APMOVE: A Service Migration Technique for Connected and Autonomous Vehicles

Muhammad Zakarya, Lee Gillam, Ayaz Ali Khan, Omer Rana, Rajkumar Buyya

Abstract—Multi-access 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 functions. A cloudlet is similar to a MEC in that it consists of many servers that provide real-time, low latency, compute services to connected users in the close proximity. In connected vehicles, services may be provisioned from the cloud or edge infrastructure 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 and seamless manner so that performance and connectivity is not affected negatively. In this paper, we propose an affective strategy for migrating connected users' services from one edge node to another or, more likely, to a remote cloud in a multi-access edge cloud. A mathematical model is, then, presented to estimate the expected time to migrate the services based on the laws of motion, instantaneous speed, velocity, and time. Our evaluations and results, based one real workload traces, accounting for mobility patterns, suggest that the proposed strategy "ApMove" migrates autonomous vehicles' services while ensuring their expected levels of performance ($\sim$0.004% – 2.99% loss), reduced runtimes (or at least comparable to the no migration strategy) therefore, users' costs ($\sim$4.3% – 11.63%), and minimizing the response time ($\sim$7.45% – 9.04%). Furthermore, approximately 17.39% migrations are avoided. We also study the impacts of variations in car's speed and network transfer rates on service migrations durations, frequencies, latencies, and service execution times.

Index Terms—Edge cloud, service migration, connected vehicles, inetrnet of things

1 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 being stored, processed, and used for various objectives. However, there are various challenges e.g. the risk of becoming the data useless on its way to the cloud due to longer distances and delays. To cope with the challenges associated to the cloud computing and network, service providers are now encouraging for working 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 is generating a lot of buzzes since it enhances service agility and performance of realtime services in terms of response time.

Multi-acces edge computing systems (MECs) offer the capabilities of cloud computing closer to the radio access network (RAN), in the cotext 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 congestions, crowd, routing decisions, and management [3]. An MEC is a cloud server running at the edge of a mobile network which 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 performance of these real-time car services; therefore, intelligent decisions should be very important. Most importantly, whether to migrate the services are not; if the car is already in 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 a MEC system. The experimentation should be completed using real datasets of service migration, in a simulated fog computing environment [5].

In this paper, 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. Such decisions are taken based on the vehicle's speed, distance traveled, and more likely its destination that could be predicted using machine learning models. For example, if a vehicle is enough fast that results 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]. However, it will essentially affect the latency and response time; thus,

- *M. Zakarya is with the Department of Computer Science, Abdul Wali Khan University, Pakistan. L. Gillam is with the University of Surrey, Guildford, UK. A.A. Khan is with the University of Lakki Marwat, Khyber Pakhtunkhwa, Pakistan. O. Rana is with the University of Cardiff, UK. R. Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, University of Melbourne, Australia.*
  E-mail(s): mohd.zakarya@awkum.edu.pk, l.gillam@surrey.ac.uk, ayazkhan@ulm.edu.pk, ranaof@cardiff.ac.uk, rbuyya@unimelb.edu.au

traffic crowd management. Similarly, if services are being migrated among edge nodes, then, each migration will have a negative impact on the response time (migration down time). Therefore, it would be essential to take appropriate migration decisions on 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 de-centralised 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 centralised cloud. The research will investigate various methodologies such as: (i) training the model on the edge and then predict; (ii) training the model on the cloud and then predict; and (iii) training the model on the cloud and then predict on the edge. The major contributions of our research are:

- we investigate whether a user is, highly, likely to keep moving quickly between edges or not;
- during mobility, when it will be effective to start migrating the running service or application to the target edge/cloud;
- an effective strategy is suggested to migrate connected users' services from one edge to another or, more likely, to a remote cloud in a MEC platform;
- a mathematical model of the service migration technique is presented that is based on the motion's laws;
- a probabilistic and a machine learning based models are used to estimate the migration time; and
- Google's and mobility datasets are used to validate the model through plausible assumptions and simulations [9].

The rest of the paper is organized as follows. In Sec. 2, we formulate the service migration problem in the context of connected cars. In Sec. 3, we propose ApMove, an approach to migrate live services across various edges and/or remote clouds for connected vehicles. In Sec. 4, we validate and evaluate the performance of the suggested technique using real workload traces from the Google clusters and cars mobility dataset. In Sec. 5, we provide an outline of the relevant work. Lastly, Sec. 6 brings the work to a close and discusses future research directions.

2 Background and Problem Description

Fig. 1 shows the system model for the service migration technique that comprises a number of edge clouds ($EC = ec_1, ec_2, ..., ec_n$) and moving vehicles. Furthermore, a special type of device is integrated in each connected or moving vehicle ($V = v_1, v_2, ..., v_r$). Moreover, each EC consists of several numbers of servers ($ES = es_1, es_2, ..., es_n$). We assume each edge server and connected device consist of sufficient number of VM or container instances to execute the services [5]. Usually, Function as a Service (FaaS) 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 ($R = r_1, r_2, ..., r_u$). All the ECs are connected to a remote virtualised 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 1.

TABLE 1: List of mathematical notations

Notation	Description
H	List of servers so that $h \in H$
VM	List of VMs so that $vm \in VM$
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
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)

The displacement is the change in position x of a moving object and can be computed through $\triangle x = x_f - x_i$, where $\triangle x$ 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 $\triangle t = t_f - t_i$, 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 that is the ration between $\triangle x$ and $\triangle t$. However, a car cannot travel at a constant speed, velocity, and the displacements may variate across the entire route. Therefore, we assume its instantaneous velocity v' while assuming $t_i = t$ and $t_f = t + \triangle t$ such that limit $\triangle t \to 0$.

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

Through dividing the total distance traveled by the elapsed time, we can compute the average speed s. However, the instantaneous speed s' the absolute value of the instantaneous velocity $|v'|$. The acceleration is the rate at which velocity changes i.e. the ration between the velocity $\triangle v$ and time $\triangle t$. Actually, the acceleration denote the rate of change in speed of the moving object. The instantaneous acceleration is given by:

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

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 distance moved over time and is given by:

$$speed\ (s) = \frac{Distance}{time} = \frac{x_2 - x_1}{t_2 - t_1} = \frac{\triangle x}{\triangle t} \tag{3}$$

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 duration for migration completion, as illustrated in Sec. 3.1. Using the above equation, if we know the speed and the distance traveled by a moving object, we can find its time at a particular position.

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

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

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

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 Eq. 1 and Eq. 2 to find the velocity and position, respectively:

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

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

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.

3 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 to from the source edge to the target edge. There are two situations: (i) when the coverage areas overlap; and (ii) 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 non-overlapping coverage 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 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 [10], [11]. 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 as way that:

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

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 earlier 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 on a remote cloud that covers large geographical area. The t_{off} can be estimated using Eq. 9.

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

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

$$t_{mig} = \frac{FaaS_{size}}{B} \tag{10}$$

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 Alg. 1. The migration duration and downtime (performance loss) is computed through the mathematical model as demonstrated in Sec. 3.1.

3.1 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 copy the dirty pages at previous round of transmission iteratively. Consider that there are n rounds, which completes the pre-copy 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 is given by:

$$V_i = \begin{cases} V_{mem} & \text{if i = 0} \\ D.T_{i-1} & \text{if i > 0} \end{cases} \tag{11}$$

$$T_i = \frac{D.T_{i-1}}{R} = \frac{V_{men}.D^i}{R^{i+1}} \tag{12}$$

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

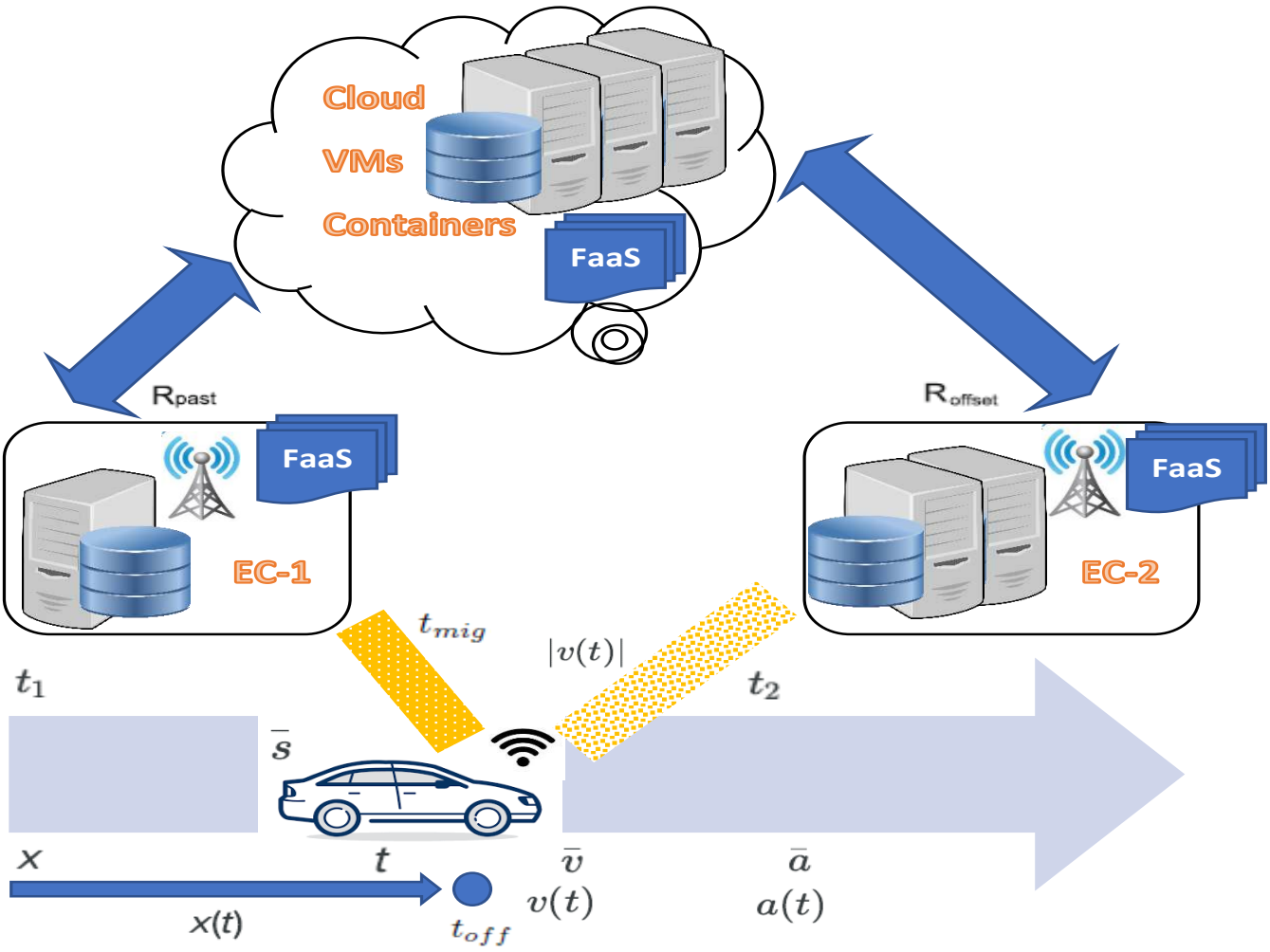

Fig. 1: The proposed service migration strategy for autonomous cars

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

$$\omega = \frac{D}{R} \tag{13}$$

Combining equations 11, 12 and 13, we get:

$$V_i = V_{mem}.\omega^i \tag{14}$$

The total network traffic is given by:

$$V_{mig} = \sum_{i=0}^{n} V_i = V_{mem}.\frac{1-\omega^{n+1}}{1-\omega} \tag{15}$$

The total migration time is given by:

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

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

$$T_{down} = T_n + T_{resume} \tag{17}$$

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

$$n = log_{\omega}.\frac{V_{th}}{V_{mem}} \tag{18}$$

From the above studies, we determine that a VM having 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 pre-copy 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 the migration will not be accomplished. We do not consider such situation in our modeling, but Xen have solved this issue using the writable working set technique. The pages which 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 [12]. Earlier studies have shown that migration durations has an important impact over the network traffic, quality of service, and service performance [13]. In this work, we assume stateful migration which means that the entire VM is

Algorithm 1: Service migration technique

Input: List of MECs (C), Service migration requests (M), Velocity $v'(t)$
Output: Service placement

for *each* $m \in M$ **do**
  for each $m \in M$ compute t_{off} using Eq. 9 ;
  if $t_{mig} + t_{off} \leq T_{mig}$ **then**
    for *each* $c \in C$ **do**
      if c *has enough resources for* m **then**
        allocate m to c;
        break ;
      end if
    end for
    if m *did not fit in any available* c **then**
      look for another c and allocate m;
    else
      "m cannot be allocated";
      "push the m request into Q (queue)";
    end if
  end if
end for
return *output*

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 [11].

4 Performance Evaluation

In order to evaluate the feasibility and performance of the proposed model, we use different approaches to service placement and migration [14]. Furthermore, we assume the relocation problem as a binpacking 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 pre-defined threshold time, which means that the vehicle is traveling fast enough, we relocate that service to the remote cloud. We use the Google dataset for service execution times and migration statistics [9]; and the cars mobility data from [15]. The Google's dataset, which is an extended version of a previous 2011 dataset, includes details about job scheduling from 8 heterogeneous Google clusters for May 2019.

4.0.1 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 connected to the main cloud through a 1GB/s network cable. There are 100 servers in the datacenter that belongs to five different CPU architectures, i.e. 20 servers of each type, as shown in Table. 2. Moreover, each edge server also belong 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 3 shows the frequency of VMs in vCPUs (cores), which were translated to ECUs (EC2 Compute Units) 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 [16]. In order to be compatible with the iFogSim [15], [16], [17], the speeds of different hosts and Containers, VMs were 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 [18]. In our previous work [14], we have described performance details for numerous workloads or applications which are operating on these hosts.

TABLE 2: Servers characteristics for simulated MEC setup [ECU = CPU speed (GHz) $\times$ 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 3: Various types of VM instances and their characteristics – MEM means memory & vCPU denotes a hyper-threaded 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

4.0.2 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 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 Eq. 19.

$$TPR = \frac{TP}{TP + FN} \tag{19}$$

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 migration are triggered

incorrectly. Higher TPR values demonstrate larger accuracy and vice versa.

4.1 Experimental Results

The experimental results are shown in Table 4. We use two different approaches to migration durations i.e., past runtimes from the Google's data (Past), prediction through simple regression (SR), and Boosted Trees (BT). Moreover, all vehicles were assumed running at speed of 80KM/h to 120KM/h (with random change). We assume a simple 2×2 topology with 4 intersection point and each edge (road) is considerably long to simulate speed variations. We assume that each vehicle once entered in the square, it can move in any direction (random selection) [19]. Vehicles are simulated through a poison process but the total number is restricted to ensure enough bandwidth. The speed of the vehicles are increased when congestion drops to low level and decreased vice versa. The accuracy is the percentage of migratable services whose were moved to the destination within the estimated time. Once the service time is over (VMs runtimes) the vehicles are assumed to reach their destination. Finally, VMs 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 dependent 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 cloud and fig, and network congestion; therefore, number of migrations. Subsequently, 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 boosted trees, might increase the number of migratable services; the more migratable services may mean higher probabilities of unsuitable migrations, therefore, lower accuracies. Fig. 2 shows the latencies which we observed in the fog infrastructure and cloud. We noted longer latencies for cloud due to increasing distances and number of running applications. However, fog latencies are significantly lower than the cloud and has lower impacts due to number of services. The experiments were run 10 times and the error bars show the variations among the outcomes.

Table 5 shows the network traffic for various experiments. With increasing number of services, a significant growth for the network traffic was observed. The proposed method "ApMove" decreases the network traffic through reducing the number of migrations. Higher number of migratable services decreases the accuracy and, subsequently, the TPR. ApMove can reduce approximately 36.42% migratable services compared to AlwaysMigrate approach at comparabale service execution times to the NoMigrate technique ($\sim$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,

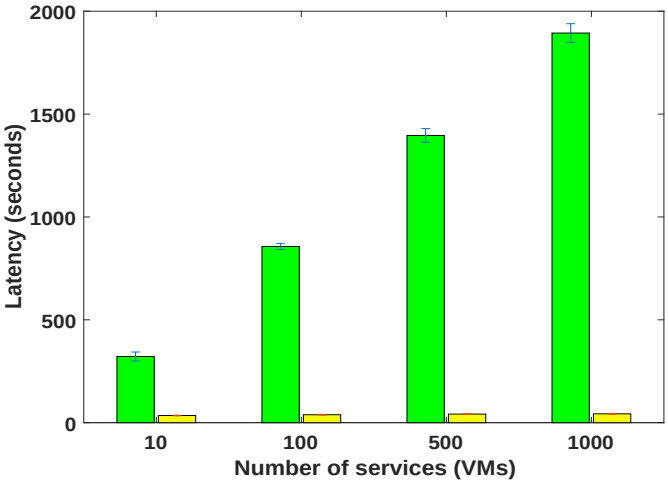

Fig. 2: Latencies observed in fog and cloud

"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. 3, 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, 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 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) [20]. 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 [10].

4.2 Results Discussion

The migration duration is significantly dependent on the VM capacity and transfer time of the network [21]. 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

TABLE 4: 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; exec. time is the sum of service runtime and latency

Service placement	Migration approach	Runtime approach	Migratable services %	Accuracy %	Latency (seconds) Fog	Latency (seconds) Cloud	Total migrations	Service exec. 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

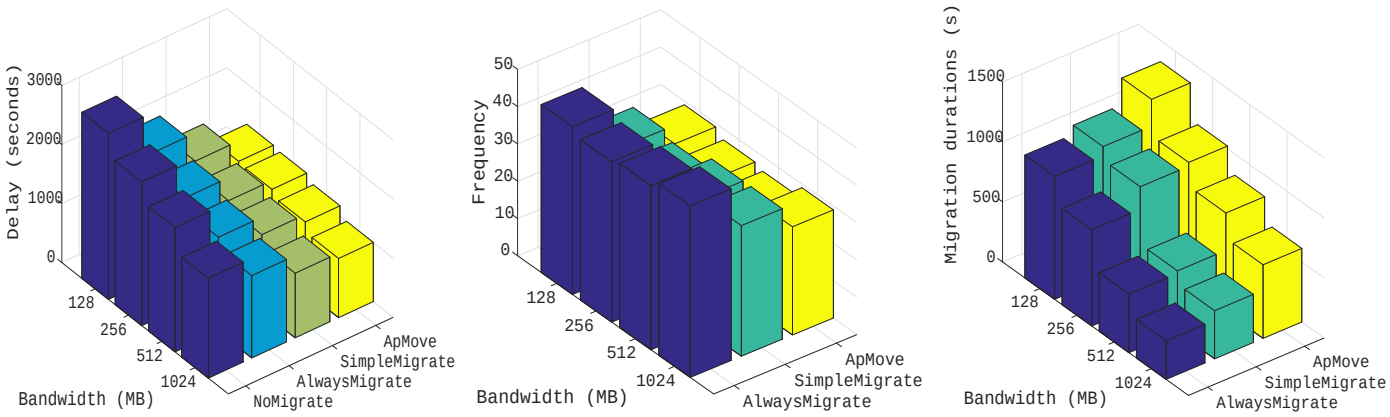

Fig. 3: Results in terms of end to end delay (latency), migration frequency, and durations for different network bandwidth

TABLE 5: Network traffic (Killobytes)

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

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 multi-tier computing infrastructure like edge/fog are dependent on: (i) the location of connected devices; (ii) the direction of the mobility; (iii) the car's speed; and (iv) 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 [15], [17]. Moreover, the mobility data was taken from realtime traces in terms of latitude and longitude values [22]. 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: (i) migration costs (in terms of performance loss); (ii) migrations among edges; and (iii) migrations from fog to cloud and vice versa. The service was assumed running in three different modules i.e. client, processing, and storage. All other parameters including cars speeds, networks, and datasets were kept the same as described in [8], [15].

The results are shown in Table 6 and Fig. 4. We observed that slower networks may increase migration times but may reduce the total number of migrations and vice versa. Furthermore, slower networks increases the service latency and the network traffic. Interestingly, the car speeds may significantly affect the total number of triggered migrations but the migrations duration 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 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.

TABLE 6: 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 10 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)
50-60	slow	567.78±13.7	43±2	1689.78±45.78	26934±109.45	2093.45±55.65
	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
80-90	slow	589.09±12.89	38±2	1763.12±43.87	27002±121.4	2102.3±51.09
	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
110-120	slow	555.68±6.49	36±3	1901.65±56.89	27012±19.67	2098.4±43.95
	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

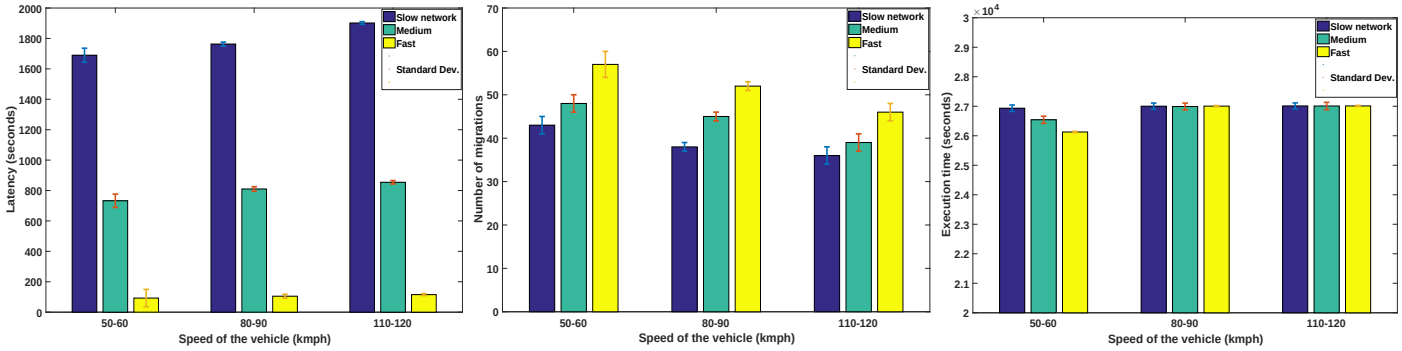

Fig. 4: Results for the proposed "ApMove" methodology in terms of latency, number of migrations, and service execution times for different network types and vehicle speeds [the service execution time is non-significantly affected by the car speed and network type]

We also noted that service execution times are also dependent on the users' mobility, speed, and network traffic. However, as shown in Fig. 4, the service execution time is non-significantly affected by the car speed and type of the network. These little variations are probably due to the latency occurred due to differences in bandwidth of the networks. Our generalization of the outcomes suggest 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 costs 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.

5 Related Work

In [2], the authors have demonstrated a comprehensive review of the literature on cloud computing's application in ITS and connected vehicles, as well as, taxonomies and use 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 (SDN) for better network management in smart and connected automobiles are discussed in [23]. The authors have also demonstrated the importance of embracing cloud, edge, and fog computing for processing huge quantity of realtime data produced by a network of interconnected vehicles, as well as, the challenges that come with this emerging technology. In [24], the authors have proposed a cloud Communication-as-a-Service (CaaS) in order to: (i) enable continuous communication to vehicles beyond the range of roadside units; (ii) ensure Quality of Service (QoS) in terms of throughput, delay, response time, and packet loss rate; and (iii) 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, connected to the network, in such a way that form a tree topology. Similarly, the second layer is dubbed to a roadside cloudlet that is a, local, cloud created between a group of RSUs. Lastly, the central remote cloud is created by a group of servers on the Internet, is the third tier of the V-Cloud architecture.

In [25], the authors have elaborated the well-known virtual machine (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 off-line VM placement challenge. The fog computing architecture is described in [26], along with its various services and applications. Then, with a focus on service and resource availability, the authors haved explored security and privacy challenges in fog computing environments.

TABLE 7: Summary of the related works

Matching criteria	related work [5]	[7]	[18]	[19]	[21]	[24]	[25]	[26]	ApMove
Framework	✓	✓	✓			✓			✓
Migration prediction							✓	✓	✓
Mobility patterns		✓		✓	✓	✓			✓
Service performance	✓		✓		✓	✓		✓	✓
Latency	✓	✓	✓	✓	✓	✓	✓		✓
Service placement	✓	✓	✓	✓	✓		✓	✓	✓
Migrations									
Among edges	✓				✓		✓	✓	✓
Edges to cloud									✓

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 a edge node, a smart, conceptual, pre-copy live migration technique is demonstrated, which predicts the downtime after each and every iteration.

In [5], the authors argue that exploiting 6G mobile networks has the potential to reduce communications delays, in particular, for execution of latency-critical and realtime 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 NIB task migration technique (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: (a) vehicle platooning; (b) lane switching; (c) intersection management; (d) road friction estimation; and (e) energy management. According to [7], while connectivity can significantly improve the connectivity of autonomous vehicle and help the development of existing transportation effectiveness, the level of benefits that can be realised 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 [19] offers the VVMM-U (uniform), VVMM-LW (the least workload), VVMM-MA (mobility aware), and MDWLAM strategies for vehicular virtual machine 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. Authors in [27] give an introduction of the general topic of vehicle behavior prediction and discusses its obstacles. Then, it classifies and reviews the most current solutions based on deep learning approaches using three criteria: (i) input representation; (ii) type of output; and (iii) prediction approach. The research also assesses the efficacy of a variety of well-known remedies, identifies research gaps, and suggests new research possibilities [28], [29].

In [30], the authors 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. In [6], authors 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 [13]. 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 [18]. The summary of the comparison between our proposed technique "CoLocateMe" and other closely related works is given in Table 7. We believe, the comparison would also help readers to quickly identify gaps for further research.

6 Conclusions and Future Work

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 paper, 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.

In the future, we will use more accurate models to characterize the network congestion, performance degradation, latencies, infrastructure heterogineities. 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 resource, application, and network. 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: (i) how often might migration be needed as a consequence of topology? and (ii) what assumptions should be used with regard to cell size, or proximity of edge servers to one or more radio masts?.

ACKNOWLEDGMENTS

This work is supported by the Abdul Wali Khan University, Pakistan and the Discovery project under the Australia Research Council (ARC).

REFERENCES

[1] L. Gillam, K. Katsaros, M. Dianati, and A. Mouzakitis, "Exploring edges for connected and autonomous driving," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 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 Transactions on Intelligent Transportation Systems*, 2021.

[3] L. Gillam, "Will cloud gain an edge, or, closer, to the edge," in *International Conference on Cloud Computing and Services Science*. Springer, 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 Vehicular Technology Magazine*, vol. 12, no. 2, pp. 26–35, 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 of Things Journal*, 2021.

[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 of Things Journal*, vol. 5, no. 2, pp. 829–846, 2018.

[7] U. Montanaro, S. Dixit, S. Fallah, M. Dianati, A. Stevens, D. Oxtoby, and A. Mouzakitis, "Towards connected autonomous driving: review of use-cases," *Vehicle system dynamics*, vol. 57, no. 6, pp. 779–814, 2019.

[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 *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2020, pp. 237–246.

[9] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–14.

[10] M. Zakarya and L. Gillam, "An energy aware cost recovery approach for virtual machine migration," in *International Conference on the Economics of Grids, Clouds, Systems, and Services*. Springer, 2016, pp. 175–190.

[11] M. Zakarya, "An extended energy-aware cost recovery approach for virtual machine migration," *IEEE Systems Journal*, vol. 13, no. 2, pp. 1466–1477, 2018.

[12] B. Shi and H. Shen, "Memory/disk operation aware lightweight vm live migration across data-centers with low performance impact," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 334–342.

[13] J. Li, X. Shen, L. Chen, D. P. Van, J. Ou, L. Wosinska, and J. Chen, "Service migration in fog computing enabled cellular networks to support real-time vehicular communications," *IEEE Access*, vol. 7, pp. 13 704–13 714, 2019.

[14] 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, 2021.

[15] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencourt, "Mobfogsim: Simulation of mobility and migration for fog computing," *Simulation Modelling Practice and Theory*, vol. 101, p. 102062, 2020.

[16] 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," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[17] 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," *arXiv preprint arXiv:2109.05636*, 2021.

[18] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, 2016.

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

[20] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott, "Consolidate iot edge computing with lightweight virtualization," *Ieee network*, vol. 32, no. 1, pp. 102–111, 2018.

[21] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "Myifogsim: A simulator for virtual machine migration in fog computing," in *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, 2017, pp. 47–52.

[22] L. Codeca, R. Frank, and T. Engel, "Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research," in *2015 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2015, pp. 1–8.

[23] R. M. Shukla, S. Sengupta, and M. Chatterjee, "Software-defined network and cloud-edge collaboration for smart and connected vehicles," in *Proceedings of the Workshop Program of the 19th International Conference on Distributed Computing and Networking*, 2018, pp. 1–6.

[24] M. Garai, S. Rekhis, and N. Boudriga, "Communication as a service for cloud vanets," in *2015 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, 2015, pp. 371–377.

[25] H. Yao, C. Bai, D. Zeng, Q. Liang, and Y. Fan, "Migrate or not? exploring virtual machine migration in roadside cloudlet-based vehicular cloud," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 18, pp. 5780–5792, 2015.

[26] 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.

[27] 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 Transactions on Intelligent Transportation Systems*, 2020.

[28] 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 Network*, vol. 34, no. 1, pp. 84–93, 2020.

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

[30] M. Al-Quraan, L. Mohjazi, L. Bariah, A. Centeno, A. Zoha, S. Muhaidat, M. Debbah, and M. A. Imran, "Edge-native intelligence for 6g communications driven by federated learning: A survey of trends and challenges," *arXiv preprint arXiv:2111.07392*, 2021.