Service Migration in Edge Computing Environments for Connected Autonomous Vehicles

Lucas Pacheco[1], Denis Rosário[1], Eduardo Cerqueira[1] and Leandro Villas[2]

[1]Federal University of Pará (UFPA), Belém – Pará – Brazil

[2]University of Campinas (UNICAMP), Campinas – São Paulo – Brazil

lucas.pacheco@itec.ufpa.br, {denis, cerqueira}@ufpa.br

leandro@ic.unicamp.br

Abstract. *In Connected Autonomous Vehicles scenarios or CAV, ubiquitous connectivity will play a major role in the safety of the vehicles and passengers. The extensive amount of sensors in each vehicle will generate huge amounts of data that cannot be processed promptly by onboard units. Edge computing is a crucial solution to provide the required computation power and extremely low latency requirements for the future generation of CAVs. However, the high mobility of vehicles, together with dynamic 5G networking scenarios, poses a challenge to keep the services always close to the users, and therefore, keep the latency very low, such as expected by CAVs. In this paper, we propose MILT, a service migration algorithm for edge computing to perform predictive migration of services based on mobility prediction, available resources, and the quality level of the networks and applications. MILT supports a mobility-based handover prediction scheme to perform a pre-migration to the best available edge server while reducing the latency and increasing the processing capacity of the services of CAVs. Simulation results show the efficiency of the proposed algorithm in terms of latency, migration failures, and network throughput.*

1. Introduction

Connected Autonomous Vehicles, also known as CAVs and autonomous driving technology will revolutionize transportation systems, and bring immeasurable benefits to our society [Lu et al. 2019, Coutinho and Boukerche 2019]. CAVs extend and rely on the notion of connected vehicles, where vehicles will provide and consume new services, such as infotainment, safety, and offloading from other vehicles and remote servers [Coutinho and Boukerche 2019, Aljeri and Boukerche 2019]. However, CAV is still in a preliminary stage, and there are many challenges to be addressed, especially due to the high mobility of vehicles and the dynamic usage of 5G radio resources [Costa et al. 2019]. Therefore, CAV needs new mobility/resource-aware approaches to reduce latency for the applications and improve the usage of network/computing resources [Aljeri and Boukerche 2019].

CAVs are equipped with a wide range of sensors and actuators that collect a great amount of data from the environment, extract context information, and perform driving or service decisions. CAVs must process a large amount of collected with highly tight latency requirements, where some works suggest that up to 2GB of collected data must be processed per vehicle every second [Liu et al. 2019]. This constitutes a great challenge

for the local processing of these data since the vehicle's onboard units (OBU) have limited computation capabilities, as well as, many decisions must be taken based on context and collaborative environmental information (not only based on local data in a vehicle). This calls for the offloading of these data to remote servers. However, the latency requirements of autonomous driving applications cannot be met with traditional computing paradigms, such as Cloud Computing, to guarantee proper safety for the vehicle [Liu et al. 2019].

Edge computing will be a significant part of 5G networks, as the computing for mobile users will happen directly at Cells and Access Points. In this sense, edge computing can be used to enhance reliability, perform the latency-sensitive computations, validate, and offload decision-making in CAVs [Vilalta et al. 2018]. Edge-enabled environments can offer high bandwidth and low latency, which will be essential in autonomous vehicle scenarios. A highly distributed architecture, edge computing can access relevant context information shared between the vehicles (and servers) in their coverage area [Mahmud et al. 2018]. However, the computing power of such approaches is still inferior to traditional cloud computing and must be carefully managed.

Since the edge computing paradigm is very geographically distributed, the services being consumed are susceptible to the high mobility of vehicles [Chen and Tsai 2018]. Traditional mobility management, which mainly consists of vehicles switching cells as the signal strength of serving cells decreases, will no longer be an efficient solution. As vehicles move to different areas in the city, the services being consumed in an edge architecture can be disrupted, or be located many hops away from the user, reducing the Quality of Service (QoS) of CAV applications. In the era of the autonomous vehicle, an efficient mobility management scheme must be considered to guarantee an acceptable QoS for CAVs [Bi et al. 2018].

Service migration is an outstanding solution to keep the services as close as possible to the CAV, assuring the minimal QoS requirements for CAV applications [Meneguette et al. 2019]. Service migration consists of transferring the service running on a virtual machine or container to a new server, where the service migration must be mobility-aware [Bittencourt et al. 2015]. A straightforward strategy to keep services close to vehicles is to perform migration after every handover to the new closest edge server. However, this strategy has a poor performance in highly dynamic CAV scenarios, which is expected in 5G and 6G systems. A migration based on QoS, in which servers with poor QoS performance are avoided, may solve some of these issues, but it still may cause service disruptions in mobile environments [Li et al. 2019].

Migrations can mitigate the problem of keeping services close to the CAVs; however, transferring virtual machines and containers is a resource-heavy operation, with considerable cost to the network. To solve the issues above, one of the strategies employed is the pre-migration of services, where the migration occurs before it becomes necessary, reducing the chance of service disruptions and disconnections. The resources at each edge server should also be taken into account, as migration to servers that have few resources available can compromise the QoS needed for autonomous vehicle applications and impact the entire vehicle's decisions.

To solve the above issues, in this paper, we propose Mobility-based Service Migration in CAV, called MILT. A service migration and resource manager for Vehicular Edge Computing, which tackles the challenge of offloading the computation from CAVs

with the smallest possible latency, and making sure that the vehicle will be served with the necessary resources. MILT considers QoS, resources, and mobility information for decision making, and also MILT uses a Multiple-Criteria Decision-Making algorithm to decide the best possible edge server for each vehicle, and handle the necessary migrations as vehicles move through the scenario. MILT improves the latency in up to 90%, migration failures were non-existent in MILT, and throughput in up to 25% compared to state-of-the-art algorithms.

We organize the rest of the paper as follows. Section 2 outlines the state-of-the-art edge computing resource allocation, mobility management, and service migration. Section 3 describes the MILT algorithm. Section 4 discusses the simulation description and results. Finally, Section 5 introduces the conclusions.

2. Related Work

Allocating resources and performing computing tasks at the edge of the network is a prominent solution to provide low latency and high bandwidth requirements to applications and services in-vehicle environments. However, it brings with itself challenges concerning resources and mobility management. Different approaches have been suggested in recent years to solve these problems. One of the promising ones is the pre-migration of services in order to follow the user's mobility and keep low latency. This section reviews some of the state-of-the-art solutions to these challenges.

Some works study the requirements of vehicular applications in autonomous and connected vehicles, and the limitations of onboard units in performing the necessary computations, such as [Li et al. 2018]. The work proposes a computation allocation framework for offloading of CAVs tasks from onboard units to Vehicular Edge Cloud Computing (VECC). The proposed solution proves itself with greater energy efficiency by allocating the minimum required resource blocks to each vehicle. However, this works does not tackle some of the critical issues VECC carries, including mobility and resource management.

In real-world scenarios, the accuracy of the mobility prediction decreases the more extended the predicted period is. [Yu et al. 2019] proposes an offline pre-migration of services in mobile edge computing. The work uses a mobility prediction scheme to minimize the average latency of the service in the long term. The algorithm, while finding an optimal solution, may have limitations if being used in real-time, as the processing is assumed to be offline.

Other approaches assume mobility management in vehicular networks from a transportation point-of-view. [Liao et al. 2019] proposes a vehicle-as-a-service approach to mobility management and computational tasks migration in edge-enable vehicular networks for path planning. This work takes advantage of the innate mobility of the vehicles to distribute computing capability through the network evenly. This work assumes that computing tasks can be done locally and does not consider the case of service migration. However, the results do not show the impact of the proposed scheme in a real or simulated vehicular scenario.

[Chen et al. 2019] investigates the new paradigm of Cognitive Edge Computing. In this work, cognitive engines in edge servers can learn the computing and network resources available at the edge and solve the problem of communication bandwidth and

delay through the fusion of computing, communication, and storage. The introduction of Cognition to the network operation showed that predicting user behavior can significantly improve the quality level perceived by the end-user. However, it is not best suited for the CAV use case, as it was designed for an application that is not as stringent in requirements.

[Ouyang et al. 2018] tackles the problem of keeping services close to used in edge computing scenarios, where user mobility is unpredictable. The system does not need prior information about user mobility statistics, as it applies a real-time optimization in order to reduce the complexity of the problem. The solution aims to reduce overall migration costs but could be optimized for vehicular scenarios with specialized mobility models. Also, [Gao et al. 2018] proposes a heuristic-based migration algorithm to serve users with varying deadlines, considering user-generated data and the contact patterns between the nodes. Despite employing mobility models in the decision, the proposed solution lacks in terms of QoS and radio resources support for the applications and services.

3. MILT Algorithm

This section details MILT, a QoS-, resource- and mobility-based service migration algorithm for CAV in edge-enabled environments. MILT takes into account CAV's mobility patterns, available resources, delay in the candidate edge servers, and migration costs to detect the need of service migration, and selects the best server to which migrate to, while reducing the latency and keeping high the quality level of CAV applications.

MILT considers a scenario where CAV offload computation to the nearby edge and edge servers as expected in future vehicular systems. MILT's goal is to keep services available for vehicles in CAVs with high reliability and low latency. MILT assumes that service migrations in this scenario are always made in a live manner, *i.e.*, a server instance can only be interrupted after a copy of it has been transferred to the target edge server. MILT assumes a scenario containing CAVs, cellular networks, and edge servers, such as presented by state-of-the-art works [Liu et al. 2019, Yu et al. 2019]:

The scenario is composed of a set of n CAV. Each CAV is assumed to have a radio transceiver to enable the communication between vehicles (V2V) and with an infrastructure (V2I). CAVs have a set of sensors that generate data with a rate of D; this data is later transmitted to the vehicle's cell through radio interfaces. In this sense, the scenario is composed of a set of cells, where each cell has an individual identity. We consider that each cell in the scenario is associated with an Edge server through a reliable point-to-point link. Additionally, we consider a centralized orchestrator in which MILT is executed, such as the one presented in [Hwang et al. 2015]. The orchestrator manages things such as the available resources in each edge server, schedules migrations, triggers the mobility prediction, and other tasks.

Edge servers must perform computation tasks that are either part of a container, or a Virtual Machine (VM) instance. There are advantages and drawbacks to the use of either; this work considers that both may be present in the scenario. Edge servers can connect through a wired channel to perform the migration of Containers and VMs as expected in 5G. Each vehicle sends raw sensor data to an edge server e_n associated with the cell C_n to which the vehicle is connected. The edge servers compute the raw data and return decisions or context information to vehicles. Thus, CAV can make the best decisions for their applications, such as lane changing, collision avoidance, overtaken,

and so on.

Figure 1 shows the main system components in this CAV architecture and their functionalities, and which modules communicate with one another. On the highest level of the system, we have an orchestrator, which is an abstraction for the entity that handles resource management in edge servers, mobility prediction, and migration decisions. Edge servers act only as computation providers that communicate with CAV using the Cells at the network edge.

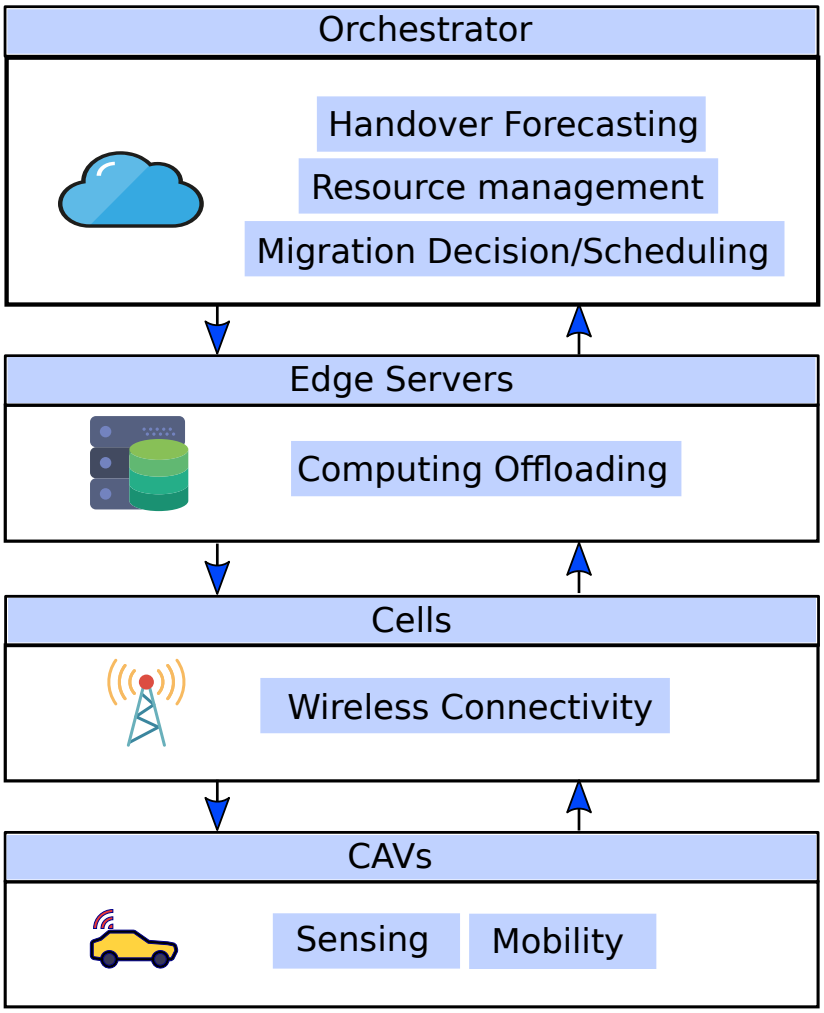


Figure 1. MILT Scenario Components Overview

MILT considers two different events that may trigger a service migration: compromised QoS in a particular edge server, and a handover event. QoS may be compromised by several reasons, such as, if a server is not able to provide the application requirements for whatever reason, its computing tasks must be passed on to a more reliable server. Handover events, on the other hand, will change the path that the data takes between vehicles and servers, and migration may be necessary to keep latency low. Therefore, these events must be predicted and dealt with appropriately.

A poor QoS performance is one of the factors that can trigger a service migration. Thus, MILT requests to the clients of each edge server the latency and RTT values that

they are experiencing, and a threshold based on the application is applied to define if the latency if low enough or not. To guarantee safety applications for CAV, a maximum latency of $10ms$ is necessary [Simsek et al. 2016], thus this is the threshold applied by MILT. The threshold can be adapted to different application requirements.

3.1. Handover Forecasting

In order to execute a seamless pre-migration, our orchestrator is able to forecast and predict imminent changes to network topology and recalculate the appropriate servers for each application under the new topology. In mobile networks, the event that causes topology changes most frequently is the handover process. Cells coverage areas often overlap, and consequently a CAV may have several cells, and edge servers to choose from, as the orchestrator must find the most appropriate one.

A handover generally occurs when a user leaves the coverage area of a cell and enters the one of another cell. This causes a topology change in the network, where the user's previous wireless connection must be broken, and another connection is made to the new network. In current LTE networks, the network must manage session transfer and synchronization between the user and the new network. In an edge scenario, a service migration may also be necessary, as a handover means that the previous edge server may not be the best or closest one.

To provide a reliable handover forecasting, several schemes have been proposed in the literature, such as based on user trajectory [Arshad et al. 2016], or signal strength [Miyim et al. 2013]. We consider a trajectory-based forecasting scheme, for its general reliability and reduced complexity. The forecast consists of predicting in which cell coverage area the vehicle will be short. To this end, MILT performs a position prediction, and a signal estimation, which are enough in the presence of traditional handover algorithms, such as the Strongest Cell one.

MILT uses the mobility prediction to detect when a handover is going to occur. In this context, MILT considers a position prediction to estimate the future positions of vehicles. While any mobility predictor could be integrated to MILT, we chose an AutoRegressive Integrated Moving Average (ARIMA), since it is a robust and low complexity forecaster, as it learns the statistical properties of the time series it is forecasting as the samples arrive in the forecaster.

ARIMA is a statistical model to analyze and forecast time series, which is one of the most general time series forecasting schemes. ARIMA works by taking values of series and making them stationary if necessary. A stationary time series has no trend, and the amplitude of its variations around the mean is constant. An ARIMA model, future value of series, is assumed to be a linear combination of past values and past moving averages.

ARIMA is described as a 3-tuple p, d, q, where p corresponds to several past measurements weighted in the estimation, d consists of the number of differencing series to make statistically stationary, and q corresponds to the number of past moving averages. The basic formulation of the model is given by Eq. (1). We denote past terms as y, past moving averages as ϵ, while θ and ϕ are individual weights for each term and will be trained by the model.

$$y_t = \theta_0 + \phi_1 y_{t-1} + \phi_2 y_{t-1} + \phi_3 y_{t-3} + \cdots + \phi_p y_{t-p} \\ \epsilon_0 + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-1} + \theta_3 \epsilon_{t-3} + \cdots + \theta_q \epsilon_{t-q} \tag{1}$$

The number of past value terms and past moving averages depends on the studied series, where some series are mostly dependant on weighted past values and do not need any moving average terms. The model can be represented by the notation $ARIMA(5, 1, 0)$, which means we use five past terms, perform one differentiation, and consider no past moving averages.

ARIMA is used to forecast a single-variable time series, and thus it has to be done a training step separately for the latitude and longitude measurements. The first step for the general ARIMA formulation is to define the differencing order, *i.e.*, the number of times each term is subtracted from the next of, given by the parameter d, as shown in Eq. (2). ARIMA model can be used for vehicle mobility prediction $L(x_i, y_i, t + 1)$. In this sense, the model must be trained for each vehicle separately for each coordinate (*i.e.*, x, and y).

$$y_t = \begin{cases} Y_t, & \text{if } d = 0 \\ (Y_t - Y_{t-1}), & \text{if } d = 1 \\ (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}), & \text{if } d = 2 \\ \text{And so on} \end{cases} \tag{2}$$

The predicted position is then used as input to a propagation model of choice to estimate the strongest cell at a given moment. In a signal-based handover algorithm, this has relatively high accuracy, as cells tend not to be so close to each other that signal fluctuations have a significant effect — the propagation model used by MILT is the CI, or Close In, propagation loss model [Sun et al. 2016].

The CI propagation loss model is a generic, all-frequencies loss model, that operates at large scales, given by Eq. 3, where PL represents the computed path loss, FSPL represents the loss given by the free space model at $1m$ distance in the given frequency, d is the distance between transmitter and receiver, n is the loss exponent, which depends on the environment being considered, and X_σ is a Gaussian distributed random variable with standard deviation σ. The CI model is widely used in 5G research for its ability to perform reliable path loss prediction in a wide range of frequencies.

$$PL(f, d) = FSPL(f, 1m) + 10nlog_{10}(d) + X_\sigma \tag{3}$$

3.2. Migration Decision

As a general rule, every handover calls for migration in an edge scenario. However, cell coverage areas may overlap, providing multiple options of edge server and cells. It is also essential to check if the target edge server has enough resources for the application. Mobility prediction is used to forecast the position of the vehicle shortly. The longer we extend the prediction, the easier it would be to pre-migrate the services; on the other hand, the less accurate the prediction becomes.

For each server, MILT keeps track of the following parameters at all times: QoS, defined here as the average latency form the server to its connected CAVs; free resources, given in resource units; and the number of hops between the user and the server. QoS and latency are parameters that must be decreased, so in the computation, MILT considers the inverse of the numerical values for them, as it decreases the higher the parameter value is.

MILT finds the best edge server is given the collected metrics, configuring a Multiple-Criteria Decision-Making problem. We chose AHP to balance the input metrics, since it considers a pairwise comparison between the numerical values of each collected parameter and their relative degrees of importance, in order to adjust their weights of each parameter at runtime. The weights of the inputs must be defined when configuring the algorithm. High weight means more importance should be attached to this particular metric, and we define five importance levels, as shown in Table 1.

Table 1. Pairwise Context Importance

$c_{i,j}$	Definition
4	i is much more important than j
2	i is more important than j
1	i is as important as j
$1/2$	i is less important than j
$1/4$	i is much less important than j

MILT constructs for each vehicle a matrix to compare all pairs of metrics. We denote $c_{i,j}$ as how important the $i-th$ element is compared with the $j-th$ element. Also, A = $(C_{i,j})_{nxn}$ represents the comparison matrix, where n denotes the number of elements to be compared, as shown in Eq. (4).

$$A = (C_{i,j})_{nxn} = \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \end{array} \begin{array}{c} \begin{array}{ccc} c_1 & c_2 & c_3 \end{array} \\ \begin{pmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix} \end{array} \tag{4}$$

The metrics collected are placed in the matrix as Eq. 5 shows. Since the priority for MILT is to keep a low latency for CAVs, the QoS parameter is the most important one, two times more important than the Resources, three times more important than the Distance. The rest of the comparisons are made to be coherent with these values, as shown in Eq. 5. We find the eigenvector of the matrix $(C_{i,j})$ by dividing each element by the sum of its column, obtaining the eigenvector $[0.57\ 0.28\ 0.18\ 0.12]$, which means that QoS has weight 0.57, Resources have weight 0.28, 0.14 for the Distance. In this way, MILT performs a product between the eigenvector and a vector that stores the measured values, obtaining the score for each available server.

$$A = (C_{i,j})_{nxn} = \begin{array}{c} \\ QoS \\ R \\ D \end{array} \begin{array}{c} \begin{array}{ccc} QoS & R & D \end{array} \\ \begin{pmatrix} 1 & 2 & 3 \\ 1/2 & 1 & 3/2 \\ 1/3 & 2/3 & 1 \end{pmatrix} \end{array} \rightarrow [0.57\ 0.28\ 0.14] \tag{5}$$

Algorithm 1 presents how the migration process is triggered. The first decision of MILT is whether migration is necessary of not. Migration may be necessary because of vehicle mobility, as it gets more distant from the server, and the latency increases, or for QoS reasons (a server is too loaded, for example).

Algorithm 1: MILT monitor

```
1 while vehicle is connected do
2     Run predictor;
3     if Handover is eminent then
4         MILTMIGRATION;
5     Measure QoS; if QoS is below to the threshold then
6         MILTMIGRATION;
```

Algorithm 2 shows how MILT chooses the edge server to which migrate the user session by the MILTMIGRATION method. The essential characteristic of the decision is whether the target server can deliver the latency and computation requirements and, if so, if the migration can be made promptly. MILT assumes that each edge server can assess the link bandwidth from itself to other edge servers, and uses this bandwidth value to estimate the time it would take to migrate the user session to candidate edge servers. MILT is relatively low complexity, as the AHP calculation is not an expensive operation. The complexity of the algorithm is proportional to the product of the number of CAVs and the number of Edge Servers.

As soon as MILT detects that migration is necessary, the algorithm must evaluate all available servers in the predicted routes (servers that would meet distance and latency requirements) regarding the server's resources and the cost to migrate the service to that specific server.

Algorithm 2: MILTMIGRATION

```
   Data: Number of Hops Acceptable
1  List available servers;
2  for Each available edge server do
3      Get QoS for the server;
4      Estimate the number of hops from user to server;
5      Run AHP and give the server a score;
6  while Server has not been chosen do
7      Get a server with the greatest score;
8      Estimate migration time;
9      if Migration can be done on time then
10         Choose this server as target;
11     else
12         Remove this server from list;
```

4. Evaluation

This section describes the evaluation methodology, including scenario description, simulation parameters, and metrics used to evaluate the performance of different migration algorithm for Edge-enabled CAV scenarios.

4.1. Scenario description and methodology

We implemented the handover algorithms by using NS-3.30[1] simulator, which implements the LTE protocol stack for V2I communication. We consider a vehicular scenario, such as described by [Liu et al. 2019] and [Yu et al. 2019]. In our scenario, we consider a $2Km \times 2Km$ area with composed of 2 macrocells, and several small cells around each macrocell covering the whole scenario, based on the 3GPP LTE release 13 [Access 2013]. This kind of scenario complies with the trend of Ultra-Dense Networks, which makes for an important deployment scheme in next-generation cellular networks [Filo et al. 2020]. Macrocells have a transmission power of 46dBm, while small cells have transmission power of 23dBm The simulation considers the Nakagami path loss model, which can be very suitable for urban scenarios [Rubio et al. 2007]. We conducted 33 simulations with different randomly generated seeds fed to the simulator's pseudo-random number generator (MRG32k3a). Results show the values with a confidence interval of 95%. Services are part of either VMs or containers, which are equally distributed in the network, and only affect the size of the migration here. Note that live migrations are computationally expensive, as services must run in two servers at the same time, however, since decision offloading is a critical application, we assume that the network must be planned to offer the necessary resources.

For the simulation of traffic and vehicle mobility, we employed the BonnMotion[2], which is a topology generation and analysis tool. BonnMotion allows us to reproduce the desired vehicle movements with a predefined path and speeds based on empirical data. We consider a scenario composed of 30, 60, and 90 CAVs moving at different speeds as expected in real cities (ranging between 10-70 km/h).

MILT is tested against the other three service migration strategies. First, a scenario where no migrations happen at all; in this scenario, the edge server to which the CAV is initially connected keeps providing service until the end of the simulation. Then, a greedy strategy was also implemented, where the service is migrated to the new closest edge server after every handover. The last strategy is the one proposed by [Li et al. 2018], already discussed in Section 2, where a QoS-based migration happens.

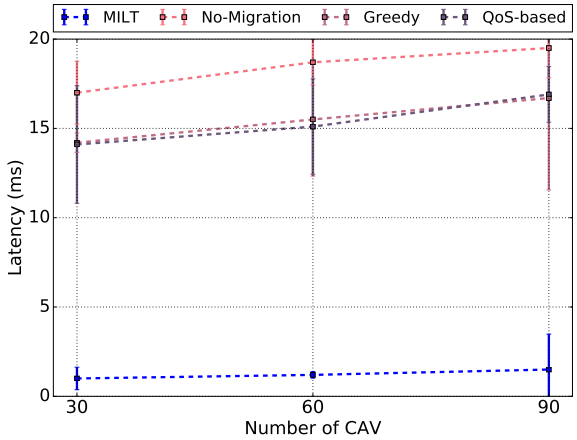
4.2. Results

Figure 2(a) shows the average latency achieved by each of the algorithms tested in a CAV scenario. By analyzing the results, it is possible to see that MILT is the only one of the algorithms that kept latency around 1ms, which is the recommended latency for driving decision making in CAV [Simsek et al. 2016]. This is because of MILT keeps CAVs connected to close servers more often. Other algorithms have higher latency, *i.e.*, up to 90% compared to MILT, especially the No-Migration scenario. This is because these algorithms often keep CAVs connected to sub-optimal edge servers, due to poor decision-making, or migration failures.

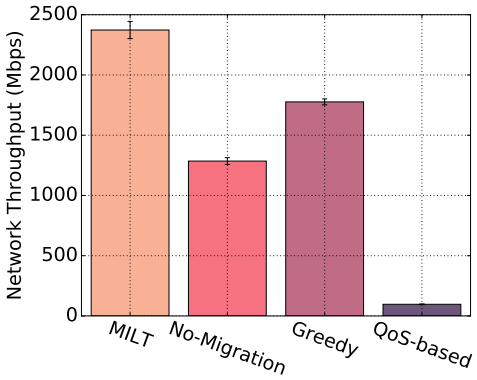
[1]http://www.nsnam.org/

[2]https://sys.cs.uos.de/bonnmotion/

Figure 2(b) shows the average network throughput during the evaluated algorithms. This throughput corresponds to the data flow between the edge servers and CAVs only, *i.e.*context and decision data. We can see that MILT provided 84% more throughput than No-Migration, 33% more than Greedy, and 24% more than QoS-based. No-migration and Greedy algorithms performed reasonably well because they could spread the network load over several edge servers. QoS-based, while minimizing the latency, compromises the network throughput because it overloads the servers with the best QoS, compromising the throughput and the QoS in the process. Only with a multi-criteria decision-making scheme, the best edge server to balance all metrics could be chosen, as MILT does. Finally, MILT improves the network throughput by 25% compared to the second-best performing algorithm.



(a) Average Latency for Different Number of Vehicles in a CAV scenario



(b) Network Throughput for Different Number of Vehicles in a CAV scenario

Figure 2.

Figure 3(a) shows the average amount of failed migrations in the network during the execution of the evaluated algorithms. Migration failures may happen if a CAV leaves the coverage area of its edge server before its session has been transferred to the new one. Another possibility is that the target edge server does not have the available resources to receive the new session, and then must reject the migration. In other words, migration jobs have deadlines that can only be sufficiently met in a predictive manner. We can observe that MILT has no migration failures during the execution, this is because MILT performs predictive migrations to comply with deadlines, and assesses the target resources before migrating the service. The No-Migration scenario also delivers no migration failures since it does not perform the migration. The Greedy algorithm is the one with the highest failure rate, and the Qos-based algorithm exhibits around 110 failures since it is not aware of resources when making migrations.

However, to keep the migration failures at a low, MILT does not perform any sort of cost management, potentially having higher monetary costs. Therefore, we use the AWS TCO Calculator[3] to calculate the financial cost of CPU time per hour. The CPU time cost proportionally decreases when renting a higher number of CPU cores in the same AWS region. We considered the deployment into two regions, corresponding to

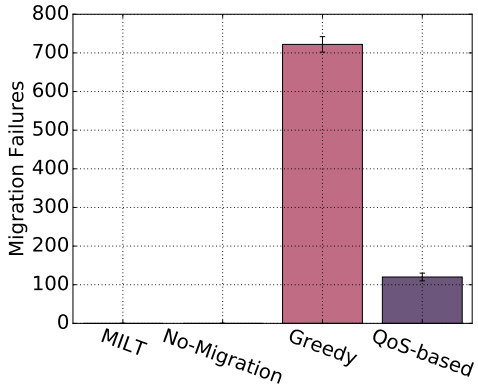
[3]https://awstcocalculator.com/

when the edge services are on the cell, which the CAV is directly connected to, and the case in which the service is more distant than the user. Costs for the tiers used are shown in Table 2.

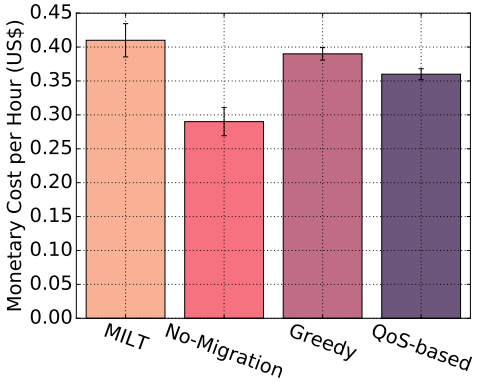
Figure 3(b) shows the average monetary cost for the offloading of each vehicle to the edge under each of the algorithms tested, according to the values of Table 2. We can see that MILT, together with the Greedy approach have the highest monetary costs for their execution, since they, more often than not, keep their users in a tier 1 server. However, only MILT is able to do so while maintaining acceptable QoS values. No-Migration has the lowest cost, since most of its users are in a server far away, and QoS-based maintains an intermediary value because some of its migrations resulted in failures. Therefore, these users keep in a tier 2 server.

Table 2. Monetary Cost of CPU Utilization

Node	CPU Cores	Memory	Storage	Cost/Hour
Tier 2	8	32 GB	2 TB	$0,22896
Tier 1	4	16 GB	1 TB	$0,42408



(a) Migration Failures for Different Algorithms in a CAV scenario



(b) Monetary Cost per Vehicle per Hour Under Each Algorithm

Figure 3.

5. Conclusion

CAVs will revolutionize transportation systems. However, to function correctly, CAVs must process an immense amount of data in order to make decisions based on accurate context information. This computation cannot be feasibly performed on CAV's onboard units, and cloud computation is not an option either. To this extent, Edge computing offers a significant trade-off between computing power and low latency. In the researched literature, the conjunction of a multi-criteria decision-making scheme and service pre-migration has not yet been introduced. We introduce an online computation scheme for service migration, tuned specifically for CAV scenarios. This provides a meaningful contribution compared to the state-of-the-art and can be used as a base for future research.

In this paper, we introduce the MILT , a migration decision algorithm for CAVs in edge-enabled scenarios capable of choosing the best edge servers to provide computation

offloading for CAV. MILT considers CAV mobility, QoS, and edge resources to offer low latency, and high throughput by means of predictively migrating CAVs offload instances to their future Edge servers. Based on simulation results, we show that MILT achieves superior performance in terms of throughput, latency, and migration failures compared to state-of-the-art algorithms.

6. Acknowledgements

"This study was financed in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (CNPQ).

References

Access, E. U. T. R. (2013). Study on small cell enhancements for e-utra and e-utran—higher layer aspects (release 12),".

Aljeri, N. and Boukerche, A. (2019). Fog-enabled vehicular networks: A new challenge for mobility management. *Internet Technology Letters*.

Arshad, R., ElSawy, H., Sorour, S., Al-Naffouri, T. Y., and Alouini, M. (2016). Cooperative handover management in dense cellular networks. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1–6.

Bi, Y., Han, G., Lin, C., Deng, Q., Guo, L., and Li, F. (2018). Mobility support for fog computing: An sdn approach. *IEEE Communications Magazine*, 56(5):53–59.

Bittencourt, L. F., Lopes, M. M., Petri, I., and Rana, O. F. (2015). Towards virtual machine migration in fog computing. In *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 1–8. IEEE.

Chen, M., Li, W., Fortino, G., Hao, Y., Hu, L., and Humar, I. (2019). A dynamic service migration mechanism in edge cognitive computing. *ACM Transactions on Internet Technology (TOIT)*, 19(2):30.

Chen, Y.-S. and Tsai, Y.-T. (2018). A mobility management using follow-me cloud-cloudlet in fog-computing-based rans for smart cities. *Sensors*, 18(2):489.

Costa, J. B., de Souza, A. M., Rosário, D., Cerqueira, E., and Villas, L. A. (2019). Efficient data dissemination protocol based on complex networks' metrics for urban vehicular networks. *Journal of Internet Services and Applications*, 10(1):15.

Coutinho, R. W. and Boukerche, A. (2019). Guidelines for the Design of Vehicular Cloud Infrastructures for Connected Autonomous Vehicles. *IEEE Wireless Communications*, 26(4):6–11.

Filo, M., Foh, C. H., Vahid, S., and Tafazolli, R. (2020). Performance analysis of ultra-dense networks with regularly deployed base stations. *IEEE Transactions on Wireless Communications*.

Gao, Z., Meng, J., Wang, Q., and Yang, Y. (2018). Service migration for deadline-varying user-generated data in mobile edge-clouds. *IEEE World Congress on Services, SERVICES 2018*, pages 53–54.

Hwang, J., Huang, Y.-W., Vukovic, M., and Anerousis, N. (2015). Enterprise-scale cloud migration orchestrator. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 1002–1007. IEEE.

Li, J., Shen, X., Chen, L., Pham Van, D., Ou, J., Wosinska, L., and Chen, J. (2019). Service Migration in Fog Computing Enabled Cellular Networks to Support Real-Time Vehicular Communications. *IEEE Access*, 7:13704–13714.

Li, X., Dang, Y., and Chen, T. (2018). Vehicular Edge Cloud Computing: Depressurize the Intelligent Vehicles Onboard Computational Power. *IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2018-Novem:3421–3426.

Liao, S., Li, J., Wu, J., Yang, W., and Guan, Z. (2019). Fog-enabled vehicle as a service for computing geographical migration in smart cities. *IEEE Access*, 7:8726–8736.

Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y., and Shi, W. (2019). Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, 107(8):1697–1716.

Lu, H., Liu, Q., Tian, D., Li, Y., Kim, H., and Serikawa, S. (2019). The Cognitive Internet of Vehicles for Autonomous Driving. *IEEE Network*, 33(3):65–73.

Mahmud, R., Kotagiri, R., and Buyya, R. (2018). Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer.

Meneguette, R., Rodrigues, D., Costa, J., Rosario, D., and Villas, L. A. (2019). A virtual machine migration policy based on multiple attribute decision in vehicular cloud scenario. In *IEEE International Conference on Communications (ICC)*, pages 1–6.

Miyim, A., Ismail, M., Nordin, R., and Mahardhika, G. (2013). Generic vertical handover prediction algorithm for 4g wireless networks. In *IEEE International Conference on Space Science and Communication (IconSpace)*, pages 307–312. IEEE.

Ouyang, T., Zhou, Z., and Chen, X. (2018). Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing. *IEEE Journal on Selected Areas in Communications*, 36(10):2333–2345.

Rubio, L., Reig, J., and Cardona, N. (2007). Evaluation of nakagami fading behaviour based on measurements in urban scenarios. *AEU-International Journal of Electronics and Communications*, 61(2):135–138.

Simsek, M., Aijaz, A., Dohler, M., Sachs, J., and Fettweis, G. (2016). The 5G-Enabled Tactile Internet: Applications, requirements, and architecture. *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 61–66.

Sun, S., Rappaport, T. S., Rangan, S., Thomas, T. A., Ghosh, A., Kovacs, I. Z., Rodriguez, I., Koymen, O., Partyka, A., and Jarvelainen, J. (2016). Propagation path loss models for 5g urban micro-and macro-cellular scenarios. In *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pages 1–6. IEEE.

Vilalta, R., Vía, S., Mira, F., Casellas, R., Muñoz, R., Alonso-Zarate, J., Kousaridas, A., and Dillinger, M. (2018). Control and management of a connected car using sdn/nfv, fog computing and yang data models. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 378–383. IEEE.

Yu, X., Guan, M., Liao, M., and Fan, X. (2019). Pre-Migration of Vehicle to Network Services Based on Priority in Mobile Edge Computing. *IEEE Access*, 7:3722–3730.