Optimizing Task Migration Decisions in Vehicular Edge Computing Environments

Ziqi Zhou*, Youming Tao*, Agon Memedi*, Chunghan Lee†, Seyhan Ucar†,
Onur Altintas†, and Falko Dressler*
* School of Electrical Engineering and Computer Science, TU Berlin, Germany
† InfoTech Labs, Toyota Motor North America R&D, CA, U.S.A.
Email: {zhou, tao, memedi, dressler}@ccs-labs.org, {chunghan.lee1, seyhan.ucar, onur.altintas}@toyota.com

***Abstract*—Vehicular networks are undergoing rapid transformations that require efficient and reliable management of computational tasks. Edge computing is supposed to provide these resources with minimal delay. The concept of vehicular micro clouds (VMCs) offers such services through collection among the vehicles, creating a virtual edge server. Within the VMC framework, the efficient migration of computational tasks among vehicles stands as a foundational yet intricate challenge. In this paper, we propose a novel task migration mechanism that leverages both task scheduling and offloading techniques to achieve this objective. Our mechanism mainly consists of two algorithmic modules: (1) task scheduling that prioritizes the tasks according to the earliest deadline first (EDF) principle before offloading decisions and (2) task offloading that assigns each task to the most appropriate vehicle according to the completion time optimization. As a particular feature, we consider the dwell time of cars, i.e., the estimated time vehicles stay within the VMC. We conduct extensive simulations to evaluate the impact of various parameters on task performance, such as data rate, traffic density, computing capacities, and computational demands. Our research advances the state-of-the-art in intelligent transportation system (ITS) by revealing the operational benefits of task migration in the VMC.**

***Index Terms*—virtualized edge computing, task offloading, resource scheduling, task migration, vehicular micro cloud**

I. Introduction

The rapid development of vehicle-to-everything (V2X) communication [1, 2] and continuously increasing computing power of vehicles act as enablers for next generation intelligent transportation system (ITS) solutions. Together with advanced cellular V2X (C-V2X) communication services, 5G also introduced the concept of mobile edge computing (MEC) [3, 4]. Edge computing allows to offload computational tasks to edge servers for fast processing with very low communication latencies [5–8]. Given the limited deployment of MEC and also to enhance the set of features available, virtualized edge computing offers new capabilities, particularly focusing on next generation 6G networks [9]. Such virtalized edge computing is, for example, one of the main building blocks for metaverse applications including virtual reality and cooperative sensing [10–15].

In the vehicular context, the concept of vehicular micro clouds (VMCs) allows the grouping of vehicles into small clusters, which virtually provides edge computing services on the road [16, 17]. Vehicles participating in such clusters share data among each other via direct V2X communication. The next obvious step is to fully use the computational resources of modern self-driving cars for advanced ITS applications and even full-scale smart cities [18–20].

One of the core features of a VMC is a task migration mechanism that enables the dynamic allocation of computational tasks among vehicles and edge servers. If the computing tasks generated by each vehicle can be easily processed locally, both computational resources at the virtual edge server as well as communication resources can be saved. However, vehicles with weak computing power may not be able to finish all these tasks within the deadline, while vehicles with strong computing power may often be idle. Therefore, we consider migrating computing tasks between vehicles to improve the utilization of computing resources and to accelerate task processing to meet relevant deadlines. With the adaptation of network resource allocation and operational efficacy, the requirements of numerous time-sensitive tasks can thus be satisfied.

However, achieving a reasonable task redistribution among vehicles in a VMC via task migration faces many challenges. Firstly, compared with fully local computing, task migration introduces additional time costs caused by communication via wireless channels. How to optimally balance the costs incurred from communication and computation for better task processing efficiency is still not clear. Secondly, due to the mobility of vehicles, the decision-making procedure needs to take into account not only the existing resources in a VMC, but also the availability of these resources in the future.

In this paper, we present an efficient decision algorithm for task migration in a VMC. Our mechanism enhances task migration by alternately invoking two algorithm modules: (1) a task scheduling module and (2) a task offloading module. The task scheduling module sorts all the computing tasks that have not been offloaded according to the earliest deadline first (EDF) principle. That is, the task with the earliest completion deadline will be given the highest priority for offloading. For each task to be offloaded, the task offloading module greedily assigns it to the vehicle that minimizes the expected task completion time. The module considers the so-called dwell time of a car, i.e., the estimated remaining time it will be available in the VMC [21].

In order to assess the performance of our system, we conducted a thorough simulation-based analysis of task migration. We particularly focus on the impact of the traffic density and

the possible data rate of the C-V2X communication channel. Computational capabilities and demands are modeled according to adequate probability distributions. We were mainly interested in performance metrics like the task completion time, the lateness, and the failure rate.

Our main contributions can be summarized as follows:

- We present a novel task allocation algorithm for migration decisions in vehicular edge computing.
- We thoroughly evaluate the algorithm focusing on task completion time, lateness, and failure rate.

II. Problem Formulation

In the following, we introduce the system model as well as the main problem of task migration.

A. Overview of Vehicular Micro Clouds

A vehicular micro cloud represents a number of cars that is in physically close proximity and acts as a virtual edge server to nearby users [16, 17]. We consider a VMC anchored at a fixed geographical region, e.g., any cars in a particular intersection can be part of the VMC. All these existing vehicles have the ability to generate a variable number of heterogeneous computational tasks, each with random data sizes, varying execution complexities, and deadlines. Without loss of generality, we assume that the time is discretized into slots, denoted as $t = 0, 1, \ldots$, where each slot lasts for a unit duration of time. The vehicles in the VMC aim to collaborate with each other, forming a virtual edge server, to complete these computational tasks. The VMC also provides central coordination. In general, the central controller can be any entity with knowledge about computational tasks and vehicle resources in the cloud.

B. Vehicle Model

Due to the mobility of vehicles, the memberships size of the VMC can vary rapidly over time. We use M_t to denote the number of vehicles in each time slot t. Moreover, we associate each vehicle i a pair of time slot indices (t_i^{in}, t_i^{out}) to indicate when it joins or leaves the VMC, respectively. For each vehicle, we assume that the task generation follows an uniform arrival process. That is, at the beginning of each time slot, the number of newly generated tasks at each vehicle follows a uniform distribution over a bounded range of $[\underline{n}, \overline{n}]$, where $\underline{n}, \overline{n} \in \mathbb{N}^+$ are assumed to be public prior knowledge. Therefore, the number of total tasks generated in each time slot t is proportional to the vehicle number M_t. Moreover, the vehicles are equipped with different compute capabilities, determined by their different CPUs. In this paper, we measure the computing capability of the vehicles in million instructions per second (MIPS).

C. Task Model

We consider a heterogeneous task setting, that is, computational tasks have distinct computing complexities and are associated with different data sizes and completion deadlines. We describe task complexity by million instructions (MI) to compete the job. Furthermore, the task data size measures the amount of data needed during the execution of the task. Thus, the larger the data size is, the more communication time will be consumed when we migrate the task from a vehicle to another one.

D. Central Controller Model

In this paper, the central controller is the entity that performs the task scheduling and offloading decisions. Information about all vehicles in the VMC is collected leveraging beacons broadcast periodically from vehicles. The central controller thus has the global knowledge of the vehicles and tasks in the VMC. In particular, when a vehicle joins the micro cloud, it informs the controller about its computational power. The controller also knows if a vehicle is no longer part of the cloud. This can be achieved if the controller loses connection to the vehicle or if the vehicle sends a beacon when it leaves the cloud. The controller, however, does not know the future dynamics of the micro cloud. The information about newly generated tasks, including the task complexity and data size, is also updated to the controller in each time slot. Thus, the controller has knowledge about the current cars in the micro cloud, their processing capabilities, and the requirements of the remaining tasks in the system. It should be noted that the notion of a central controller does not have to be a part of an infrastructure, e.g., edge server. Rather, the controller can be referred to as any entity that has knowledge about the tasks and the current status of the VMC, e.g., a vehicle with such knowledge in the VMC can do the task allocation.

E. Task Migration Setup and Objective

The central controller sequentially assigns each newly generated task to the vehicle that will execute the task. After the assignment is made, the task will be migrated from the source vehicle to the target vehicle immediately unless the controller decides to have the task executed locally. Once the task arrives at the assigned vehicle, it enters the waiting queue for execution. Therefore, the runtime status of each task comprises: {generation, communication, waiting, processing}. Since each task will be assigned by the controller in the immediate next slot of the one when it is generated, we disregard the time from generation to migration, and mainly consider the time costs T_k of each task k as the sum of the time duration spent for communication T_k^{c}, waiting T_k^{w}, and processing T_k^{p}, i.e., $T_k \triangleq T_k^{\text{w}} + T_k^{\text{c}} + T_k^{\text{p}}$.

Ideally, one would like to minimize the total time spent for completing each task, i.e., T_k. However, from a system perspective, this is impractical, because there is resource competition between tasks. Therefore, in this paper, we aim to minimize the average task completion time in VMC, i.e., $\frac{1}{N_t}\sum_{k=1}^{N_t} T_k$, where N_t is the number of all tasks up to time slot t.

F. Formal Definitions of Main Concepts

Let $\mathsf{VMC} = \{\mathsf{Veh}, \mathsf{Tsk}\}$ be a vehicular micro cloud with Veh being the set of existing vehicles and Tsk being the set

of unfinished tasks. Note that both Veh and Tsk change over time slots. Without loss of generality, we use $\mathsf{Veh}(i)$ to denote the i-th vehicle in Veh and use $\mathsf{Tsk}(k)$ to denote the k-th tasks in Tsk.

For each $\mathsf{Veh}(i)$, we use a 3-tuple to describe it, which includes its joining time $t_{\text{join}(i)}$, leaving time $t_{\text{leave}}(i)$ and its computing capability $\mathsf{MIPS}(i)$. That is,

$$\mathsf{Veh}(i) = (t_{\text{join}(i)}, t_{\text{leave}}(i), \mathsf{MIPS}(i)). \tag{1}$$

For each $\mathsf{Tsk}(k)$, we use a 6-tuple to describe it, which includes its generating time $t_{gen}(k)$, completion deadline $t_{ddl}(k)$, complexity $\mathsf{MI}(k)$, data size $S(k)$, its source vehicle (generator) index $g(k)$ and its processor index $p(k)$. That is

$$\mathsf{Tsk}(k) = (t_{\text{gen}}(k), t_{\text{ddl}}(k), \mathsf{MI}(k), S(k), g(k), p(k)). \tag{2}$$

We now refine some notations we used in the last section. Let $T^{\text{c}}(k,j)$ be the communication time of $\mathsf{Tsk}(k)$ used for migrating from $\mathsf{Veh}(g(k))$ to j. Let the transmission rate be R, which is supposed to be the same in the VMC. Then for any $\mathsf{Tsk}(k)$, we have

$$T^{\text{c}}(k,j) = \begin{cases} 0, & \text{if } g(k) = j \\ \frac{S(k)}{R}, & \text{otherwise.} \end{cases} \tag{3}$$

Let $T^{\text{p}}(k,j)$ be the time interval of the execution of $\mathsf{Tsk}(k)$ if it is processed by $\mathsf{Veh}(j)$, which can be described as follows,

$$T^{\text{p}}(k,j) = \frac{\mathsf{MI}(k)}{\mathsf{MIPS}(j))} \tag{4}$$

Let $T^{\text{w}}(k,t,j)$ be the delay of $\mathsf{Tsk}(k)$ waiting for execution if processed by $\mathsf{Veh}(j)$ start from time slot t. Suppose $\mathsf{Veh}(j)$ has the task queue of $\mathsf{Tsk}(k_1), \cdots, \mathsf{Tsk}(k_q)$, where $\mathsf{Tsk}(k_1)$ is under processing and the other tasks are waiting for execution, then

$$T^{\text{w}}(k,t,j) = \frac{(1-\alpha_{j,t})\mathsf{MI}(k_1)}{\mathsf{MIPS}(j)} + \sum_{\kappa=2}^{q} T^{\text{p}}(k_\kappa), \tag{5}$$

where $\alpha_{i,t}$ implies the proportion of instructions that has been finished for processing $\mathsf{Tsk}(k_1)$ till time slot t.

At last, we provide a concrete example based on data analysis task to further illustrate our models and settings, and to conclude this section.

Example 1. ***Task Completion Time Calculation***
For instance, a typical analysis task with a 10 MB data size and 2000 MI task complexity is generated by a car, and the current wireless channel is supported by sub-6 GHz 5G technology with a 100 Mbps Data Rate[22]. The Oracle offloads the task to an idle car that has 4000 MIPS computing capability. The predictive completion time of this task is the following.

- *Scenario: Analyzing sensor data for anomalies.*
- *Task Data Size: 10 MB* = 8×10^7 *bits*
- *Result Size: 10 KB* = 8×10^4 *bits*
- *Data Rate: 100 Mbps* = 10^8 *bit/s*
- *Task Complexity: 2000 MI*
- *Computing Capability: 4000 MIPS*
- *Processing Time: For* 2×10^9 *instructions with 4000 MIPS computation ability, processing time* = $\frac{2\times10^9}{4\times10^9} = 0.5$ *seconds.*
- *Up-Migration Time: For 10 MB data over a 100 Mbps link, migration time* = $\frac{8\times10^7 \text{ bit}}{10^8 \text{bit/s}} = 0.8$ *seconds.*
- *Down-Migration Time: For 1 MB data over a 100 Mbps link, migration time* = $\frac{8\times10^4 \text{ bit}}{10^8 \text{bit/s}} = 0.0008$ *seconds.*
- *Completion Time: Without considering the potential waiting time, the sum of the other components is* $0.5 + 0.8 + 0.0008 \approx 1.3s$

The down-migration time is usually relatively negligible (less than 10^{-3} *of the up-migration delay) and thus is omitted in the calculation.*

Algorithm 1 Task Pre-scheduling Algorithm

Input: Unassigned task list L
Output: A re-ordered list L'
1: Sort tasks in L in increasing order based on deadlines
2: **Return** the sorted list

III. Our Task Migration Decision Method

In the following, we describe and discuss our new task migration decision technique. It consists of two parts: task pre-scheduling and task migration.

A. Task Pre-Scheduling

A crucial balance between fairness and efficiency must be acknowledged in task migration research. However, these two factors are not necessarily mutually exclusive.

Since our algorithm operates in a sequential rather than a parallel processing system, all decisions are made one at a time. Therefore, it's essential to maintain a logical order in the inputs, ensuring that our calculations are both efficient and equitable.

We decided to pre-schedule tasks in a logical order before they are offloaded through a second algorithm. This approach contrasts with first-come first-served (FCFS), where tasks are processed in the order they arrive, as it moves tasks with earlier deadlines to the front of the queue before the task offloading process begins. Compared to EDF, our pre-scheduling strategy does not disrupt the sequence of tasks already offloaded, avoiding recalculations that could decrease efficiency and render previous decisions obsolete.

Our pre-scheduling strategy (cf. Algorithm 1) is particularly effective in dynamic environments and improves overall efficiency and task success rates[1] by prioritizing urgent tasks for resource allocation over less urgent ones, provided the tasks have not yet been offloaded. The earlier a task is assigned resources, the more likely it is to be completed on time, reducing the risk of missing deadlines and thereby improving overall system performance.

[1]Task success rate is defined as the proportion of tasks completed before their deadlines.

Algorithm 2 Task Migration Algorithm

Input: initial car set Veh and task set Tsk.

for time slots $t \leftarrow 1, 2, \cdots$ **do**
  for each $\mathsf{Veh}(i)$ in the latest Veh **do** (in parallel)
    Report the newly generated and historical task information
  end for
  Controller forms the list of unassigned tasks L
  Controller obtains sorted list L' via Algorithm 1
  for each $\mathsf{Tsk}(k)$ in L' **do**
    for all $\mathsf{Veh}(j)$ in Veh **do**
      $T(k,j) \leftarrow T^{\mathrm{c}}(k,j) + T^{\mathrm{p}}(k,j) + T^{\mathrm{w}}(k,t,j)$
    end for
    Find vehicles able to complete $\mathsf{Tsk}(k)$ fastest:

$$\mathsf{OPT}(k) \leftarrow \operatorname*{argmin}_j T(k,j)$$

    if $g(k) \in \mathsf{OPT}(k)$ **then**
      $p(k) \leftarrow g(k)$
    else
      Find $j^* \in \mathsf{OPT}(k)$ s.t. $\mathsf{Veh}(j^*)$ has the least workload
      $p(k) \leftarrow j^*$
    end if
  end for
end for

B. Task Offloading Decision

Our algorithm to make offloading decisions is depicted in Algorithm 2. First, it initializes the migration system, sets the current time to zero, and starts the task generation of each car. In each time step, all the newly generated tasks are added to a pre-scheduled list and sorted based on deadlines for further processing. All the tasks are processed based on their completion time according to the pre-scheduled list order. The completion time includes the local one and the migration ones. The former is the sum of the processing time (T^{p}) and the waiting time (T^{w}), while the latter also adds the additional communication overhead (T^{c}) with the iteration on other current existing cars. In addition, we check the estimated dwell time of both the car generating the task as well as of the candidate for migration, i.e., both need to be part of the same VMC at task generation time as well as at the anticipated task completion time. The shortest migration completion times and corresponding cars will be collected and the car that has the least workload will be marked as the best migration destination.

IV. Simulation and Analysis

In order to assess the performance of the presented task scheduling and migration technique, we performed a thorough simulation-based analysis.

A. Scenario

Without loss of generality, we use a basic intersection mobility scenario to study our algorithms. This scenario contains a single traffic light, four entry points, and 16 possible routes, including straight paths, right turns, left turns, and U-turns. The map covers an area of approximately $400 \times 400\,\mathrm{m}^2$, featuring two 400-meter perpendicular lanes, as depicted in Figure 1. Trips were generated using the SUMO random trip generator, which allows for flexible control over car generation periods, thereby affecting car density.

After generating car patterns with three different traffic density levels (cf. Figure 2), the simulation is run to record mobility traces. These traces contain important characteristics such as the total number of cars, average speed, average duration of car presence on the map, simulation duration, car generation period, and tracking time step. Further details on the simulation setup are provided in Table I.

B. Task Migration Implementation

Building upon our migration approach and two baseline methods—no migration and migration without considering dwell time within the VMC scope, we implement the system in form of a Python program. Table II provides an overview of the key parameters and settings employed in our experimental setup. Several key settings determine how tasks are processed and migrated among vehicles. The computational power of each vehicle is assumed to be uniformly distributed in the range 1–5 MIPS. Tasks have deadlines in the range 2–3 s, and their data sizes vary in the range 1–10 MB. The computational complexity of tasks spans from 1,000–5,000 MI. Each vehicle generates uniformly distributed tasks between 0–3 tasks per

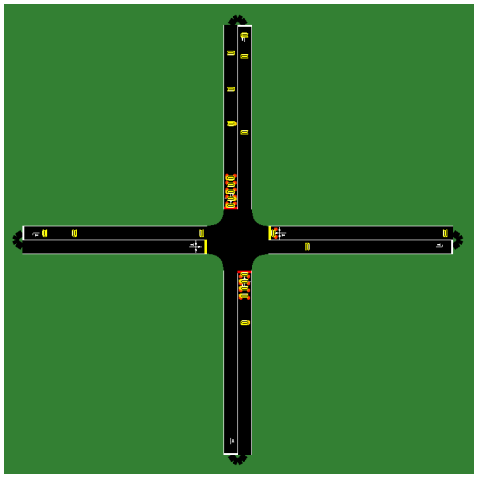


Figure 1. Intersection Scenario

Table I
Road traffic simulation parameters for SUMO

Parameter	Low	Med	High
Total number of cars	480	900	1200
Average speed [km/h]	27.86	25.74	21.20
Average dwell time [s]	865.16	1002.77	887.46
Simulation duration [s]	3600	3600	3600
Car generation period [s]	7.5	4	3
Tracking Time Step [s]	1	1	1

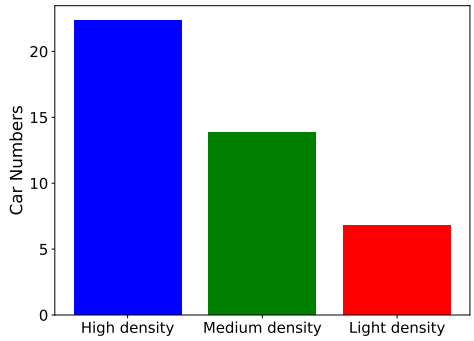


Figure 2. Average number of cars per time step

Table II
TASK MIGRATION SETTINGS

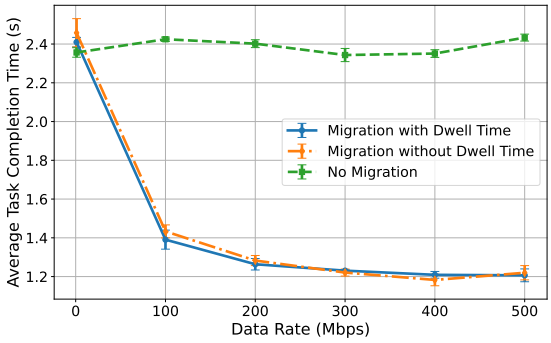
Description	Value
Computer CPU [MIPS]	$\mathcal{U}(1,5)$
Task Deadline [s]	$\mathcal{U}(2,3)$
Task Data size [MB]	$\mathcal{U}(1,10)$
Task execution complexity [MI]	$\mathcal{U}(1000,5000)$
Number of tasks generated per time step per car	$\mathcal{U}(0,3)$
Data Rate [Mbps]	1 - 500

time-step. The data rate for wireless communication and task migration is varying between 1–500 Mbit/s.

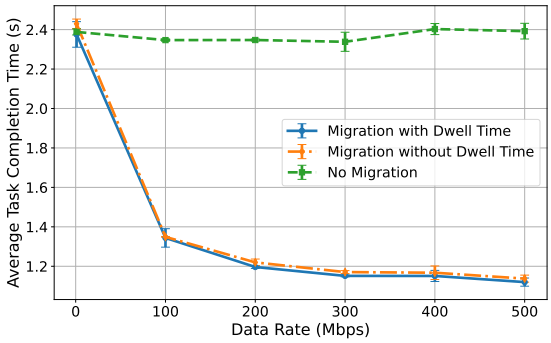
C. Experimental Results and Discussion

Figures 3 to 5 show the obtained results from our simulations for average task completion time, average task lateness rate, and average task failure rate, respectively. All figures plot three strategies. We use "no migration" as a baseline, which we compare our solution with. We also study the impact of the knowledge of the dwell time in all experiments. These results are labeled "migration without dwell time" and "migration with dwell time", respectively. For all three measures, we varied both the communication data rate as well as the traffic density. In the following, we discuss the impact of both measures separately.

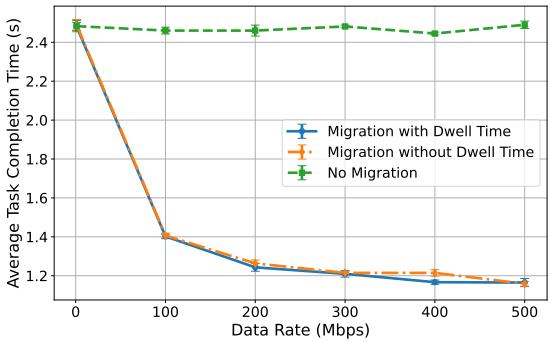
1) Impact of Data Rate: The average task completion delay is the most straightforward metric illustrated in the plots. No migration approach remains consistently high and relatively stable as the data rate changes, which aligns with expectations, i.e., local execution should not be influenced by wireless channel conditions. Both migration approaches, with and without dwell time, show a sharp decrease in completion delay as the data rate increases from 1–100 Mbit/s and then a more gradual decline from 100–500 Mbit/s. The initial significant decrease in completion delay for the migration strategies is attributed to improved transmission conditions, enabling remote processing to save considerable time. In the second regime, where data rates exceed 100 Mbit/s, the gap between the migration approaches and no migration remains larger, though additional gains in performance are limited. This is because the transmission cost becomes negligible once the data rate



(a) Light traffic density



(b) Medium traffic density

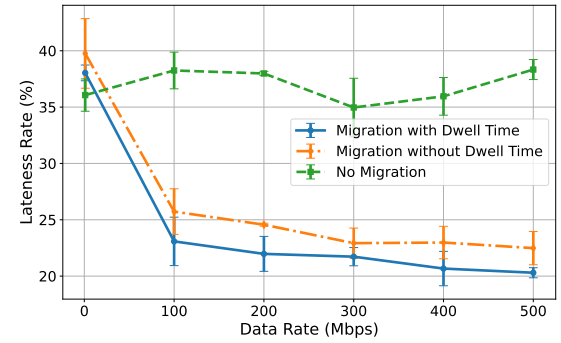


(c) High traffic density

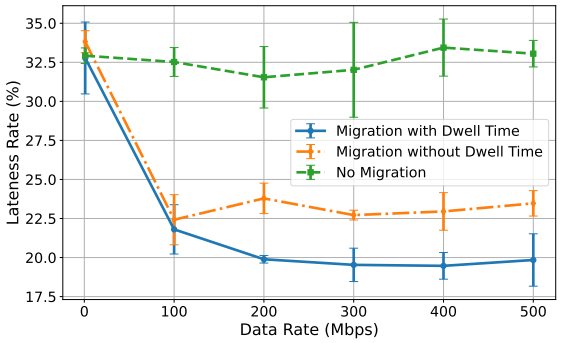
Figure 3. Average task completion time

surpasses a certain threshold, likely around 100 Mbit/s, leading to minimal room for further optimization through reduced transmission times.

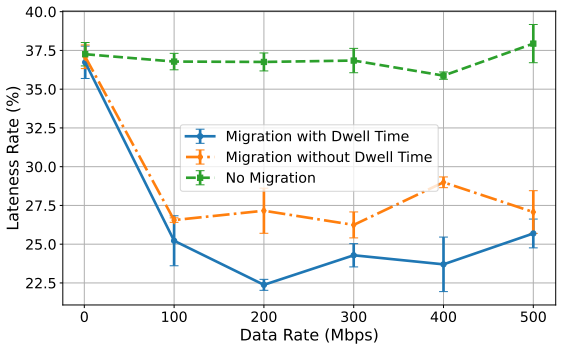
The lateness rate is an indicator of the proportion of tasks completed after their deadlines. No migration approach consistently underperforms, maintaining a high lateness rate of around more than one-third, regardless of wireless communication conditions. Migration without dwell time shows improvement as the data rate increases, indicating that better communication can positively impact task completion times if there exist task migration options. However, our migration strategy with dwell time consideration outperforms both baselines in most scenarios, achieving a lateness rate of approximately 20% to 25%. This



(a) Light traffic density

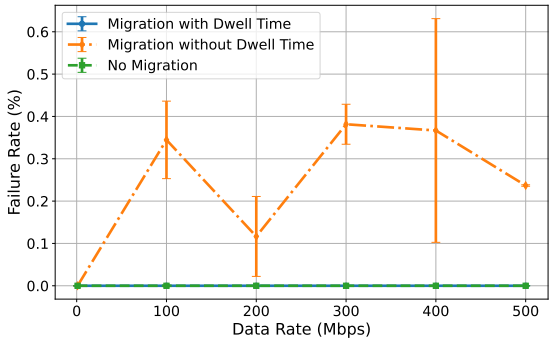


(b) Medium traffic density

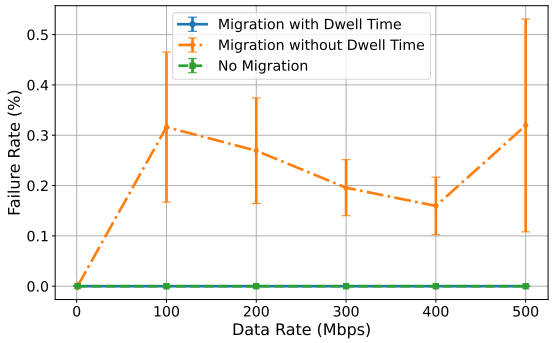


(c) High traffic density

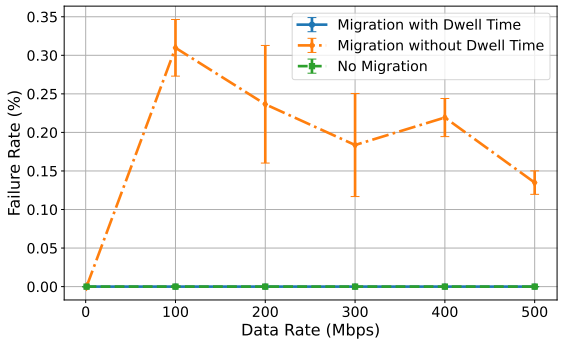
Figure 4. Average task lateness rate



(a) Light traffic density



(b) Medium traffic density



(c) High traffic density

Figure 5. Average task failure rate

lower rate suggests that our approach's additional consideration of dwell time in the VMC scope contributes to more efficient task offloading and reduced delays in environments with varying car density and wireless communication conditions. The only exception occurs in the light-density scenario, where there are fewer cars available to act as servers, leading to some fluctuation in performance. Despite this variability, our approach rarely performs worse than either of the baseline strategies, indicating that even with a smaller pool of available resources, our method maintains competitive efficiency and effectiveness. This resilience underscores the adaptability of our approach in varying traffic density conditions while continuing to outperform baseline strategies in terms of task lateness.

The failure rate is a key metric in system performance analysis. While lateness indicates that tasks are delayed but still completed, failure rate implies a more severe problem where task results are entirely lost, because the processor car has moved out of the VMC with the generator at the moment of task completion. In no migration method, failure is theoretically impossible because the processor and generator are always the same car. Our system maintains a zero failure rate across all data rates because one of the preconditions for migration is that both vehicles must remain in the VMC scope. Migration without dwell introduces a non-zero failure rate, albeit at a low level of less than 0.4%. While this might seem negligible, it poses a potential risk due to the possibility of processors

moving out of range during task execution. Additionally, if the number of tasks is large, even a small failure rate could lead to a significant number of tasks failing, which is undesirable. Given a strict adherence to a zero-tolerance policy for failure, our approach is the best in reliability and success rate.

2) Impacts of Traffic Density: Regardless of the car density, our algorithm consistently outperforms the two baseline approaches, except in one case: the migration approach without dwell time also shows competitive performance in terms of average task completion time.

Considering the outcomes observed across three density modes, it is only the lateness metric in the high-density scenario that exhibits a noticeable yet slight variation, which could be attributed to randomness or fluctuation, especially when the car number is limited. The simulation results suggest that car density does not significantly affect the average task completion time, task lateness rate, or task failure rate. The likely explanation for this observation is that the cars function both as users and service providers. As the number of tasks is approximately proportional to the number of cars, the average task rate per car appears to have a greater impact on these metrics than the density of cars alone.

V. Conclusion

Following the general concepts of next generation virtualized edge computing, in this paper, we proposed a novel task migration mechanism that leverages both task scheduling and offloading techniques to achieve this objective. Our algorithm is designed to operate in the context of vehicular micro clouds, i.e., a specific vehicular communication scenario, in which vehicles share both computational and communication resources to achieve both individual as well as shared goals such as cooperative processing of sensor data or the training of machine learning models. All of which are fundamental building blocks for next generation metaverse applications. Our task migration strategy employs a greedy algorithm that is based on two components: first, a EDF-inspired pre-scheduling of new tasks and, second, a migration decision based on the task completion time as well as the anticipated dwell time of a vehicle in the VMC. First simulation results focusing on the impact of traffic density and communication data rate demonstrate the feasibility of our solution. In future work, we aim to incorporate more realistic elements such as realistic traffic models into our simulations and further refine our model. It is also interesting to investigate the possibility of securing our mechanism with blockchain-based protocols, such as [23].

References

[1] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, 2014.

[2] Z. MacHardy, A. Khan, K. Obana, and S. Iwashina, "V2X Access Technologies: Regulation, Research, and Remaining Challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1858–1877, 2018.

[3] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, Oct. 2017.

[5] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A Survey of Computation Offloading for Mobile Systems," *ACM/Springer Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[6] D. Xu, Y. Li, X. Chen, J. Li, P. Hui, S. Chen, and J. Crowcroft, "A survey of opportunistic offloading," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2198–2236, 2018.

[7] Q.-H. Nguyen and F. Dressler, "A Smartphone Perspective on Computation Offloading – A Survey," *Elsevier Computer Communications*, vol. 159, pp. 133–154, Jun. 2020.

[8] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward Computation Offloading in Edge Computing: A Survey," *IEEE Access*, vol. 7, pp. 131 543–131 558, Jan. 2019.

[9] F. Dressler, C. F. Chiasserini, F. H. P. Fitzek, H. Karl, R. Lo Cigno, A. Capone, C. E. Casetti, F. Malandrino, V. Mancuso, F. Klingler, and G. A. Rizzo, "V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing," *IEEE Network*, vol. 36, no. 3, pp. 24–31, May 2022.

[10] K. Li, Y. Cui, W. Li, T. Lv, X. Yuan, S. Li, W. Ni, M. Simsek, and F. Dressler, "When Internet of Things meets Metaverse: Convergence of Physical and Cyber Worlds," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4148–4173, Mar. 2023.

[11] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A Survey on Metaverse: Fundamentals, Security, and Privacy," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 319–352, 2023.

[12] D. C. Selvaraj, F. Dressler, and C. F. Chiasserini, "Human-Centered Traffic Management Supporting Smart Cities and the Metaverse," in *IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom 2023)*, Kyoto, Japan: IEEE, Jun. 2023, pp. 163–168.

[13] H. Wang, H. Ning, Y. Lin, W. Wang, S. Dhelim, F. Farha, J. Ding, and M. Daneshmand, "A Survey on the Metaverse: The State-of-the-Art, Technologies, Applications, and Challenges," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14 671–14 688, Aug. 2023.

[14] E. Krijestorac, A. Memedi, T. Higuchi, S. Ucar, O. Altintas, and D. Čabrić, "Hybrid Vehicular and Cloud Distributed Computing: A Case for Cooperative Perception," in *IEEE Global Communications Conference (GLOBECOM 2020)*, Taipei, Taiwan: IEEE, Dec. 2020.

[15] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative Task Offloading in Vehicular Edge Multi-Access Networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 48 –54, Aug. 2018.

[16] T. Higuchi, J. Joy, F. Dressler, M. Gerla, and O. Altintas, "On the Feasibility of Vehicular Micro Clouds," in *9th IEEE Vehicular Networking Conference (VNC 2017)*, Turin, Italy: IEEE, Nov. 2017, pp. 179–182.

[17] F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, T. Higuchi, and O. Altintas, "Virtual Edge Computing Using Vehicular Micro Clouds," in *IEEE International Conference on Computing, Networking and Communications (ICNC 2019)*, Honolulu, HI: IEEE, Feb. 2019.

[18] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-Driven Intelligent Transportation Systems: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[19] M. Veres and M. Moussa, "Deep Learning for Intelligent Transportation Systems: A Survey of Emerging Trends," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3152–3168, Aug. 2020.

[20] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-Computing-Enabled Smart Cities: A Comprehensive Survey," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 200–10 232, Oct. 2020.

[21] G. S. Pannu, S. Ucar, T. Higuchi, O. Altintas, and F. Dressler, "Dwell Time Estimation at Intersections for Improved Vehicular Micro Cloud Operations," *Elsevier Ad Hoc Networks*, vol. 122, p. 102 606, Nov. 2021.

[22] M. Cantero, S. Inca, A. Ramos, M. Fuentes, D. Martín-Sacristán, and J. F. Monserrat, "System-Level Performance Evaluation of 5G Use Cases for Industrial Scenarios," *IEEE Access*, vol. 11, pp. 37 778–37 789, 2023.

[23] M. Xu, F. Zhao, Y. Zou, C. Liu, X. Cheng, and F. Dressler, "BLOWN: A Blockchain Protocol for Single-Hop Wireless Networks under Adversarial SINR," *IEEE Transactions on Mobile Computing*, vol. 22, no. 8, pp. 4530–4547, Aug. 2023.