and Disk resources, respectively. The vehicles move 4 km along a single-lane highway and reach a density of around 15 vehicles per km. The traffic flow reaches 900 vehicles per hour. Each vehicle runs the *"UESafetyApp"*. Besides, it considers diverse data rate patterns in a realistic environment; we choose two kinds of uplink data rates: about 30% with the high data rate and the remaining vehicles with a small data rate. Every experiment is repeated 30 times with different seeds. Table III summarizes the main simulation parameters.

B. Proposed algorithms

1) ***MATM*** algorithm, as described in section IV by the combination of PMN and DLTO.
2) ***DLTO*** algorithm, is the MATM without migration as described in section IV.
3) ***First-Fit with migration (FFM)*** enables migration by using proximity-based migration notifications (PMN) to detect the need and relocates vehicle apps with the First-Fit algorithm.
4) ***Best-Fit with migration (BFM)*** uses proximity-based migration notifications (PMN) to detect when migration is needed and relocates vehicles using the Best-Fit algorithm.

C. Benchmark Algorithms

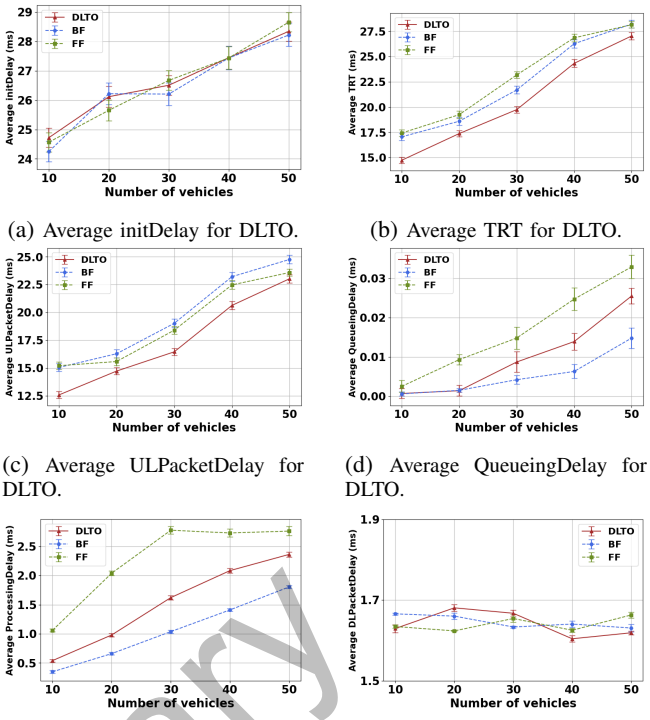
We consider two approaches as benchmarks to evaluate our algorithm. They are implemented in Simu5G and described as follows:

1) ***First-Fit (FF)*** corresponds to "MEC-Service Based" [26] selects the first available MEC host meeting the application's resource and service requirements. This involves choosing the first host with the necessary capacity and services to support the MEC application.
2) ***Best-Fit (BF)*** corresponds to "Available-Resource Based" [26] [33] evaluates all MEC hosts in the list to find the one with the necessary resources and the highest CPU speed.

D. Results and Discussion

To evaluate our proposed approach, we analyze the different components of E2E delay for different numbers of vehicles. The proposed approach refers to the (MATM) algorithm that comprises the Proximity-Based Migration Notification (PMN) and the Distance-Load Trade-Off (DLTO) algorithm. We start with evaluating the DLTO algorithm with the benchmark algorithms mentioned in the previous section.

*1) **DLTO algorithm Assessment**:* In this experiment, illustrated in Fig. 6, we compare the effectiveness of the DLTO algorithm without the PMN assigned for migration. We should remember that the DLTO is an algorithm applied by the orchestrator to select a MEC host for task/application offloading by ensuring Distance-Load trade-off. The *"initDelay"* in Fig. 6a reflects the independence between the algorithm used and the initiation period, attributed to the similarity in the complexity of the algorithms evaluated in this study. As a result, shown Fig. 6b while the DLTO algorithm outperforms the BF and FF algorithms, its advantage is always present even with a higher number of vehicles, but the percentage of delay reduction is reduced with a high number of vehicles from 14% with 10 vehicles to 2% with 50 vehicles. It is worth mentioning the effectiveness of DLTO by decreasing the *"ULPacketDelay"* as shown in Fig. 6c and the ability to balance the load to have a tendency near to the BF algorithm

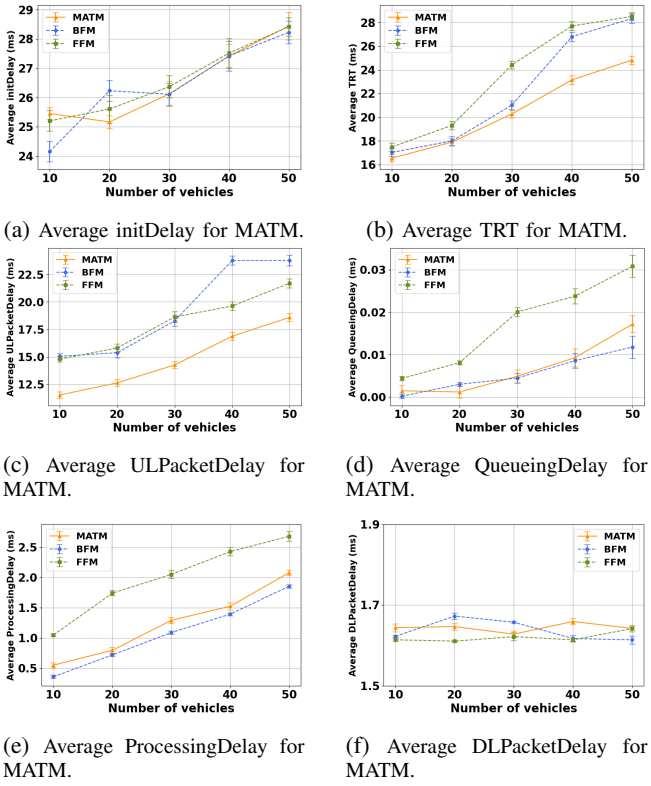


(a) Average initDelay for DLTO.
(b) Average TRT for DLTO.
(c) Average ULPacketDelay for DLTO.
(d) Average QueueingDelay for DLTO.
(e) Average ProcessingDelay for DLTO.
(f) Average DLPacketDelay for DLTO.

Fig. 6: Evaluation of E2E Delay components for DLTO algorithm without Migration.

that offload applications in balancing way, this is present in the Fig. 6d and Fig. 6e refer to *"QueueingDelay"* and *"ProcessingDelay"* respectively. Our algorithm selects the MEC host based on availability, and distance to MEC hosts during mobility, this reflects the tendency to have a Fairness and distance trade-off, which explains the decrease in the *"ULPacketDelay"* and the behavior of *"QueueingDelay"* and *"ProcessingDelay"*. In contrast, Fig. 6f illustrates the relative stability of *"DLPacketDelay"* regardless of the algorithm used and the number of vehicles on the road, attributed to the small size of downlink packets sent from the MEC host to the vehicles. In this stage, with the reduction of the performance for a higher number of vehicles, we resort to extending the DLTO with the migration approach.

*2) **MATM algorithm Assessment**:* In this experiment, we apply the proximity-based migration notification (PMN) combined to the DLTO algorithm to have the MATM algorithm. We compare the performance of the MATM algorithm with both BFM and FFM algorithms. Fig. 7 presents the different E2E delay components for the different algorithms. As shown in Fig. 7b, the TRT increases with a growing number of vehicles, given a fixed number of MEC hosts. This trend is expected as a higher vehicle density demands more MEC resources. Compared to the other algorithms detailed in Section V, the MATM algorithm demonstrates a significant advantage, achieving an average TRT reduction of approximately 15%. This reduction was achieved with an interruption time (downtime) of around 3 ms less than the application downtime in previous works from the literature [15] [17]. This downtime reduction is due to the efficiency of our migration implementation, where the orchestrator proactively selects a new MEC host and allocates the vehicle application to it in pre-relocation



(a) Average initDelay for MATM.

(b) Average TRT for MATM.

(c) Average ULPacketDelay for MATM.

(d) Average QueueingDelay for MATM.

(e) Average ProcessingDelay for MATM.

(f) Average DLPacketDelay for MATM.

Fig. 7: Evaluation of E2E Delay components for MATM.

mode, where the source MEC host continues to serve the vehicle without any interruptions during the migration time. It is worth mentioning that the migration time (i.e., the time required to complete the migration process) is in the range of 10s and 15s, which is compatible with the literature [34]. This migration time is back end-stage during migration, which means the migration occurs while the source MEC host still serves the vehicle. In addition, Fig. 7b highlights the reduction in TRT achieved by the MATM algorithm relative to BFM and FFM. The percentage of reduction grows with the number of vehicles, underscoring the effectiveness of the MATM algorithm in managing higher vehicle flows. This improvement is primarily attributed to the algorithm's ability to optimize application migration, which becomes increasingly critical as vehicle density rises.

As shown in Fig. 7c, the MATM algorithm achieves the lowest delay among all approaches. This superior performance can be attributed to its ability to migrate vehicle applications to MEC hosts near the vehicles during mobility. This proximity significantly improves the *"ULPacketDelay"*, as the algorithm described in Section IV prioritizes selecting MEC hosts with high capabilities and closer proximity to the vehicles. In contrast, the BFM algorithm combines the best-fit strategy with Proximity-Based Migration Notification (PMN) to enhance its performance. However, this approach focuses solely on load balancing across MEC hosts without considering the vehicle's distance or mobility patterns. Fig. 7d and Fig. 7e present the evaluation of *"Processing Delay"* and *"Queueing Delay"* for the MATM algorithm compared with the BFM and FFM under varying vehicle loads. The FFM algorithm shows the highest *"ProcessingDelay"* and *"QueueingDelay"* as the number of vehicles increases, likely due to unbalanced allocation distribution, where the FFM algorithm allocates vehicles in the first MEC host available, which makes the first MEC host available overloaded. In contrast, the BFM algorithm maintains relatively lower *"processingDelay"* and *"QueueingDelay"* due to their well-load allocation strategies, which do not overload a specific MEC host. The MATM algorithm consistently exhibits a *"ProcessingDelay"* and *"QueueingDelay"* comparable to that of the BFM algorithms while outperforming the FFM algorithm. This improvement is due to the DLTO migration algorithm's ability to ensure the trade-off between load distribution and proximity. It aims to achieve effective load balancing, similar to the BFM algorithm while ensuring that application allocation remains close to the vehicle during mobility.

Fig. 7f and Fig. 7a illustrate the evaluation of *"DLPacketDelay"* and *"initDelay"* respectively across BBM and FFM algorithms. The results indicate that both delays remain unaffected by the choice of algorithm. The stability of *"DLPacketDelay"* is primarily due to the small size of the packets transmitted from the MEC to the vehicles. Fig. 7a shows that the *"initDelay"* increases with the number of vehicles due to the congestion in requesting initiation from vehicles simultaneously. In this experiment, we suppose that the vehicles ask for the allocation at the same time at the beginning of the simulations. The two Fig. 6b and Fig. 7b compare the MATM algorithm (with migration) and the DLTO algorithm (without migration) in terms of average TRT across varying vehicle counts. With more vehicles, MATM achieves lower TRT (24.5 ms) compared to DLTO (27 ms), demonstrating the effectiveness of migration. Our MATM algorithm consistently outperforms DLTO, particularly under higher loads, improving system responsiveness in MEC environments. Practically speaking, our proposed solutions are ideal for use as xApps at near-real RAN Intelligent Controllers (i.e. in the timeframe between 10 ms and 1 s) in the Open-RAN architecture for 5G networks and beyond [35].

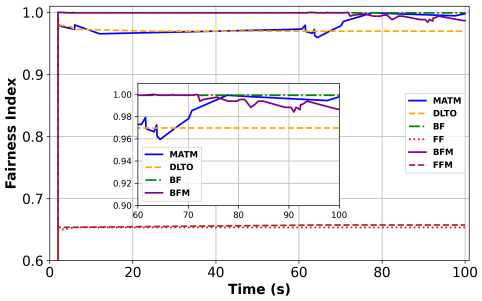


Fig. 8: Fairness Index over the time.

*3) **Fairness Analysis**:* The graph in Fig. 8 illustrates the fairness index over time for different algorithms in a two-lane vehicle mobility scenario. The main plot shows that MATM and BFM consistently achieve a fairness index close to 1, demonstrating their high effectiveness in ensuring a good level of fairness across the MEC hosts. The results of these two algorithms are the closest to the BF algorithm, which has the proportional distribution of vehicles between the MEC hosts, which ensures the optimal fairness index. On the other hand, we can see that DLTO performs moderately, maintaining stable fairness indices between 0.9 and 1. Comparing DLTO to MATM, we notice that MATM increases the fairness by taking into consideration both the load and the distance, where the DLTO maintains the same allocation throughout the simulation time, as MEC applications in MEC hosts do not track vehicle mobility. This causes some MEC hosts to

be overloaded compared to others. Regarding both BFM, we notice that the migration decreases a little bit in the fairness index compared with BF algorithm. In contrast, FF and FFM exhibit the lowest fairness indices, with values stabilizing around 0.65 and 0.67, respectively. Overall, we can emphasize the significant advantage of algorithms like MATM and BFM in ensuring a good level fairness, while the performance of FF and FFM indicates that more optimization is required in order improve their equity in resource allocation. The results also highlight the efficiency and effectiveness of both **MATM** and **DLTO** algorithms, particularly in environments where low delay and fairness are critical.

VI. Conclusion and Future Work

In this paper, we design a novel mobility-aware task migration algorithm to achieve a desirable balance between faster serving for delay-sensitive applications and fairness of resource utilization. This work is motivated by the rapid demand for autonomous vehicles, and it is becoming increasingly challenging for the edge nodes to handle a safety application in complex traffic conditions. To do so, our methodology has two pillars, the proximity-based migration notification (PMN) and the distance-load trade-off (DLTO) algorithm. We implement the PMN in Simu5G as an extension of the location service and the orchestrator rules to enable the migration notification by tracking the mobility information of the vehicles ensuring real-time decision-making. It is a proactive solution that decreases the interruption time by making the orchestrator pre-relocate the vehicles' applications using our DLTO algorithm. Simulation results demonstrate that our proposed methodology effectively reduces E2E delay by up to 15%, compared to other approaches from the state of the art while achieving a good level of fairness. As a future work, we aim to extend the MATM algorithm to consider additional QoS constraints and to further improve load balancing through machine learning techniques. We will evaluate the performance in diverse scenarios with varying traffic densities, heterogeneous edge node capabilities, and different network topologies. In addition, other important metrics for delay-sensitive data processing in 5G-IoT settings, like age-of-information (AoI), will be considered.

Acknowledgment

This work was partially funded by the ANR HEIDIS (https://heidis.roc.cnam.fr/; ANR-21-CE25-0019) project.

References

[1] ETSI Multi-access Edge Computing (MEC). https://www.etsi.org/technologies/multi-access-edge-computing.

[2] A. A. Kherani, G. Shukla, S. Sanadhya, N. Vasudev, M. Ahmed, A. S. Patel, R. Mehrotra, B. Lall, H. Saran, M. Vutukuru, A. Singh, S. Seshasayee, V. R. Viswakumar, and K. Loganathan. Development of mec system for indigenous 5g test-bed. In *2021 International Conference on COMmunication Systems NETworkS (COMSNETS)*, 2021.

[3] L. Hou, M. A. Gregory, and S. Li. A survey of multi-access edge computing and vehicular networking. *IEEE Access*, 10:123436–123451, 2022.

[4] S. Hakak, T.R. Gadekallu, P. K. R. Maddikunta, S. Koppu, M. Parimala, C. d. Alwis, and M. Liyanage. Autonomous vehicles in 5g and beyond: a survey. *Vehicular Communications*, 39:100551, 2023.

[5] ETSI MEC model in Simu5G. https://simu5g.org/users-guide/mec.

[6] X. Dionysis, P. Nikos, M. Lazaros, and V. Christos. Mobility management for femtocells in lte-advanced: Key aspects and survey of handover decision algorithms. *IEEE Communications Surveys Tutorials*, 16(1):64–91, 2014.

[7] ETSI GS MEC 013: Multi-access Edge Computing (MEC); Location API. https://forge.etsi.org/rep/mec/gs013-location-api.

[8] Simu5g. https://github.com/Unipisa/Simu5G, 2024.

[9] L. Gkatzikis and I. Koutsopoulos. Migrate or not? exploiting dynamic task migration in mobile cloud computing systems. *IEEE Wireless Communications*, 20(3):24–32, 2013.

[10] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis. Live service migration in mobile edge clouds. *IEEE Wireless Communications*, 25(1):140–147, 2018.

[11] T. Taleb, A. Ksentini, and P. A Frangoudis. Follow-me cloud: When cloud services follow mobile users. *IEEE Transactions on Cloud Computing*, 7(2):369–382, 2019.

[12] X. Zhu, W. Yao, and W. Wang. Load-aware task migration algorithm toward adaptive load balancing in edge computing. *Future Generation Computer Systems*, 157:303–312, 2024.

[13] T. Ouyang, Z. Zhou, and X. Chen. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. *IEEE Journal on Selected Areas in Communications*, 36(10):2333–2345, 2018.

[14] Z. Ma, S. Shao, S. Guo, Z. Wang, F. Qi, and A. Xiong. Container migration mechanism for load balancing in edge network under power internet of things. *IEEE Access*, 8:118405–118416, 2020.

[15] M. A. Hathibelagal, R. G. Garroppo, and G. Nencioni. Experimental comparison of migration strategies for mec-assisted 5g-v2x applications. *Computer Communications*, 197:1–11, 2023.

[16] F. De Vita, G. Nardini, A. Virdis, D. Bruneo, A. Puliafito, and G. Stea. Using deep reinforcement learning for application relocation in multi-access edge computing. *IEEE Communications Standards Magazine*, 3(3):71–78, 2019.

[17] A. Rezazadeh, D. Abednezhad, and H. Lutfiyya. Migrror: Mitigating downtime in mobile edge computing, an extension to live migration. *Procedia Computer Science*, 203:41–50, 08 2022.

[18] P. J. Araújo, H. F. López, and A. Santos. Efficient mobility management for mec orchestration in vehicular scenarios. *2024 20th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 383–390, 2024.

[19] Simulte. https://omnetpp.org/download-items/SimuLTE.html.

[20] ETSI MEC Sandbox. https://try-mec.etsi.org/.

[21] Mobfogsim. https://github.com/diogomg/MobFogSim.

[22] SUMO Simulator. https://www.eclipse.org/sumo/.

[23] O. Wendlasida, A. Andrea, J. Badii, C.T. Hind, and G. Rémy. Can edge computing fulfill the requirements of automated vehicular services using 5G network ? In *The 2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, Singapore, Singapore, June 2024. IEEE.

[24] N. Auluck, A. Azim, and K. Fizza. Improving the schedulability of real-time tasks using fog computing. *IEEE Transactions on Services Computing*, 15(1):372–385, 2022.

[25] A. Bauer S. Kounev P. Heegaard F. Metzger, T. Hofeld. Modeling of aggregated iot traffic and its application to an iot cloud. *Proceedings of the IEEE*, 107:679–694, 2019.

[26] G. Stea A. Virdis A. Noferi, G. Nardini. Rapid prototyping and performance evaluation of etsi mec-based applications. *Simulation Modelling Practice and Theory*, 123:102700, 2023.

[27] C. Li, Y. Wang, H. Tang, and Y. Luo. Dynamic multi-objective optimized replica placement and migration strategies for saas applications in edge cloud. *Future Generation Computer Systems*, 100:921–937, 2019.

[28] J. Zhou, F. Chen, Q. He, X. Xia, R. Wang, and Y. Xiang. Data caching optimization with fairness in mobile edge computing. *IEEE Transactions on Services Computing*, 16(3):1750–1762, 2023.

[29] R. Sayegh, A. Dandoush, H. Marouane, and S. Hoteit. Preliminary Evaluation and Optimization of Task Offloading and Latency in Vehicular Edge Computing. *IEEE 21st International Conference on Smart Communities: Improving Quality of Life Using AI, Robotics and IoT*, 2024.

[30] Omnetpp. https://github.com/omnetpp/omnetpp, 2024.

[31] Matm. https://github.com/rimsayegh/simu5g-MATM.

[32] inet. https://github.com/inet-framework/inet/releases/tag/v4.5.0, 2024.

[33] Paulo J. Araújo, Helena Fernández López, João Faria, and Alexandre Santos. Towards mobility management in mec simulation. *2023 IEEE 9th International Conference on Network Softwarization (NetSoft)*, pages 207–211, 2023.

[34] G. Nardini, D. Sabella, G. Stea, P. Thakkar, and A. Virdis. Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks. *IEEE Access*, 8:181176–181191, 2020.

[35] O-RAN WhitePaper - Building the Next Generation RAN. https://www.o-ran.org/resources, October 2018.