ApMove: A Service Migration Technique for Connected and Autonomous Vehicles

Muhammad Zakarya, *Senior Member, IEEE*, Lee Gillam, Ayaz Ali Khan, Omer Rana, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—Multiaccess edge computing systems (MECs) bring the capabilities of cloud computing closer to the radio access network (RAN), in the context of 4G and 5G telecommunication systems, and converge with existing radio access technologies like satellite or WiFi. An MEC is a cloud server that runs at the mobile network's edge and is installed and executed using virtual machines (VMs), containers, and/or functions. A cloudlet is similar to an MEC that consists of many servers which provide real-time, low-latency, computing services to connected users in close proximity. In connected vehicles, services may be provisioned from the cloud or edge that will be running users' applications. As a result, when users travel across many MECs, it will be necessary to transfer their applications in a transparent manner so that performance and connectivity are not negatively affected. In this article, we propose an effective strategy for migrating connected users' services from one edge to another or, more likely, to a remote cloud in an MEC. A mathematical model is presented to estimate the expected times to allocate and migrate services. Our evaluations, based on real workload traces and mobility patterns, suggest that the proposed strategy "ApMove" migrates connected services while ensuring their performance (∼0.004%–2.99% loss), reduced runtimes, therefore, users' costs (∼4.3%–11.63%), and minimizing the response time (∼7.45%–9.04%). Furthermore, approximately 17.39% migrations are avoided. We also study the impacts of variations in the car's speed and network transfer rates on service migration durations, latencies, and service execution times.**

***Index Terms*—Connected vehicles, edge cloud, Internet of Things, service migration.**

I. Introduction

Connected, or more formally, autonomous cars [1] are considered as mitigators of issues, such as traffic congestion, road safety, inefficient fuel consumption, and pollutant emissions, that current road transportation system suffers from [2]. These cars are usually connected to the remote cloud where their data is stored, processed, and used for various objectives. However, there are various challenges, e.g., the risk of data becoming unusable as it travels to the cloud due to longer distances and delays. To cope with the challenges associated with cloud computing and networks, service providers are now encouraging to work in the direction of massively distributed small-sized datacenter infrastructures (known as cloudlets) that are installed at the edge of the network in close proximity to users. The fog/edge concept generates a lot of buzz since it enhances service agility and the performance of real-time services in terms of response time.

Multiaccess edge computing systems (MECs) offer the capabilities of cloud computing closer to the radio access network (RAN), in the context of 4G and 5G telecommunication systems, and converge with other radio access technologies, such as satellite or WiFi. The MEC could be more effective than the cloud for traffic flow in terms of avoiding congestion, crowds, routing decisions, and management [3]. An MEC is a cloud server running at the edge of a mobile network that is deployed and executed over virtual machines (VMs), containers, and functions. A cloudlet is like an MEC which implies several servers, providing compute services to connected users (cars) in their close proximities [4]. The services of each car are assumed to run in VMs in a cloudlet (MEC) covering a specific geographic area. Therefore, when users (cars) move across several MECs, it would be necessary to move and migrate their services seamlessly and transparently to the new MEC. Since migrations could degrade the performance of these real-time car services; therefore, intelligent decisions should be very important. Most importantly, whether to migrate the services or not; if the car is already at its destination. In this research, our focus would be to propose an effective strategy for migrating connected cars' services from one edge node to another or, more likely, to a remote cloud in an MEC system. The experimentation should be completed using real data sets of service migration, in a simulated fog computing environment [5].

In this article, we propose an effective strategy for migrating connected users' services from one edge node to another or, more likely, to a remote cloud in the MEC systems. We investigate when the migration will be more effective in terms of energy savings, performance, and cost. Such decisions are taken based on the vehicle's speed, distance traveled,

Manuscript received 14 April 2024; revised 2 May 2024; accepted 15 May 2024. Date of publication 20 May 2024; date of current version 23 August 2024. This work was supported in part by the Abdul Wali Khan University, Pakistan, and in part by the Sohar University, Oman. *(Corresponding author: Muhammad Zakarya.)*

Muhammad Zakarya is with the Faculty of CIT, Sohar University, Sohar 311, Oman, and also with the Department of Computer Science, Abdul Wali Khan University, Mardan 23200, Pakistan (e-mail: mohd.zakarya@awkum.edu.pk).

Lee Gillam is with the Department of Computer Science, University of Surrey, GU2 7XH Guildford, U.K. (e-mail: l.gillam@surrey.ac.uk).

Ayaz Ali Khan is with the Department of Computer Science, University of Lakki Marwat, Lakki Marwat, Khyber Pakhtunkhwa 28420, Pakistan (e-mail: ayazkhan@ulm.edu.pk).

Omer Rana is with the School of Computer Science and Informatics, University of Cardiff, CF10 3AT Cardiff, U.K. (e-mail: ranaof@cardiff.ac.uk).

Rajkumar Buyya is with the CLOUDS Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3052, Australia (e-mail: rbuyya@unimelb.edu.au).

Digital Object Identifier 10.1109/JIOT.2024.3403415

2327-4662 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

and more likely its destination that could be predicted using machine learning models. Recent state-of-the-art researches mainly focus on the migration between the edges. However, in this work, we also consider the migration to a remote cloud, in particular, when it is beneficial and performance efficient. For example, if a vehicle is fast enough that result in quick service migrations across edge nodes, then, probably it would be better to migrate its services to a remote cloud that covers a large geographic area [6], [7]. This will avoid frequent migrations. However, it will essentially affect the service latency and response time which have negative impacts over road traffic and crowd management. Similarly, if services are being migrated among edge nodes, then, each migration will have a negative impact on the response time (migration downtime). Therefore, it would be essential to make appropriate migration decisions at the right time without affecting the performance of running applications [8]. The main part in reaching these decisions is the deep neural network that will take traffic data in terms of vehicle speed, distance traveled, distance remaining, destination, and road congestion details that are stored in a decentralized storage area network (SAN). Furthermore, routing decisions could be taken on the edge while traffic management could be based on the huge amount of data stored on a centralized cloud. The research will investigate various methodologies, such as 1) training the model on the edge and then predict; 2) training the model on the cloud and then predict; and 3) training the model on the cloud and then predict on the edge. The major contributions of our research are as follows: 1) we investigate whether a user is, highly, likely to keep moving quickly between edges or not; 2) we investigate when it will be effective to start migrating the running service or application to the target edge/cloud during mobility; 3) an effective migration strategy is suggested to migrate connected users' services from one edge to another edge or, more likely, to a remote cloud in an MEC platform; 4) the proposed algorithm uses a probabilistic approach to estimate the service migration time to decide whether the migration is effective or not; and 5) Google's and mobility data sets are used to validate the model through several plausible assumptions and simulations [9], [10].

The remainder of this article is organized as follows. In Section II, we formulate the service migration problem in the context of connected cars. In Section III, we propose ApMove, an approach to migrate live services across various edges and/or remote clouds for connected vehicles. In Section IV, we validate and evaluate the performance of the suggested technique using real workload traces from the Google clusters and cars mobility data set. In Section V, we provide an outline of the relevant work. Section VI provides a summary of this work. Finally, Section VII discusses several future directions for further research.

II. Background and Problem Description

Fig. 1 shows the system model for the service migration technique that comprises a number of edge clouds ($EC = ec_1, ec_2, \dots, ec_n$) and moving vehicles. Furthermore, a special type of device is integrated into each connected or moving vehicle ($V = v_1, v_2, \dots, v_r$). Moreover, each EC consists of several numbers of servers ($ES = es_1, es_2, \dots, es_n$). We assume each edge server and connected device consist of a sufficient number of VM or container instances to execute the services [5]. Usually, Functions as a Service (FaaSs) are preferred over VMs and containers due to the fact that FaaS functions are lightweight. The resources of each VM or container are denoted by R where ($R = r_1, r_2, \dots, r_u$). All the ECs are connected to a remote virtuality cloud. In this section, we briefly discuss the law of motion and the service migration approach in connected vehicles. A list of all mathematical notations and their description is shown in Table I.

The displacement is the change in position x of a moving object and can be computed through $\Delta x = x_f - x_i$, where Δx is the displacement, x_f is the final position, and x_i is the initial position. Subsequently, we can compute the total displacement as the sum of all displacements between two points. To calculate the other physical quantities of a moving object, we must introduce the time variable t. The elapsed time, given by $\Delta t = t_f - t_i$, is the amount of time it takes to travel between two points—where t_f is the time noted at position x_f, and t_i is the time when the object is at position x_i. The time variable allows us to specify not only where the object is but also for how long it has been there (its position) during its travel, but also how fast (speed) the object is traveling. How fast and quickly an object is traveling can be illustrated by the rate at which the position changes with time, known as velocity v which is the ratio between Δx and Δt. However, a car cannot travel at a constant speed, or velocity, and the displacements may vary across the entire route. Therefore, we assume its instantaneous velocity v' while assuming $t_i = t$ and $t_f = t + \Delta t$ such that limit $\Delta t \to 0$

$$v'(t) = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) \ - x(t)}{\Delta t} = \frac{dx(t)}{dt}. \quad (1)$$

Through dividing the total distance traveled by the elapsed time, we can compute the average speed s. However, the instantaneous speed s' is the absolute value of the instantaneous velocity $|v'|$. The acceleration is the rate at which velocity changes, i.e., the ratio between the velocity Δv and time Δt. Actually, the acceleration denotes the rate of change in speed of the moving object. The instantaneous acceleration is given by

$$a'(t) = \lim_{\Delta t \to 0} \frac{v(t+\Delta t) \ - v(t)}{\Delta t} = \frac{dv(t)}{dt}. \quad (2)$$

Therefore, instantaneous acceleration is the derivative of the velocity function, analogous to how velocity is the derivative of the position function. Speed is measured as the distance moved over time and is given by

$$\text{speed } (s) \ = \frac{\text{Distance}}{\text{time}} = \frac{x_2 - x_1}{t_2 - t_1} = \frac{\Delta x}{\Delta t}. \quad (3)$$

Speed is commonly measured in meters per second (m/s) or kilometers per hour (km/h). Note that identifying the expected maximum speed or velocity of a user might dictate the duration for migration completion, as illustrated in Section III-B. Using the above equation, if we know the speed and the distance

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

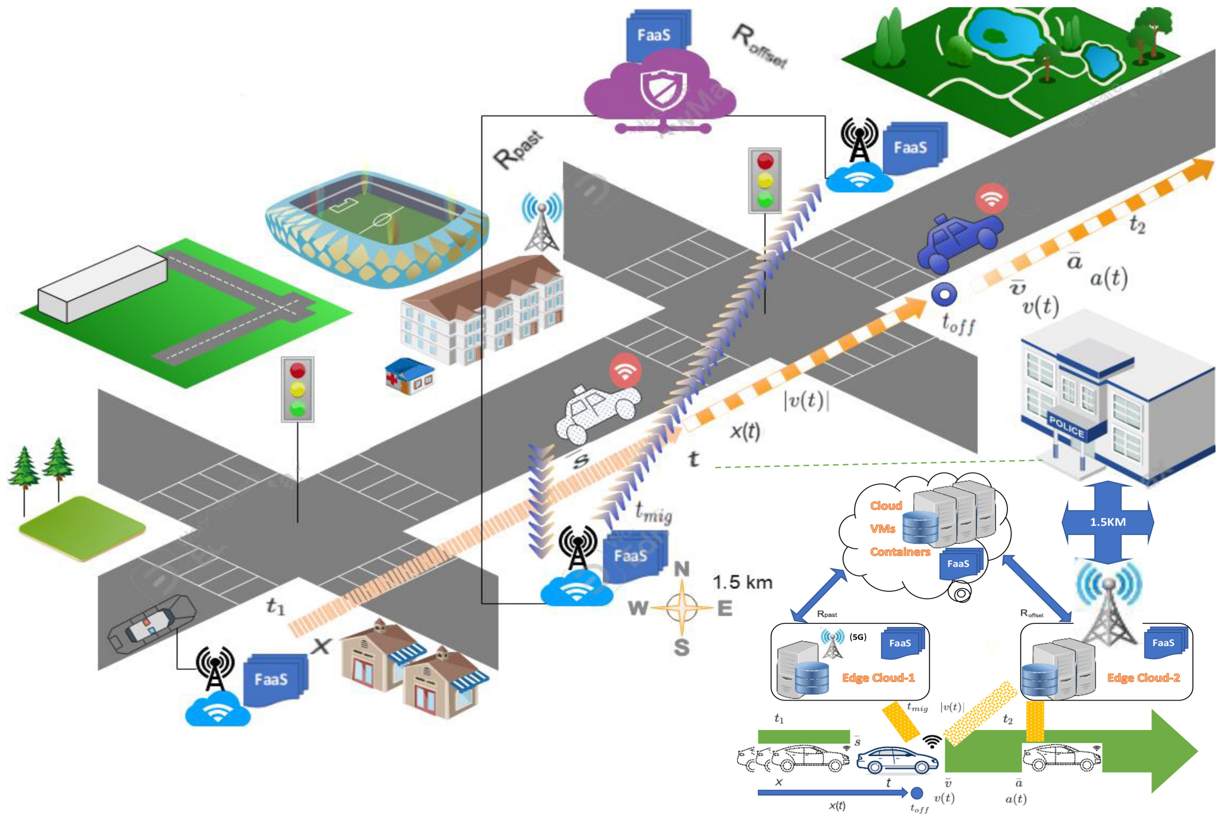

Fig. 1. Proposed service migration strategy for autonomous cars [edge clouds are assumed to use a cellular (5G) technology and a long range WiFi with a communication range of 1.5 km—FaaS means Functions as a Service]—illustration of experimental setup and ApMove methodology using a straight line road.

traveled by a moving object, we can find its time at a particular position

$$\text{time} = \frac{\text{Distance}}{\text{speed}}. \tag{4}$$

This is also possible to compute the time factor from the velocity equation since velocity is the ratio of displacement and time

$$\text{time} = \frac{\text{Displacement}}{\text{velocity}}. \tag{5}$$

To keep it simple, we assume that the distance between two base stations or edge clouds and the speed of the car are known in advance. We discovered the velocity function by taking the derivative of the position function, and similarly, we discovered the acceleration function by taking the derivative of the velocity function. We can calculate the velocity function from the acceleration function and the position function from the velocity function using integral calculus.[1] We can take the indefinite integral of both sides for (1) and (2) to find the velocity and position, respectively

$$v(t) = \int a(t)dt + C_1 \tag{6}$$

$$x(t) = \int v(t)dt + C_2 \tag{7}$$

[1]https://courses.lumenlearning.com/suny-osuniversityphysics/chapter/3-6-finding-velocity-and-displacement-from-acceleration/

where C_1 and C_2 are constants of the integration. Now, if we describe the problem of the connected car service migration, i.e., when the services should be migrated or at which time a vehicle is expected to enter a new edge cloud? The estimated entrance time can help in improving the agility of the services while the running service is already migrated to the destination. We know that acceleration and speed are fundamental physics concepts, but they can be integrated into connected cars for modeling automobile movement within an edge network. This involves dividing the road network into cells using cellular automata models. We believe adapting these models to the specific network of edges, considering factors like geographical restrictions and traffic flow dynamics, and using large data sources like GPS traces can lead to more innovative solutions.

III. Proposed Solution

As shown in Fig. 1, prior to when a particular car enters a specific coverage area of the edge cloud, its running services should be migrated from the source edge to the target edge or more likely to a remote cloud in the case when the car is moving fast and experiences frequent migrations between edges. There are two situations: 1) when the coverage areas overlap and 2) when there is no overlap. In respect of overlapping regions, certain threshold values should be defined to judge the strength of the edge signals and implicitly trigger service migrations. In respect of nonoverlapping coverage

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

TABLE I
LIST OF MATHEMATICAL NOTATIONS

Notation	Description
$\mathbf{H}$	List of servers so that $h \in H$
VM	List of VMs so that $vm \in VM$
$\mathbf{M}$	List of service migration requests
$\mathbf{C}$	List of edge clouds
x	Position
t	Time
s	Speed
s'	Instantaneous speed
$v(t)$	Velocity
$v'(t)$	Instantaneous velocity
$X(t)$	Displacement
a	Acceleration
a'	Instantaneous acceleration
$a(t)$	Acceleration at time t
t_{mig}	Migration time
t_{off}	The point at which migration is proffered
t_1	Previous time (distance)
R_{past}	Previous service execution time
R_{rel}	Relative value for VM runtime
t_2	Remaining time (distance)
R_{offset}	The time at which the migration cost is recouped
C	Constant
V_i	Amount of VM data to be migrated in each round
T_i	The time of each round in the migration
T_{down}	VM down time during the migration process
V_{mem}	Size of the VM (to be migrated)

areas, the time and distance factors will help to explicitly trigger migration decisions. Note that t_1 and t_2 refer to t_i and t_f, respectively. Furthermore, t_{mig} is the duration needed for the completion of the migratable service, and t_{off} is the point at which the migratable service is ready for running on the destination edge cloud [11], [12]. Thus, our method and intention is to ensure that the service is being migrated before t_{off}—otherwise, the performance of the service is not guaranteed to meet service level agreements (SLAs). The main objective of the service migration problem is to compute the t_{off} value (expected) in such a way that

$$t_{\text{mig}} + t_{\text{off}} \leq T_{\text{mig}} \tag{8}$$

where T_{mig} is the expected time of car entrance to the destination edge cloud. In other words, the vehicle should be given resource before T_{mig} on the destination cloud. Too early t_{off} values may waste resources since there will be two services running for the duration of migration t_{mig}. Similarly, too later t_{off} values may degrade the performance of the service since service migration is not completed yet. In the former case, it is possible to run the service either on a remote cloud that covers a large geographical area or push it in a particular queue (Q) for periodical placement. We assume that the scheduler periodically looks for the Q and if there are certain unallocated services; then, the scheduler tries to place them on appropriate resources. The t_{off} can be estimated using

$$t_{\text{off}} = \frac{\Delta x}{s} \times t_{\text{mig}} \tag{9}$$

where Δx denotes the difference between the energy consumption and/or performance of source and target servers [12]. As shown in Fig. 2, t_{off} is a time when the VM will recover its cost of migration Cost_{mig}. Furthermore, $R_{\text{offset}} = t_{\text{mig}} + t_{\text{off}}$ denotes the minimum value that is sufficient to offset Cost_{mig}

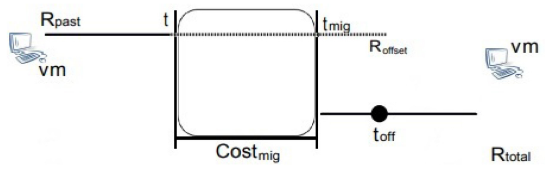

Fig. 2. Process of computing t_{off} [12].

at time t for a VM. This means that t_{off} is related to the efficiencies of source and target servers, i.e., edge clouds.

The migration time t_{mig} can be computed using (10). However, we will use a more robust approach - as described in the subsequent section

$$t_{\text{mig}} = \frac{\text{FaaS}_{\text{size}}}{B} \tag{10}$$

where B is the bandwidth of the network that can be used to migrate the function. In order to minimize t_{mig}, we assume that each FaaS has a redundant, periodically synchronized, copy on the remote cloud. Once a migration decision is triggered, instead of directly transferring the source FaaS to the destination, the FaaS service is copied from the remote cloud. The steps are shown in Algorithm 1. The migration duration and downtime (performance loss) are computed through the mathematical model as demonstrated in Section III-B. The algorithm [line 3 to 5] checks whether the migration will be to another edge or a remote cloud through: 1) observing the vehicle's speed and/or 2) checking if there are very frequent migrations in the list. In line 6, the algorithm computes the expected time when the service migration occurs. Next, we compute the runtimes of all VMs, compute the relative runtimes, and sort them accordingly [lines 7–9]. In case the migration is feasible [line 10], then complete the migration [lines 11–14] otherwise choose another host at the edge/cloud [lines 17–19]. If there are no available resources to accommodate the service, then put the migration request in a queue for allocation at a later time [lines 20–23]. The process continues until all the services are allocated to the appropriate hosts.

A. Relative Runtime

In fact, the past runtimes of VMs, denoted by R_{past}, on a particular host are known to us, therefore, we can statistically derive relative runtimes, denoted by R_{rel} that can guide us to a most suitable host for the placement. For example, suppose we have n VMs running on a certain host (H), and we designate their individual runtimes as $H = \{R_{\text{past}}^{VM_1}, R_{\text{past}}^{VM_2}, \ldots, R_{\text{past}}^{VM_n}\}$. We determine the relative execution timings for each VM in order to compare the runtimes of various VMs running on different hosts. To do this, one method is to divide the execution time (R_{past}) of each VM by the minimum runtime of the VM among all VMs, denoted by set (H). There may be alternative statistical ways to calculate the relative runtime duration, however, we think this method is the easiest. The following (11) is used to determine the relative runtime duration for

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Service Migration Technique

```
Input: List of edge servers/MECs ($\mathbf{C}$), List of service
       migration requests ($\mathbf{M}$), Car's velocity $v'(t)$,
       Migrations history ($\mathbf{H}$), and Threshold ($S_t$)
Output: Service placement
1  $S_t \leftarrow$ Threshold value for migration $v'(t)$;
2  for each $m \in \boldsymbol{M}$ do
3      if $v'(t) \geq S_t$ or $\boldsymbol{H}$ frequently updated then
4          migrate to cloud;
5      end if
6      compute $t_{\text{off}}$ for each $m \in \mathbf{M}$ using Eq. (9) ;
7      $R^m_{\text{past}} \leftarrow$ actual runtimes, regression, XGBoost ;
8      Compute relative runtimes for $c \in C$ with Eq. (11);
9      Sort $\mathbf{C}$ in descending order;
10     if $t_{mig} + t_{\text{off}} \leq T_{mig}$ then
11         for each $c \in \boldsymbol{C}$ do
12             if $c$ has enough resources for $m$ then
13                 allocate $m$ to $c$;
14                 break ;
15             end if
16         end for
17         if $m$ did not fit in any available $c$ then
18             look for another $c$ and allocate $m$;
19         else
20             $m$ cannot be allocated;
21             push the request $m$ into $Q$ (queue);
22         end if
23     end if
24 end for
25 return output
```

each VM:

$$R^{VM_i}_{\text{rel}} = \frac{R^{VM_i}_{\text{past}}}{\min(H)} \tag{11}$$

where $\min(H)$ denotes the VM that has the minimum R_{past}. We use two different approaches to compute $R^{VM_i}_{\text{past}}$, i.e., actual past runtimes and prediction through linear regression and XGBoost approaches. We do the above computation for all available hosts and then select the one that has a VM with the shortest duration (for short-running workloads) or a VM with the longest duration (for long-running workloads). This process ensures that the resources of an appropriate host are provisioned to the VM or application.

B. Migration Time (Performance)

Modeling the performance of migration includes numerous factors, including the VM memory size, network transmission rate, the migration algorithm, and the workload features, i.e., memory dirtying rate. The key parameters are VM size (V_{mem}), network traffic (V_{mig}), total migration time (T_{mig}), down time (T_{down}), memory transmission rate (R), memory dirty rate (D), threshold for last round (V_{th}) and writable working set (W) to transfer hot pages. To minimize T_{down}, live migration copies the dirty pages at the previous round of transmission iteratively. Consider that there are n rounds, which completes the precopy algorithm then the volume of data at round i is V_i and the elapsed time is T_i for $0 \leq i \leq n$. The data transmitted and time during each round are given by

$$V_i = \begin{cases} V_{\text{mem}}, & \text{if } i = 0 \\ D.T_{i-1}, & \text{if } i > 0 \end{cases} \tag{12}$$

$$T_i = \frac{D.T_{i-1}}{R} = \frac{V_{\text{men}}.D^i}{R^{i+1}}. \tag{13}$$

Consider that $D < R$ on average, and ω denotes the ratio of D to R then

$$\omega = \frac{D}{R}. \tag{14}$$

Combining (12)–(14), we get

$$V_i = V_{\text{mem}}.\omega^i. \tag{15}$$

The total network traffic is given by

$$V_{\text{mig}} = \sum_{i=0}^{n} V_i = V_{\text{mem}}.\frac{1-\omega^{n+1}}{1-\omega}. \tag{16}$$

The total migration time is given by

$$T_{\text{mig}} = \sum_{i=0}^{n} T_i = \frac{V_{\text{mem}}}{R}.\frac{1-\omega^{n+1}}{1-\omega}. \tag{17}$$

The migration downtime contains two different parts: 1) the time to transfer lasting dirty pages in the stop-and-copy period, i.e., T_n and 2) the time to resume the VM at the destination host, i.e., T_{resume} which has slight variation and can be characterized as a constant value of 20 ms. The migration downtime is given by

$$T_{\text{down}} = T_n + T_{\text{resume}}. \tag{18}$$

The inequality $V_n \leq V_{th}$ can be written as $V_{\text{mem}}.\omega^n \leq V_{th}$ to calculate the total number of rounds for algorithm convergence, which is given by

$$n = log_{\omega}.\frac{V_{th}}{V_{\text{mem}}}. \tag{19}$$

From the above studies, we determine that a VM having a small memory image and trivial ω would cause a smaller amount of network traffic leading to shorter T_{mig}, therefore is a better nominee for migration. Note that if ω is smaller, then the precopy technique will converge faster. If the D is even larger than the R then the amount of data transmitted in each round i will beat the VM size, which will increase the total migration time even in the worst case the migration will not be accomplished. We do not consider such a situation in our modeling, but Xen has solved this issue using the writable working set technique. The pages that are rottenly dirtied, i.e., hot pages are ignored to transfer till the last round of migration. More details on such type of study can be found in [13]. Earlier studies have shown that migration durations have an important impact over the network traffic, Quality of Service (QoS), and service performance [14]. In this work, we assume stateful migration which means that the entire VM is being migrated. However, stateless migrations, i.e., migrating just the relevant data, as needed for a given service, would require short durations. In this context, those services that require large amounts of history will have high-performance impacts on the infrastructure [12].

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

C. Complexity Analysis

The computational complexity of the ApMove method is strongly dependent on the number of services, servers, migrations, prediction approach, and MECs. We use previous runtimes to estimate the relative value instead of a learning approach. The worst-case complexity is $O(mn)$ where m denotes services and n denotes servers. However, if a learning method is used to estimate the relative value, then complexity will increase depending on the size of the data set, features, and computational resources. Since we use previous runtimes, thus the computational and space complexities of the learning part are constant, i.e., $O(1)$. Furthermore, our method minimizes the number of migrations leading to a logarithmic time complexity, making it more efficient than complex and dynamic programming-based methods. As discussed in [12], the time taken to check if migration is feasible is nonsignificant and can be ignored. When a service is migrated, we increment its value to count its migrations. Therefore, the time needed to check whether the service is being migrated frequently is trivial and can be ignored.

IV. Performance Evaluation

In order to evaluate the feasibility and performance of the proposed model, we use different approaches to service placement and migration [15]. Furthermore, we assume the relocation problem as a bin-packing issue and prefer to solve it with heuristics rather than optimality. We assume that the services are running in VMs that we resemble to functions (FaaS). In case, a service is being migrated several times within a predefined threshold time, which means that the vehicle is traveling fast enough, we relocate that service to the remote cloud. Defining such a threshold value is challenging and has a significant impact on migration statistics and outcomes. Furthermore, static threshold values are not considered good in cloud platforms due to the nature of dynamic workloads and services [16]. In this work, we use static threshold values for migrating services to a remote cloud if cars are moving at higher than 110KM/h or the service has been migrated two times in the last hour. We use the Google data set for service execution times and migration statistics [9]; and the car's mobility data from [10]. The mobility data set and details can be accessed from.[2] The Google's data set, which is an extended version of a previous 2011 data set, includes details about job scheduling from 8 heterogeneous Google clusters for May 2019. The data set contains millions of job submissions that report on various characteristics, such as user, priority, CPU and memory demand, resource usage, submission time, finish time, and many more [17]. We use actual job durations from the data set in our model for migration decisions. Further details about the data set can be found at.[3] Fig. 1 illustrates the experimental design of the ApMove approach where all edge clouds are assumed running on a straight road and have access to the remote cloud.

TABLE II
Servers Characteristics for Simulated MEC Setup
[ECU = CPU Speed (GHz) × Number of Cores]

CPU model	Speed (MHz)	No of Cores	No of ECUs	Memory (GB)	Storage (TB)
E5430	2,830	8	22.4	16	4
E5507	2,533	8	20	8	8
E5645	2,400	12	28.8	16	4
E5-2650	2,000	16	32	24	8
E5-2651	1,800	12	21.6	32	12

TABLE III
Various Types of VM Instances and Their Characteristics—Mem Means Memory and vCPU Denotes a Hyperthreaded Core

Instance type	No of vCPUs	No of ECUs	Speed (GHz) MIPS	Mem (GB)	Storage (GB)
t1.micro	1	1	1.0	0.613	1
t2.nano	1	1	1.0	0.5	1
m1.medium	1	2	2.0	3.75	410
m3.medium	1	3	3.0	3.75	4

A. Experimental Set-Up

Our simulated MEC environment consists of a main datacenter and 30 edge servers which are related to 30 separate edge locations. These edge servers are arranged in a straight line on a single road. We assume that each edge cloud uses cellular 5G technology and equipped with a long range WiFi[4] that has a signal range of up to 1.5km [18]. These edge servers are connected to the main cloud through a 1GB/s network cable. We are aware that roads complexity, number of vehicles, road intersections, and other factors will have an impact over the obtained outcomes [7]. However, we do not take these factors into account to keep it simple for experimental simplification. There are 100 servers in the datacenter that belong to five different CPU architectures, i.e., 20 servers of each type, as shown in Table II. Moreover, each edge server also belongs to these five server types. The linked connected vehicles' services were supposed to be run by virtual computers of four different sizes and speeds. All services are assumed to utilize their provisioned resources as normally distributed. Table III shows the frequency of VMs in vCPUs (cores), which were translated to EC2 compute units (ECUs) and mapped to MIPS ratings. The ECU is characterized as having the "equivalent CPU capability of a 1.0–1.2-GHz 2007 Opteron or 2007 Xeon Processor" and is rated per vCPU/core; hence, the VM total rating is the multiple of cores (number) and ECU rating. The rating is then converted to MIPS for compatibility with iFogSim, which does not accept ECU. The considerable disparity in VM storage capacity assures heterogeneity, however, we are aware that this will have a significant influence on migration costs [19]. The *Scikit-Learn*[5] library is used to implement various machine learning models, such as regression and XGBoost.

In order to be compatible with the iFogSim [10], [19], [20], the speeds of different hosts and Containers, VMs were

[2]https://github.com/diogomg/MobFogSim
[3]https://github.com/google/cluster-data/
[4]https://en.wikipedia.org/wiki/Long-range_Wi-Fi
[5]https://scikit-learn.org/

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
EXPERIMENTAL RESULTS (FOR NUMBER OF MIGRATIONS, THE "+" DENOTES NUMBER OF SERVICES BEING MIGRATED FROM FOG TO CLOUD AND "−" REPRESENTS CLOUD TO FOG MIGRATIONS)—NOMIGRATE MEANS SERVICE ALWAYS RUNNING IN THE CLOUD WHILE ALWAYSMIGRATE MEANS SERVICES ARE MIGRATED BETWEEN EDGES ALONG WITH THE VEHICLES; SIMPLEMIGRATE MEANS THAT SERVICES ARE ONLY MIGRATED BETWEEN CLOUD AND EDGES. FURTHERMORE, EXECUTION TIME IS THE SUM OF SERVICE RUNTIME AND LATENCY MEASURED IN SECONDS

Service placement	Migration approach	Runtime approach	Migratable services %	Accuracy %	Latency (seconds) Fog	Latency (seconds) Cloud	Total migrations	Service execution time	TPR
FillUp	NoMigrate	Past	0	100	1211.67	922.2	0	20821	1
	AlwaysMigrate		7.9	87.6	611.67	2156.54	1203(+0,-0)	28995	0.876
	SimpleMigrate		3	67.6	198.45	1258.89	123(+12,-5)	21690	0.676
	ApMove		1.90	98.1	183.67	1145.1	119(+22,-3)	20849	0.981
	NoMigrate	SR	0	100	1091.13	823.55	0	22799	1
	AlwaysMigrate		8.67	81.65	557.23	2095.43	1283(+0,-0)	29164	0.816
	SimpleMigrate		3.79	89.92	199.81	1301.13	117(+21,0)	24891	0.899
	ApMove		1.53	71.11	190.99	1211.98	112(+10,-6)	22794	0.7
	NoMigrate	BT	0	100	1001.3	1121.3	0	21988	1
	AlwaysMigrate		10.78	79.79	587.51	2100.65	1178(+0,-0)	28902	0.8
	SimpleMigrate		4.53	83.1	199.89	1287.3	129(+17,-1)	21967	0.831
	ApMove		1.53	93.98	188.88	1212.7	121(+13,-4)	21915	0.94

matched to millions of instructions per second (MIPS). We assume a reliable connection among cloud and edges, i.e., zero packet loss. This should be noted that latency is computed using $0.02k^2$ (seconds), where k is the distance of hop from the edge where services are running [21]. In our previous work [15], we have described performance details for numerous workloads or applications that are operating on these hosts. The proposed method is compared with three approaches, i.e., NoMigrate, AlwaysMigrate, and SimpleMigrate. NoMigrate means that services are always running in the cloud and no migrations are happening. Moreover, AlwaysMigrate means that services are migrated between various edges along with the vehicles while SimpleMigrate means that services are only migrated between the cloud and edges. Furthermore, we have compared ApMove with other state-of-the-art methods.

B. Evaluation Metrics

We consider total number of migrations (within the edges and edge-cloud), performance degradation (migration duration), and response time (in minutes) as the performance metrics. Moreover, service execution time, i.e., runtimes of all VMs (seconds), and the accuracy of the prediction approach are also measured. To check the model performance in predicting accurate migration times the true positive rate (TPR) is calculated using

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}. \tag{20}$$

The true positive (TP) is defined as the number of times when migrations are correctly triggered. Similarly, false negative (FN) consists of the times that migrations are triggered incorrectly. Higher TPR values demonstrate larger accuracy and vice versa.

C. Experimental Results

The experimental results are shown in Table IV. We use two different approaches to migration durations, i.e., past runtimes from Google's data (Past), prediction through linear or simple regression (SR), and boosted trees (BTs), i.e., XGBoost. These machine learning models with default parameters (tree depth of 6, learning rate of 0.3, regularization L1-α and L2-λ are set to 1, and γ is 0 for BT) and optimizers (Gradient Descent for SR) were assumed running at the edge cloud on a node that is responsible for triggering migration decisions. Moreover, all vehicles were assumed to run at a speed of 80KM/h to 120KM/h (with random change). We assume a simple 2×2 topology with four intersection points and each edge (road) is considerably long to simulate speed variations. We assume that each vehicle once entered in the square, can move in any direction (random selection) [22]. Vehicles are simulated through a poison process but the total number is restricted to ensure enough bandwidth. The speeds of the vehicles are increased when congestion drops to low levels and decreases vice versa. The accuracy is the percentage of migratable services which were moved to the destination within the estimated time. Once the service time is over (VM runtimes) the vehicles are assumed to reach their destination. Finally, VM sizes denote the service types (applications or workloads), which are randomly picked, for simulation purposes only. This should be noted that each VM denotes a connected vehicle to make consistency with the iFogSim simulator. The latency and migration durations were observed depending on the placement and migration policies. Similarly, increasing the network bandwidth essentially improves the migration process through reducing their durations.

We observed that the placement and migration techniques both have significant impacts over the latency, both in the cloud and fog, and network congestion; therefore, the number of migrations. Subsequently, a higher number of migrations increases the service execution time. The fewer number of migrations ensures short execution times and lower latencies. We also observed that a good prediction approach, such as BTs, might increase the number of migratable services; the more migratable services may mean higher probabilities of unsuitable migrations and, therefore, lower accuracies. In Table IV, ApMove is observed to decrease the TPR for the SR

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

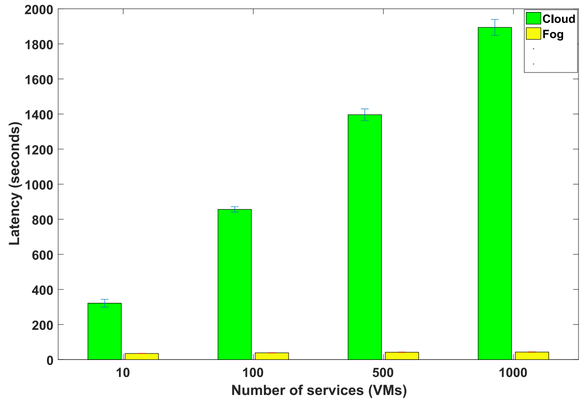

Fig. 3. Latencies observed in fog and cloud.

TABLE V
NETWORK TRAFFIC (KILOBYTES)

VMs	Cloud traffic		Fog traffic	
	SimpleMigrate	ApMove	SimpleMigrate	ApMove
10	2389	2298	1786	1702
100	3409	3367	1894	1893
500	4325	4103	2021	1991
1000	4943	4204	2190	2089

model, which suggests a significant decrease in the accuracy of regression predictions, i.e., 71.11%. We think that the churn in migration data may be one of the probable causes of this scenario. Fig. 3 shows the latencies that we observed in the fog infrastructure and cloud. We noted longer latencies for the cloud due to increasing distances and the number of running applications. However, fog latencies are significantly lower than the cloud and have lower impacts due to the number of services. The experiments were run 10 times and the error bars show the variations among the outcomes.

Table V shows the network traffic for various experiments. With the increasing number of services, a significant growth in the network traffic was observed. The proposed method ApMove decreases the network traffic through reducing the number of migrations. The higher number of migratable services decreases the accuracy and, subsequently, the TPR. ApMove can reduce approximately 36.42% migratable services compared to the AlwaysMigrate approach at comparable service execution times to the NoMigrate technique (∼1.01% loss in performance). For the NoMigrate approach, albeit we observed longer latencies in the fog; however, migration techniques could play a significant role in reducing the fog latencies. These outcomes are also dependent on the prediction approaches used for services' runtimes. For example, "SR" increases the number of migratable services than the "Past" and "BT" than the SR. Thus, a good prediction technique may not be always better than a worse method in producing outcomes. Besides, service placement policies have also shown significant impacts over the obtained results, which are not reported here.

As shown in Fig. 4, the network capacity has an impact on the end-to-end delay and migration durations; however, migration frequency is not affected. How often migrations are triggered is mostly affected by the migration approach. Furthermore, the migration ratio can negatively affect the response time. This should also be related to the available network bandwidth for migration traffic. This should be noted that when the same experiments with the same experimental parameters were carried out in containers (or microservices and functions), instead of VMs, we observed reduced latencies and quicker migration durations, therefore, smaller execution times (the best performance) [23]. Similarly, reduced network traffic, both in the remote cloud and fog infrastructure, further reduces the end-to-end delays. We believe, network traffic along with service execution times could also be minimized through using an approach that migrates less data, e.g., zip the data before transferring it over the network [11].

D. Results Discussion

The migration duration is significantly dependent on the VM capacity and transfer time of the network [24]. Moreover, the service latency depends over the distances and capacities of the links between cloud-fog and fog-vehicle. Subsequently, the latency relies on the migration strategy. There is a trade-off between migration durations and service latencies when service migrations are considered. On the one side, service migration aids in bringing services closer to cars, resulting in a low-latency value once the transfer is complete. Transferring service-related files to the target edge server, on the other hand, takes time. When the transferred files are enormous, the migration time can take quite a long time. However, if the service is not migrated, the packet must travel a considerable distance to reach the edge server or cloud that hosts the service's VMs, resulting in significant latency as the vehicle goes away.

The mobility of connected cars, or IoT devices, can have an impact on edge/fog computing performance, especially when they switch among edges often. Such service migration operations in a logical multitier computing infrastructure like edge/fog are dependent on: 1) the location of connected devices; 2) the direction of the mobility; 3) the car's speed; and 4) the recognition of a transitional edge to which the migratable service can be uploaded, with the notable exception of network area storage, and later on can be downloaded to continue running. The performance metrics of the edge/fog environment, such as network latency, application performance, and service QoS, can all change dramatically based on them. In order to see similar impacts, we implemented the mobility model from the MobFogSim simulator [10], [20]. Moreover, the mobility data was taken from real-time traces in terms of latitude and longitude values [25]. Since, iFogSim2 supports microservices; therefore, we assumed that services are running in a container/microservice instead of VMs. We modified the migration model to account for: 1) migration costs (in terms of performance loss); 2) migrations among edges; and 3) migrations from fog to cloud and vice versa. The service was assumed to run in three different modules, i.e., client, processing, and storage. All other parameters, including cars speeds, networks, and data sets, were kept the same as described in [8] and [10].

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

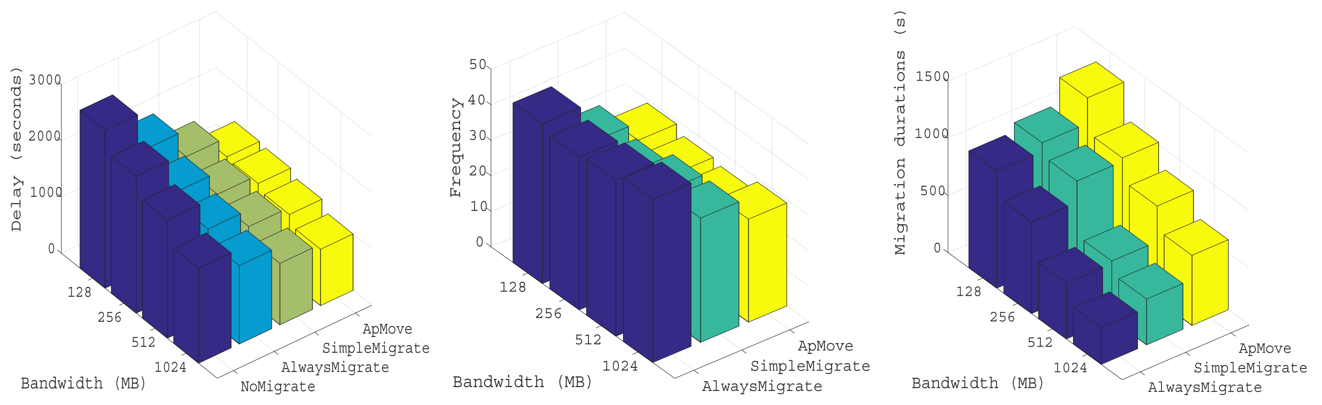

Fig. 4. Results in terms of end-to-end delay (latency), migration frequency, and durations for different network bandwidth.

TABLE VI
EXPERIMENTAL RESULTS FOR THE PROPOSED APMOVE MIGRATION TECHNIQUE IN TERMS OF VARIATIONS IN CAR SPEED, AND NETWORK TYPE [THE VALUES ARE AVERAGE AND THE ± DENOTE THE STANDARD DEVIATION ACROSS TEN EXPERIMENTS, A SLOW NETWORK HAS LOWER BANDWIDTH AND LOWER TRANSFER RATES THAN THE MEDIUM AND FAST NETWORKS]

Speed (kmph)	Network type	Migration time (seconds)	Number of migrations	Latency (seconds)	Execution time (seconds)	Network traffic (Mbps)
	slow	567.78±13.7	43±2	1689.78±45.78	26934±109.45	2093.45±55.65
50-60	medium	489.9±9.21	48±1	732.65±12.6	26542±107.33	1893.12±36.78
	fast	273.67±5.32	57±2	92.64±8.59	26128±101.9	1253.09±12.98
	slow	589.09±12.89	38±2	1763.12±43.87	27002±121.4	2102.3±51.09
80-90	medium	433.34±12.01	45±1	810.09±14.78	26992±114.56	1883.9±31.32
	fast	268.32±3.44	52±2	105.03±11.23	27005±123.89	1267.97±10.58
	slow	555.68±6.49	36±3	1901.65±56.89	27012±19.67	2098.4±43.95
110-120	medium	487.08±5.01	39±1	854.2±12.98	27009±17.74	1892.89±18.88
	fast	298.06±2.21	46±2	115.4±6.31	27011±18.46	1309.11±9.9

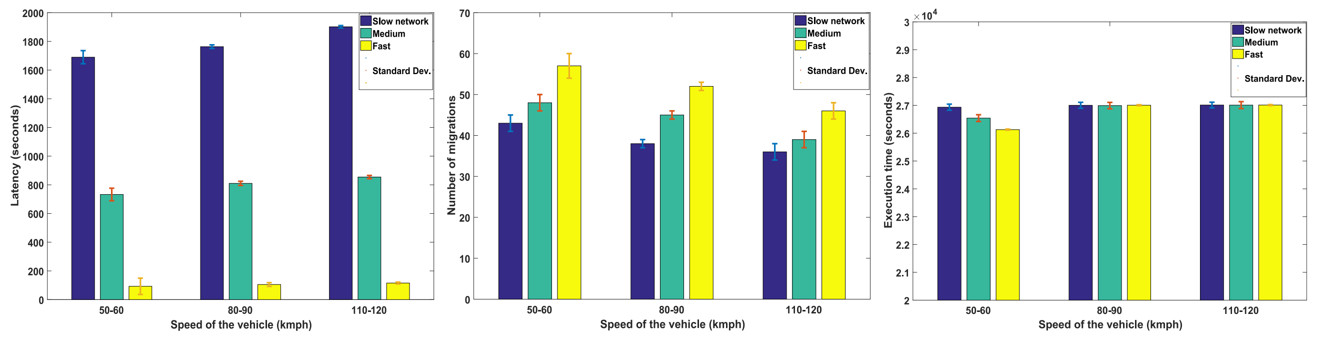

Fig. 5. Results for the proposed ApMove methodology in terms of latency, number of migrations, and service execution times for different network types and vehicle speeds [the service execution time is nonsignificantly affected by the car speed and network type].

TABLE VII
COMPARISON OF THE APMOVE WITH OTHER STATE-OF-THE-ART METHODS [THE BEST VALUES ARE SHOWN BOLD]

Method	Energy consumption (KWh)		Execution time (seconds)	Latency (seconds)	Migration time (seconds)	Costs ($)	Accuracy (%)	TPR
	Edge	Cloud						
NTM	1476.34	9034.04	27943.98	1017.78	**278.45±2.93**	123.45	93.02	0.969
VVMM-MA	1553.8	10127.5	31128.8	1189.56	388.2±3.04	157.8	89.59	0.958
MDWLAM	1499.1	9845.78	30032.7	1156.36	369.3±3.14	155.6	91.87	0.97
ApMove	**1370.7**	**8692.88**	**27183.33**	**993.8**	281.21±1.02	**115.08**	**95.56**	**0.936**

The results are shown in Table VI and Fig. 5. We observed that slower networks may increase migration times but may reduce the total number of migrations and vice versa. Furthermore, slower networks increase the service latency and the network traffic. Interestingly, the car speeds may significantly affect the total number of triggered migrations but the migrations durations are trivially affected. We noted that when cars are moving faster, up to a particular range, the service latency may be higher than if they keep moving slower. The latency variations could be reduced through

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

improving the network speeds. Network traffic is affected by the total number of migrations, migration scheme, speed of the vehicles, mobility patterns, and network bandwidth. We also noted that service execution times are also dependent on the users' mobility, speed, and network traffic. However, as shown in Fig. 5, the service execution time is nonsignificantly affected by the car speed and type of the network. These little variations are probably due to the latency that occurred due to differences in the bandwidth of the networks. Our generalization of the outcomes suggests that at ∼0.004%–2.99% performance loss, the service response time could be improved significantly, i.e., 7.45%–9.04%. Furthermore, approximately 17.39% migrations were avoided as compared to the simple migration approach. The cost savings due to service execution times were observed from 4.3% to 11.63%. We believe these savings will be higher for more congested networks.

We assume that each car runs a single service (inside a VM), therefore the number of migrations and VMs relate to the number of vehicles. As shown in Table V, the number of VMs increases the network traffic that subsequently has impacts on the network performance, i.e., bandwidth. Furthermore, as shown in Fig. 5, bandwidth has an impact on the number of migrations, service performance in terms of latency, and execution time.

E. Comparison With the Closest Rivals

In this section, we compare ApMove with three state-of-the-art methods, i.e., NIB task migration technique (NTM) [5], VVMM-MA, and MDWLAM [22], in terms of energy consumption, costs, and performance (latency, execution time). Furthermore, we also compare these methods in terms of migration durations. The experimental parameters were kept the same as described earlier in Section IV-A. The costs are computed with respect to the service execution times using the VM prices given in [12]. The results are shown in Table VII and Fig. 6. We observed that the NTM approach has the least migration times; however, its performance in terms of latency and execution time is worse than the ApMove method. The migration time is dependent on the service being migrated and the time it is triggered for migration. On average, our method is approximately 6.78% more cost-efficient and 11.67% more energy-efficient than the NTM approach. We also observed almost comparable migration times of the ApMove and NTM, shown with overlapping standard deviations (±) in Table VII, in particular, when vehicles are either moving fast or when the bandwidth is increased. The ApMove policy triggers only those migrations whose energy and performance costs can be recovered back, i.e., migrations to more energy and performance-efficient target servers than the source servers. Furthermore, frequent service migrations are avoided between edges for fast-moving cars which leads to performance and cost improvements.

F. Migrations Statistics

As mentioned in Section I, besides interedges migration we also account for the migration to a remote cloud (intraclouds).

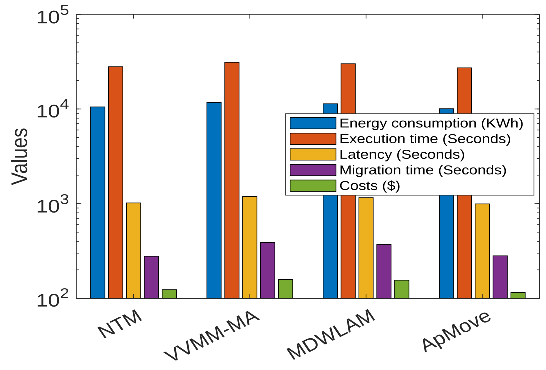

Fig. 6. Comparison of the ApMove with other state-of-the-art methods [lower values denote best results].

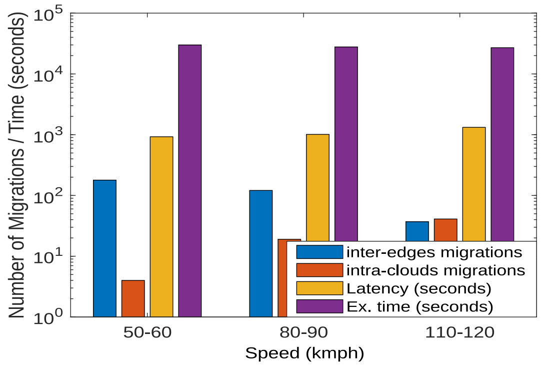

Fig. 7. Impacts of interedges and intraclouds migrations on vehicle's speed, service latency, and execution time.

The necessity of such a choice is shown by experiments, in terms of, the ratio of all the situations where migration to a remote cloud is needed and beneficial. Improved performance is one of the benefits such a choice can bring. To study the impacts of intracloud migrations, the speeds of cars were varied in the simulations and their services were explicitly migrated to the remote cloud. We assume that the remote cloud covers 3 hops, i.e., four edge clouds from a distance point of view. All other parameters, including the network bandwidth, were assumed the same as discussed in Section IV-A. The obtained outcomes are shown in Fig. 7. In fact, it is more performance-efficient to run services on edges (in terms of low latency); however, we observed an increased number of migrations for vehicles with high speeds. Moreover, frequent migrations increase service execution times, therefore, migration decisions and users' monetary costs. Similarly, during the intraclouds migration, the number of migratable services is significantly reduced, i.e., approximately 69.42%–79.33% times reduced as compared to the interedges migration. We observed that approximately 2.93%–9.87% service execution times could be reduced; however, these benefits may be achieved at a cost of 2.86%–3.06% increased latency.

V. Related Work

Arthurs et al. [2] have demonstrated a comprehensive review of the literature on cloud computing's application in ITS and connected vehicles, as well as, taxonomies and use

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

cases. Moreover, they have identified areas in which more research is needed to enable vehicles and ITS to employ edge cloud computing in a completely managed, intelligent, and automated manner. The importance and problems of software defined networks (SDNs) for better network management in smart and connected automobiles are discussed in [26]. The authors have also demonstrated the importance of embracing cloud, edge, and fog computing for processing huge quantities of real-time data produced by a network of interconnected vehicles, as well as, the challenges that come with this emerging technology. Garai et al. [27] have proposed a cloud Communication-as-a-Service (CaaS) in order to: 1) enable continuous communication to vehicles beyond the range of roadside units; 2) ensure QoS in terms of throughput, delay, response time, and packet loss rate; and 3) deal with resource constraints in vehicular networks. Moreover, a vehicular cloud architecture (V-Cloud) built of three layers is presented to implement these solutions. The vehicular cloudlet layer is designed through a collection of vehicles that are grouped and connected to the network in such a way that forms a tree topology. Similarly, the second layer is dubbed to a roadside cloudlet that is a, local, cloud created between a group of RSUs. Finally, the central remote cloud is created by a group of servers on the Internet, is the third tier of the V-Cloud architecture.

Yao et al. [28] have elaborated on the well-known VM migration problem in a roadside cloudlet-based vehicular network, determining: 1) whether a VM should be migrated or not and 2) where a VM should be migrated, in order to reduce the overall network cost for both VM migration and normal data traffic. The authors have formulated the problem as a mixed-integer quadratic programming (MIQP) problem after treating it as a static offline VM placement challenge. The fog computing architecture is described in [29], along with its various services and applications. Then, with a focus on service and resource availability, the authors have explored security and privacy challenges in fog computing environments. Virtualization is a critical technique in edge, fog, and cloud computing because it allows VMs to cohabit and share resources on a real server (host). These VMs could be targeted by malware, or the physical machine accommodating them might suffer a system failure, resulting in the loss of services and resources. In order to assess whether to proceed the stop-and-copy stage during a system failure or an assault on an edge node, a smart, conceptual, precopy live migration technique is demonstrated, which predicts the downtime after each and every iteration.

Xu et al. [5] argued that exploiting 6G mobile networks has the potential to reduce communications delays, in particular, for the execution of latency-critical and real-time tasks. The 6G-enabled NIBs, i.e., network in boxes, are installed in connected cars, for instance, can connect with MEC servers or dissimilar NIBs present in other cars. Albeit, these NIBs can deliver adaptable and dynamic computing resources to run real-time Internet of Vehicle (IoV) applications, however, the communication and computational operations have high-energy costs. The authors have successfully built an NTM for IoV in order to obtain an optimal balance, to control the existing tradeoff, between energy usage and time cost during the service transfer. In [7], researchers have elaborated the possible benefits of networked autonomous cars through looking at five different use cases: 1) vehicle platooning; 2) lane switching; 3) intersection management; 4) road friction estimation; and 5) energy management. According to [7], while connectivity can significantly improve the connectivity of autonomous vehicles and help the development of existing transportation effectiveness, the level of benefits that can be realized is dependent upon several factors, including connected vehicle penetration rate, traffic scenarios, and the process of amplifying off-board data into vehicle control frameworks.

The research conducted in [22] offers the VVMM-U (uniform), VVMM-LW (the least workload), VVMM-MA (mobility aware), and MDWLAM strategies for vehicular VM migration (mobility and destination workload aware migration). Simulations with varied levels of vehicular traffic congestion, VM sizes, and levels of load restriction are used to evaluate their performance against a set of metrics. The most advanced technique (MDWLAM) considers both the original host's workload and mobility, as well as the prospective destinations. A legitimate destination will have adequate time to accept the workload and, if required, relocate the increased load as a result of this. Mozaffari et al. [30] give an introduction of the general topic of vehicle behavior prediction and discuss its obstacles. Then, it classifies and reviews the most current solutions based on deep learning approaches using three criteria: 1) input representation; 2) type of output; and 3) prediction approach. The research also assesses the efficacy of a variety of well-known remedies, identifies research gaps, and suggests new research possibilities [31], [32].

AI-Quraan et al. [33] have provided an overview of the fundamentals and enabling technologies of federated learning FL. Furthermore, a comprehensive analysis is offered that details several FL applications in wireless networks, as well as, their challenges and limitations. Beyond 5G and 6G communication technologies, the efficacy of FL is being investigated. The goal of this survey is to present an overview of the current state of FL applications in key wireless technologies, which will serve as a foundation for gaining a thorough grasp of the subject. Finally, the authors suggest a path forward for future research. Kuutti et al. [6] have extended the investigation by looking at strategies that take advantage of off-board data collected from V2X communication channels in addition to vehicle sensory data. The findings illustrate that adding off-board information with sensor information has the capability to potentially develop low-cost, robust localization systems that could be highly accurate; nevertheless, their performance is directly proportional to the speed and rate at which adjacent connected vehicles or infrastructure are connected, and also the quality of network service [14]. The "Follow-Me-Cloud" system uses a Markov-process-based decision method to make cost-effective, performance-optimized service choices, while two separate methods based on software-defined networking technologies or the locator/identifier separation protocol are suggested to guarantee service continuity and uninterrupted execution [21]. The summary of the comparison between our proposed technique ApMove and other closely related

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

TABLE VIII
SUMMARY OF THE RELATED WORKS

Matching criteria	related work								ApMove
	[5]	[7]	[21]	[22]	[24]	[27]	[28]	[29]	
Framework	✓	✓	✓			✓			✓
Migration prediction							✓	✓	✓
Mobility patterns		✓		✓	✓	✓			✓
Service performance	✓		✓		✓	✓		✓	✓
Latency	✓	✓	✓	✓	✓	✓	✓		✓
Algorithmic criteria									
Service placement	✓		✓	✓	✓		✓		✓
Technique	Evolutionary		Markov Chain	Workload Aware	Distance Latency	Latency Aware	Cost Weights FF, BF		FillUp
Energy efficiency	✓	✓							✓
Previous runtimes									✓
Migrations									
Among edges	✓				✓		✓	✓	✓
Edges to cloud									✓

works is given in Table VIII. We believe, the comparison would also help readers to quickly identify gaps for further research.

VI. Conclusion

Real-time applications for connected cars may develop sensitivity to the quality of networks, for instance, longer latencies between services and vehicles. As a result, their needs may be fulfilled by the new fog technology, which lets calculations to be performed at the network's edge. In this article, we suggested a mathematical model that can be used to migrate running services across various regions for connected vehicles. Moreover, we computed an appropriate time in order to start migrating the running application to the destination server. Through a number of simulations, over realtime workload traces from Google, we showed that the proposed strategy migrates running services while ensuring their expected levels of performance and minimizing the response time. The suggested solution can help service providers become more cost- and energy-efficient. Our approach may be used by automakers to improve the adaptability and dependability of connected and self-driving cars, which will increase customer satisfaction by optimizing resource use, avoiding downtime, and cutting operating expenses. Additionally, consumers who use autonomous car services stand to benefit from increased accessibility and dependability of transportation services, which might result in cheaper costs and better convenience.

VII. Future Work

In the future, we will use more accurate models to characterize the network congestion, performance degradation, latencies, and infrastructures' heterogeneities. Similarly, we will study energy efficiency in cloud/fog/edge infrastructure through characterizing different use cases. In addition, the connected services and their qualities are mainly dependent on various service providers, such as resources, applications, and networks. These providers will have their own objectives and, perhaps, may compete for optimizing their desirable objectives. There are other questions that should be investigated in the future: 1) How often might migration be needed as a consequence of topology? and 2) What assumptions should be used with regard to cell size, or proximity of edge servers to one or more radio masts? We will study the impact of dynamic threshold values to trigger migrations to edge or cloud on migration statistics and outcomes. We will also investigate the impact of performance improvements in terms of the number of vehicles that could be serviced at a time because this would be valuable to serve multiple connected vehicles simultaneously. Similarly, if service execution is slower or faster on different hardware (edge clouds), it could imply that more or fewer resources are required to achieve the same outcome on the target edge cloud. Understanding these variations is crucial because it helps determine the resources needed for equivalent performance on the target edges. We will use more advanced methods to model automobile movement within edge networks, considering traffic conditions, road geometry, geographical restrictions, traffic flow dynamics, road layout, driver behavior, and using large data sources. Finally, we will build a small real platform to study the impacts of the proposed approach.

References

[1] L. Gillam, K. Katsaros, M. Dianati, and A. Mouzakitis, "Exploring edges for connected and autonomous driving," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2018, pp. 148–153.

[2] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, "A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6206–6221, Jul. 2022.

[3] L. Gillam, "Will cloud gain an edge, or, CLOSER, to the edge," in *Proc. Int. Conf. Cloud Comput. Services Sci.*, 2018, pp. 24–39.

[4] S. K. Datta, J. Haerri, C. Bonnet, and R. F. Da Costa, "Vehicles as connected resources: Opportunities and challenges for the future," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 26–35, Jun. 2017.

[5] X. Xu, L. Yao, M. Bilal, S. Wan, F. Dai, and K.-K. R. Choo, "Service migration across edge devices in 6G-enabled Internet of Vehicles networks," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1930–1937, Feb. 2022.

[6] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, Apr. 2018.

[7] U. Montanaro et al., "Towards connected autonomous driving: Review of use-cases," *Veh. Syst. Dyn.*, vol. 57, no. 6, pp. 779–814, 2019.

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.

[8] D. Gonçalves, C. Puliafito, E. Mingozzi, O. Rana, L. Bittencourt, and E. Madeira, "Dynamic network slicing in fog computing for mobile users in MobFogSim," in *Proc. IEEE/ACM 13th Int. Conf. Utility Cloud Comput. (UCC)*, 2020, pp. 237–246.

[9] M. Tirmazi et al., "Borg: The next generation," in *Proc. 15th Eur. Conf. Comput. Syst.*, 2020, pp. 1–14.

[10] C. Puliafito et al., "MobFogSim: Simulation of mobility and migration for fog computing," *Simulat. Model. Pract. Theory*, vol. 101, May 2020, Art. no. 102062.

[11] M. Zakarya and L. Gillam, "An energy aware cost recovery approach for virtual machine migration," in *Proc. Int. Conf. Econ. Grids, Clouds, Syst., Services*, 2016, pp. 175–190.

[12] M. Zakarya, "An extended energy-aware cost recovery approach for virtual machine migration," *IEEE Syst. J.*, vol. 13, no. 2, pp. 1466–1477, Jun. 2019.

[13] B. Shi and H. Shen, "Memory/disk operation aware lightweight VM live migration across data-centers with low performance impact," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 334–342.

[14] J. Li et al., "Service migration in fog computing enabled cellular networks to support real-time vehicular communications," *IEEE Access*, vol. 7, pp. 13704–13714, 2019.

[15] A. A. Khan, M. Zakarya, R. Buyya, R. Khan, M. Khan, and O. Rana, "An energy and performance aware consolidation technique for containerized datacenters," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1305–1322, Oct.–Nov. 2021.

[16] I. El-Taani, M.-C. Boukala, S. Bouzefrane, and A. I. Amrous, "Robust approach for host-overload detection based on dynamic safety parameter," in *Proc. 9th Int. Conf. Future Internet Things Cloud (FiCloud)*, 2022, pp. 266–271.

[17] F. H. Bappy, T. Islam, T. S. Zaman, R. Hasan, and C. Caicedo, "A deep dive into the Google cluster workload traces: Analyzing the application failure characteristics and user behaviors," in *Proc. 10th Int. Conf. Future Internet Things Cloud (FiCloud)*, 2023, pp. 103–108.

[18] Z. Wang, X. Zhao, and Z. Xu, "Offline mapping for autonomous vehicles with low-cost sensors," *Comput. Elect. Eng.*, vol. 82, Mar. 2020, Art. no. 106552.

[19] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017.

[20] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "IFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments," 2021, *arXiv:2109.05636*.

[21] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 369–382, Apr.–Jun. 2019.

[22] T. K. Refaat, B. Kantarci, and H. T. Mouftah, "Virtual machine migration and management for vehicular clouds," *Veh. Commun.*, vol. 4, pp. 47–56, Apr. 2016.

[23] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate IoT edge computing with lightweight virtualization," *IEEE Netw.*, vol. 32, no. 1, pp. 102–111, Jan./Feb. 2018.

[24] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "MyiFogSim: A simulator for virtual machine migration in fog computing," in *Proc. Companion Proc. 10th Int. Conf. Utility Cloud Comput.*, 2017, pp. 47–52.

[25] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO traffic (LuST) scenario: 24 hours of mobility for vehicular networking research," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, 2015, pp. 1–8.

[26] R. M. Shukla, S. Sengupta, and M. Chatterjee, "Software-defined network and cloud-edge collaboration for smart and connected vehicles," in *Proc. Workshop Program 19th Int. Conf. Distrib. Comput. Netw.*, 2018, pp. 1–6.

[27] M. Garai, S. Rekhis, and N. Boudriga, "Communication as a service for cloud VANETs," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2015, pp. 371–377.

[28] H. Yao, C. Bai, D. Zeng, Q. Liang, and Y. Fan, "Migrate or not? Exploring virtual machine migration in roadside cloudlet-based vehicular cloud," *Concurr. Comput., Pract. Exp.*, vol. 27, no. 18, pp. 5780–5792, 2015.

[29] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K.-K. R. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework," *IEEE Access*, vol. 5, pp. 8284–8300, 2017.

[30] S. Mozaffari, O. Y. Al-Jarrah, M. Dianati, P. Jennings, and A. Mouzakitis, "Deep learning-based vehicle behavior prediction for autonomous driving applications: A review," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 33–47, Jan. 2022.

[31] R. A. Addad, T. Taleb, H. Flinck, M. Bagaa, and D. Dutra, "Network slice mobility in next generation mobile systems: Challenges and potential solutions," *IEEE Netw.*, vol. 34, no. 1, pp. 84–93, Jan./Feb. 2020.

[32] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 87–93, Feb. 2019.

[33] M. Al-Quraan et al., "Edge-native intelligence for 6G communications driven by federated learning: A survey of trends and challenges," 2021, *arXiv:2111.07392*.

Authorized licensed use limited to: University of Melbourne. Downloaded on August 25,2024 at 05:15:26 UTC from IEEE Xplore. Restrictions apply.