applied sciences

MDPI

*Article*

# Predictive Migration Performance in Vehicular Edge Computing Environments

**Katja Gilly [1,*,†], Sonja Filiposka [2,†] and Salvador Alcaraz [1,†]**

[1] Department of Computer Engineering, Miguel Hernandez University of Elche, 03202 Elche, Spain; salcaraz@umh.es
[2] Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Rugjer Boshkovikj 16, 1000 Skopje, North Macedonia; sonja.filiposka@finki.ukim.mk
* Correspondence: katya@umh.es; Tel.: +34-966-658-565
† These authors contributed equally to this work.

**Featured Application: Employing proactive multi-access edge computing (MEC) service migration techniques helps provide offloaded computing power to highly dynamic vehicular networks while maintaining continuously low latency. This enables vehicles to take advantage of extra processing and storage, which can be used for employing deep learning algorithms for autonomous driving.**

**Abstract:** Advanced learning algorithms for autonomous driving require lots of processing and storage power, which puts a strain on vehicles' computing resources. Using a combination of 5G network connectivity with ultra-high bandwidth and low latency together with extra computing power located at the edge of the network can help extend the capabilities of vehicular networks. However, due to the high mobility, it is essential that the offloaded services are migrated so that they are always in close proximity to the requester. Using proactive migration techniques ensures minimum latency for high service quality. However, predicting the next edge server to migrate comes with an error that can have deteriorating effects on the latency. In this paper, we examine the influence of mobility prediction errors on edge service migration performances in terms of latency penalty using a large-scale urban vehicular simulation. Our results show that the average service delay increases almost linearly with the migration prediction error, with 20% error yielding almost double service latency.

**Keywords:** edge computing; migrations; predictive modelling; urban vehicular scenarios



**Citation:** Gilly, K.; Filiposka, S.; Alcaraz, S. Predictive Migration Performance in Vehicular Edge Computing Environments. *Appl. Sci.* **2021**, *11*, 944. https://doi.org/10.3390/app11030944

Academic Editors: Javier Alonso Ruiz, Jeroen Ploeg, Martin Lauer, Angel Llamazaers and Noelia Hernández Parra
Received: 29 October 2020
Accepted: 15 January 2021
Published: 21 January 2021



## 1. Introduction

Advances in vehicular networking in recent years have tremendously changed the requirements of next-generation networks. The communication patterns start from the ad hoc creation of dynamic vehicular networks (VANETs) in an infrastructure-less environment, but also include infrastructure-oriented communication, such as information exchange between vehicles and smart road infrastructure, or, lately, the emerging 5G ecosystem, which has created creates an opportunity for the development of so-called 5G-enabled vehicular communications and networking (VCN) [1]. In essence, the introduction of 5G offers the possibility to reach ultra-reliable, extremely low-latency, high-speed communication among a plethora of devices, with vehicles being one of the main use-case scenarios [2]. Since its introduction, 5G has involved not only changes in radio access technology, but drastic changes in the overall architecture of service delivery. In the new landscape, 5G Vehicle-to-Enerything (V2X) communications can be considered as an integral part of the advancements in autonomous driving. The autonomous driving use cases and requirements, defined as V2X phase 3, are being developed in the latest 5G Standard Release

16 defined by the 3GPP [3]. In addition to these standardisation efforts, the 5G Automotive Association has developed a roadmap for advanced driving use cases, connectivity technologies, and radio spectra related to the use of 5G for V2X communications [4].

In this ubiquitously connected environment supported by the 5G ultra-communication features, the number of possibilities and services that can be developed around the premise of autonomous driving is staggering. However, these go hand in hand with the demand for increased processing power that goes beyond the capabilities of the vehicle processing unit. Autonomous vehicles are expected to be able to deal with big data on the fly while being highly mobile. To overcome these challenges, the services and applications running on the vehicles need to rely on a mix of a local on-board processing and remote additional processing power [5]. To be able to provide these remote processing features, the 5G architecture is extended with support for multi-access edge computing (MEC). Using the possibilities for remote heavy weight processing, autonomous driving can be augmented with services such as automated traffic management, where a real-time analysis of the current traffic status in a whole area can be conducted to help the vehicle efficiently navigate to its destination. In parts of the literature, the specific use cases that focus on the integration of vehicular networks and MEC are known as vehicular edge computing (VEC) [6].

The MEC concept describes an architecture where the all-purpose processing and storage power of traditional cloud computing datacentres is relocated on the edge of the network. Thus, the 5G micro-base stations come equipped with a number of general-purpose virtualised servers to provide additional processing power requested by the users, as can be observed in Figure 1. In this way, even when the processing is partially or fully dislocated from the vehicle, the delay in obtaining the results is negligible due to the combination of the extreme low latency of the 5G network and the co-location of the processing power right next to the base station. By taking advantage of this opportunity, vehicular services practically have no limitations due to lack of resources.

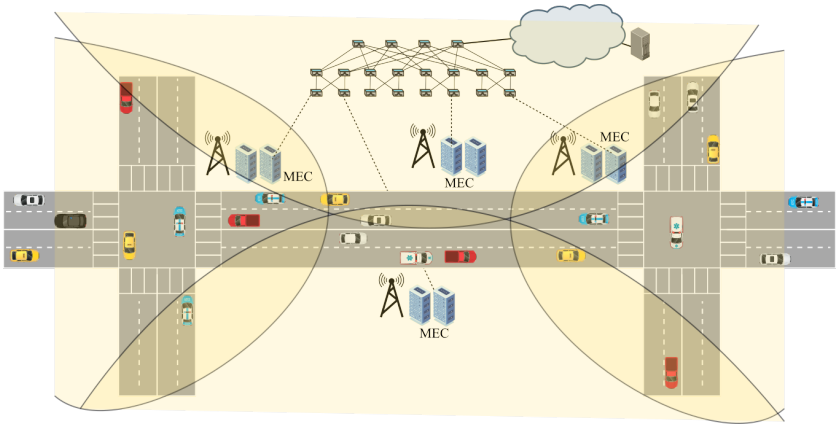



**Figure 1.** An example of a vehicular multi-access edge computing (MEC) architecture.

However, to be able to efficiently implement this scenario for 5G user equipment that is highly dynamic and mobile, as is the case with autonomous vehicles, there are some considerations that need to be taken into account [7]. The main problem in exploiting the MEC processing capabilities in vehicular environments is the frequent handover (or change of base stations) due to mobility. In other words, once the vehicle exits the coverage area of the base station with the co-located MEC servers that host those vehicle services, the premise of ultra-low latency becomes an issue. To solve this problem, efficient resource management of the MEC resources is required. This includes solving problems such as the placement of the services on the MEC resources that are closest to the vehicle and the continuous migration of the services as the vehicle moves from one base station's coverage area to another under the constraints of the finite MEC resources available at each location. The aim of this paper is to discuss the possibilities of using prediction in order to anticipate

these handovers and consequent MEC service migrations in order to analyse how various degrees of prediction errors can affect MEC service delays.

The rest of this paper is organised as follows: Section 2 depicts the related work. In Section 3, the MEC architecture is introduced, focusing on the problem of resource management in MEC. A hierarchical approach to resource management proposed by the authors is briefly described, focusing on how it can be combined with prediction modelling to incite proactive behaviour of the MEC system. Section 4 describes the details of the simulation process. In Section 5, the performances of predictive migrations are analysed based on a large-scale urban vehicular simulation scenario, followed by the obtained results and a discussion. Section 6 concludes the paper.

## 2. Related Work

A number of studies [8–10] have researched the problem of offloading tasks to the edge, focusing on various aspects of the problem, such as when to offload and when not to offload or partial or full offloading; a number of algorithms [11–13] have been proposed to deal with the decision-making problem of where to place the service and when and where to migrate a service in order to maintain the agreed-upon service level. Lately, these MEC resource management approaches have turned towards proactive methods that use predictive techniques [14,15] to preemptively reserve the resources in the MEC infrastructure and, thus, guarantee continuous service delivery with minimum delay. The proactive behaviour is based on learning the end device's patterns and predicting its future requirements. This preemptive resource management in terms of service placement and migration involves learning the mobility patterns and responding to the changes imposed by the predicted new locations.

The MEC reference architecture, as specified by ETSI [16], defines three main components that enable the management and hosting of user services, i.e., MEC applications. Starting from the hosts that comprise the MEC computing infrastructure, each mobile edge host is managed via the mobile edge platform that is in charge of the virtual resources offered by the edge host. A group, or cluster, of edge hosts, i.e., local MEC datacentre, is managed by the mobile edge platform manager, which decides how to best use the resources in the cluster in order to provide the highest efficiency in terms of service performances and energy consumption. The heart of the system is implemented in the mobile edge orchestrator, which manages the complete MEC service lifecycle. It receives all end-user equipment requests for service provisioning and management and decides which platform manager will be in charge of fulfilling a given request. Hence, the orchestrator has a top-level view of all MEC resources and their current states combined with the information about the past and current positioning of the end-user equipment based on the base stations used for communication.

The main goal of the MEC orchestrator is to enable highly effective resource management of the MEC hosts so that the MEC services provided to the user adhere to the requirements in terms of processing power, storage, bandwidth, and latency. The resource management activities can be divided into two categories: placement of newly requested services where the orchestrator needs to decide on the optimal location, i.e., MEC host, where a new MEC application will be instantiated using available virtual resources, and migration of existing services to a different MEC host so that the service performance requirements are kept optimal during the service lifetime.

In the recent literature, there have been many different proposals on how to tackle the complexities of the resource management problem, which has been shown to be NP-hard. Focusing on scenarios related to using the MEC infrastructure to host vehicular services, existing methods include augmenting the MEC infrastructure with next-generation networking technologies, such as software-defined networks (SDNs) and network function virtualisation (NFV), so that network agility can be achieved to match the dynamics of the vehicular network. An example is presented in [8], where the authors introduce SDN-enabled resource management to support the heterogeneous quality of service re-

quirements. They investigate resource management by balancing resource usage under the constraints of computing, storage, and bandwidth. The problem is formulated as an optimisation framework that focuses on maximising network utility, which is represented as a logarithmic function. To tackle the problem of not having enough resources in the MEC infrastructure to host all requested services, the authors in [9] propose a collaborative computation offloading and resource allocation optimisation scheme that utilises the cloud when the MEC servers do not meet the resource demand. For a comprehensive survey of resource management techniques in fog/edge computing, please refer to [10], where different approaches in terms of architecture, infrastructure, and algorithms are presented.

The focus of this paper is on proactive mechanisms for resource management that are based on prediction of service requests and preemptive implementation of service actions so that the latency experienced is as low as possible. A number of different predictive approaches aiming to increase the resource management performance have been proposed. The authors in [11] provide a survey of artificial intelligence (AI) techniques used for computation offloading in edge computing. A special section is dedicated to the problem of offloading when the users represent a vehicular network, as well as problems such as a highly dynamic environment, intermittent connectivity, and the requirement of minimum latency. However, the techniques discussed mostly focus on the scheduling aspects—deciding when and where to do offloading and how much should be offloaded. For example, in [12], the authors propose a deep Q-network technique for task migration in order to help autonomous driving applications. Reinforcement learning is used to make decisions on when to migrate the task based on past experience, aiming to maximise the total sum of reward. A different approach is considered in [13], which proposes an intelligent task prediction and computation offloading, where a deep learning algorithm based on long short-term memory (LSTM) is used for prediction of computing tasks. The decision on whether to offload is based on minimising the delay, which is represented as a function of the predicted task size, computing resources on the user side, and computing resources in the edge. A similar objective function is used for task migration in the case when there is a problem with the execution of the task in the edge. However, no matter the technique used for learning and predicting, there is always a degree of error in the prediction, which might introduce adverse effects in the service levels.

In this work, we aim to investigate the performance when using proactive algorithms for service migration in which AI techniques are used to predict user mobility and, hence, to foresee the need for service migration. For a visualisation of the problem, please refer to Figure 3. The main idea of the proactive service migration is to respond to the predicted handover event when the vehicle switches from one base station to another, and to preemptively start a live migration process so that the change of base station does not impact the latency when the vehicle accesses the hosted MEC service. Of course, the risk of mobility prediction is that it is not guaranteed that it will be 100% correct at all times, and there might be cases when latency will be introduced into the system because of an incorrect prediction, in which case the service migration is done as a reactive instead of proactive process.

A similar consideration is analysed in [14], where a Glimpse mobility prediction model is proposed to predict the next user locations using deep reinforcement learning. The results presented for a small-scale scenario show that the minimum latency is observed in the situations where the chosen target MEC server is located close to the user. On the other hand, in [15], the authors argue that using simpler prediction mechanisms that focus on the prediction of handover events instead of the prediction of the detailed user mobility can outperform the more complex AI-based techniques, which require training and a lot of preprocessing. They compare the probabilistic approach to more enhanced approaches, including k-nearest neighbours and decision trees, in the cases when prediction is based on global knowledge and per-user-only knowledge. Their results show that the handover events can be predicted with an accuracy between 57% and 90% depending on the employed technique. These results raise the question of how this accuracy affects the

performances of the services when handover prediction mechanisms are employed to do proactive migration. Moreover, another important factor that needs to be considered is that the performances of most of the discussed schemes and analyses are presented using a simplified MEC scenario consisting of less than 30 MEC servers in a simple setting, which is usually presented as a straight 1 km two-way street with base stations positioned at regular intervals.

The main goal of this paper is to analyse how various degrees of prediction inaccuracy affect the MEC service delay by investigating proactive service migration performance in a large-scale scenario using non-uniformly distributed base stations scattered in a realistic urban vehicular traffic setting. Firstly, in the next section, we introduce our proactive approach to MEC resource management. Secondly, we present the use case implemented in Section 5 to analyse a large-scale deployment of a vehicular network in a realistic urban scenario connected via 5G and supported with a number of distributed MEC servers. The use case for the analysis is a large-scale deployment of a vehicular network in a realistic urban scenario connected via 5G and supported with a number of distributed MEC servers. The simulation parameters are chosen to be aligned with the identified use cases and examples for autonomous driving with specific service-level requirements provided by the 5G Automotive Association (5GAA) [17]. Our simulation results show that 20% of the error in the mobility prediction can introduce an additional delay of over 80% to services, which needs to be carefully considered and taken into account for ultra-low-latency, highly reliable services, such as autonomous driving.

## 3. Proactive Approach to MEC Resource Management

The analysis of the proactive service migration is done by enhancing our hierarchical community-based approach to service allocation and migration presented in [18] with a mobility prediction module that targets handover prediction. As depicted in the diagram in Figure 2, the approach taken to resource management is hierarchical and closely follows the geographical dispersion of the network provider base stations and associated MEC servers in the local MEC datacentres. The hierarchical approach is also aligned with the multi-layer distribution of the MEC resources, as described in the ETSI MEC specification [16]. The main idea is that, in order to solve the problem of where to place a newly requested MEC service, the decision making is divided into two stages. In the initial stage, the best possible group, or a so-called community, of MEC servers is identified such that: (1) There are enough resources to fulfil the service request and (2) the latency between the identified community and the vehicle requesting the service is at its minimum. The result of the first stage of decision making is a list of potential communities where the service can be placed in an increasing order based on the latency that would be experienced by the end-user equipment. This list is then used in the second stage of decision making, where the best possible MEC server is chosen from the identified community of MEC servers. While the high-level goal of the first stage is to ensure minimum latency, the lower-level goal of the second stage is the smart use of the available MEC servers' resources for load balancing, server consolidation, minimum energy consumption, or a combination of multiple goals.

In a formal representation of this hierarchical community-based resource management approach, each MEC host is denoted as $H_i$, where $i = 1, \ldots, H$ and $H$ is the total number of MEC servers in the 5G edge network. The available resources of each server $\vec{H}_i$ at the moment $t$ are described with the vector $\{C_i(t), M_i(t), S_i(t), B_i(t)\}$, where $C_i(t)$ is the available processing power at the moment $t$, bounded with $C_{max}$. Similarly, $M_i(t)$ is the current available RAM memory bounded with $M_{max}$, $S_i(t)$ is the current available storage capacity bounded with $S_{max}$, and $B_i(t)$ is the available network bandwidth bounded with $B_{max}$. Using a community detection algorithm over the graph that describes the service provider links between all MEC servers in the network, such as the Girvan–Newman algorithm [19], the natural hierarchical structure of communities of MEC servers in the network, or the network dendrogram, can be discovered, denoted with $Com_i$, and $i = 0, \ldots, D$, where $Com_0$ is the highest-level community that contains all MEC servers in

the network, and $Com_{D-H-1}$ to $Com_D$ are the lowest-level communities that contain each individual single MEC. Depending on the size and complexity of the network provider, there can be one or multiple levels in the community dendrogram in between these two extreme cases. Similarly, the MEC services requested are represented as service requirement vectors $\vec{S_i}(t_s, t_e) = \{SC_i, SM_i, SS_i\}$, which describe the amount of resources needed by the service $S_i$ that is requested at $t_s$ and is no longer needed at $t_e$.

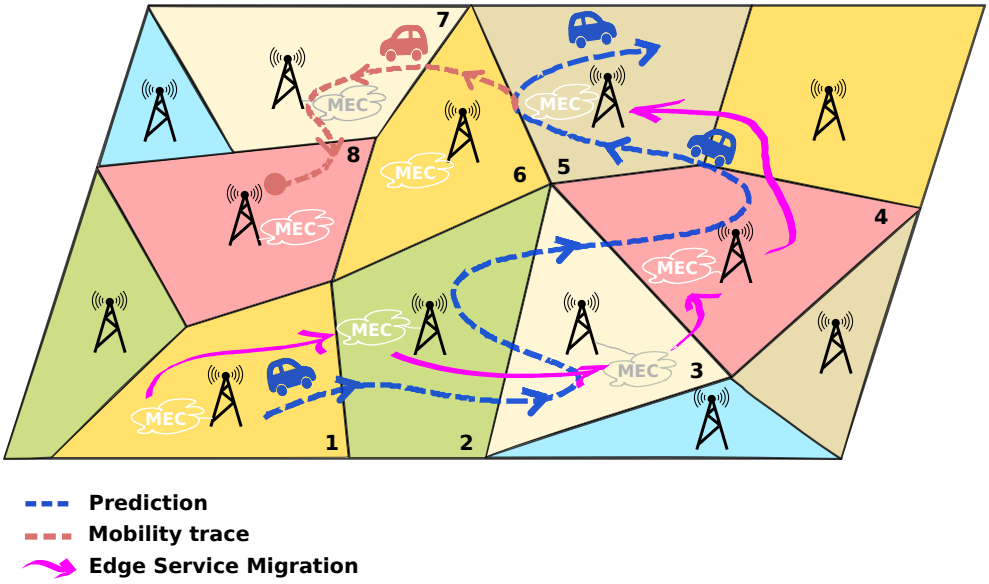



**Figure 2.** Example service migration due to change in coverage area.

The first stage of the hierarchical community-based placement algorithm aims to optimise an objective function that can be described as finding $j$ such that the latency between the vehicle requesting the service $S_i$ at $t_s$ and the MEC host community $Com_j$ is minimised while the chosen community has enough resources to host the service for each dimension of the resource vectors $\sum_{k \in Com_j}(\vec{H_k}(t)) \leq \vec{S_i}(t_s, t_e)$. In the second stage, the objective function is used to find $\vec{H_h} \in Com_j$ such that the resource usage of the MEC hosts in the community $Com_j$ is optimal and $\vec{H_h}(t) \leq \vec{S_i}(t_s, t_e)$. If there is no such $h$ available, then there are not enough resources in the MEC, and the service is forwarded to be hosted in the cloud, which serves as a backup solution.

Once the initial placement is done, the proactive status analysis starts based on the input from the mobility prediction module, which predicts the base station handover events for all current MEC users. The prediction information is used so that the user service is migrated to an optimal new location before the handover event actually occurs. In other words, the mobility prediction module should be able to predict a handover with a time of at least $X$ before it actually happens, where $X$ is the time needed to migrate the service, which depends on the bandwidth and service size, through the MEC network infrastructure. In this way, the user does not experience any additional service delay due to migration. Not all handover events require service migration, depending on the definition of the communities relative to the locations of base stations. Only if the change of base station entails a change in the MEC community is the attempt to migrate the service to a new MEC community made. The migration is only activated if there are enough resources to migrate the service to the new optimal MEC community. If not, the service remains at the old location. The new optimal MEC community is identified in the same way as described for the initial placement algorithm. Due to the fact that the migration process is based on live migration, during migration, the service will use double resources: at the old MEC host where it resided, and at the new MEC host where it is decided to be moved. If there are no available resources in the new optimal MEC community, then the migration cannot be performed preemptively, and the system regularly checks if the migration can

be performed at some point later in time until it succeeds in migrating; otherwise, a new handover event is predicted and a new optimal MEC community destination is chosen, or the service end time is reached and the service is no longer needed.

It is straightforward to conclude that a correct handover prediction done sufficiently in advance will yield the best results, enabling the best chances to migrate the service in order to maintain the minimum latency perceived by the end-user equipment. Suboptimal situations might occur due to two reasons: (1) As discussed previously, the handover prediction may be correct and available in time, but the lack of resources in the new community imposes a higher latency that is experienced at least temporarily until the situation changes; (2) the handover prediction is not correct, and thus, the service is not migrated or is migrated (see Figure 3). A more detailed implementation of the proactive migration check algorithm is provided in pseudocode in Listing 1, and the service migration function is given in Listing 2. The details of the implementation of the initial placement algorithm are out of the scope of this paper, and its pseudocode can be found in [18].

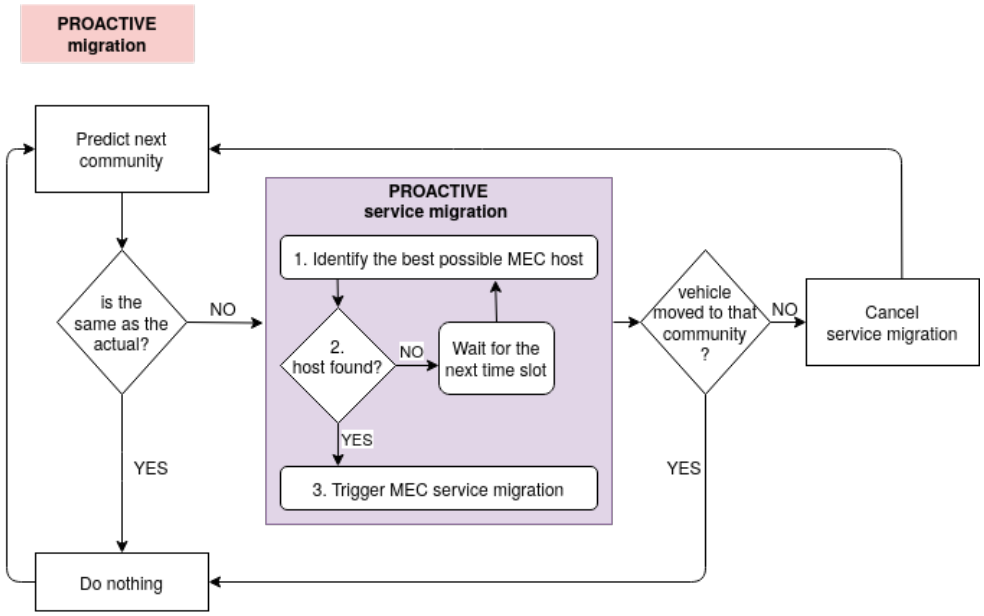



**Figure 3.** Proactive hierarchical community approach to service migration.

**Listing 1.** Proactive migration check pseudocode.

```
void ProactiveMigration (Service[] S, Community[] Com, BaseStation[] BS) {
    int handover = -1; // no handover predicted
    for (int i = 0; i < S.length(); i++) {
        // call mobility prediction algorithm
        Location loc = predictUserLocation(S[i]);
        // check for handover event
        BaseStation bs = getBaseStationCoveringLocation(loc);
        Community c = getCommunity(bs, BS, Com);
        if (c != getCommunity(S[i])) {
            int c = ServiceMigration(S[i], bs, Com);
            if (c != -1)
                // verify correct prediction in X seconds
                Scheduler.add(MigrationCheck(S[i]), X);
        }
    }
    // schedule new proactive check every second
    Scheduler.add(ProactiveMigration(S, Com, BS), 1);
}
```

**Listing 2.** Service migration pseudocode.

```
int ServiceMigration (Service s, BaseStation newBS, Community[] com) {
    int newSlocation = -1; // no location found, migration is canceled
    if (isInCloud(s)) // try to migrate in edge infrastructure
        // equals to allocation attempt
       newSlocation = initialPlacement(s, newBS, com)
    else {   // handover predicted, try to move to the new community
       Communities[] potentialCom =
                        getSmallestCommunityWithBS(com, newBS);
       if (resourcesAreAvailable(s, potentialCom))
                // activates lower layer placement algorithm
          newSlocation = chooseEdgeServer(s, potentialCom);
    }
    return newSlocation;
}
```

The proposed approach to service migration supports the use of different modules/techniques that provide the function of predictive handovers, i.e., the function call predictUserLocation() in Listing 1. This means that the functional block that forecasts the next community, as given in Figure 3, can be implemented using regressive approaches, such as ARIMA, neural network approaches, deep learning techniques, etc. Incorrect handover prediction is the result of the imperfections of the probabilistic or AI-based learning algorithms that supply the information about its prediction. As already mentioned, the typical mobility prediction algorithms today [15] have different rates of accuracy in their prediction depending on the size of the time window, size of the available training dataset, and level of knowledge.

Formally, accuracy in this context can be defined as the fraction of predictions that are correct, or accuracy = number of correctly predicted handovers/total number of handovers. Both false positives and false negatives have a deteriorating impact on the MEC service migration performance. In the case of a false positive, the handover event is falsely predicted and migration is triggered, thus taking up unnecessary resources in the MEC infrastructure. However, from the point of view of latency, this type of error does not have an effect because the service continues to be delivered with the minimum latency to the end-user equipment, and once the false positive is detected, the migration that was started can be cancelled and the resources freed. On the other hand, false negatives include handover events that the prediction module does not report on, in which case the migration occurs as a reaction to the handover, involving a time period when there is suboptimal service delivery with increased latency perceived by the end-user equipment. The main goal of this paper is to analyse the service migration performance in terms of latency experienced in the presence of different prediction error rates compared to a perfect prediction.

## 4. Simulation Description and Details

For the purposes of creating a realistic—yet dense—simulation scenario with autonomous driving as an example MEC service, we studied service-level requirements that were defined for the examples and use cases described by the 5GAA in its white paper [17]. Autonomous driving is considered as a separate use-case group by the 5GAA, and is related to the self-driving levels 4 and 5, as seen by the Society of Automotive Engineers. The described use-case examples include assistance for intersection movement, emergency break warning, hazardous location warning, speed harmonisation, sensor sharing, etc. There are different user stories that are related to these use cases, such as exchange of projected trajectories between vehicles, software updates, tele-operated driving, automated parking, map collection and sharing, etc. The typical service-level requirements for these examples include: vehicle range of around 350 m, max velocity of 50 km/h in urban areas, vehicle density of 1500 vehicles/km$^2$, and max latency of around 120 ms. The simulation parameters' tables presented below show that the chosen values have been tuned to cor-

respond to the parameters defined by the 5GAA. Although the latency presented in the results is well below 120 ms, please note that the service latency analysed in this part is the MEC-specific latency, which is one segment of the overall latency experienced by the endpoint. Thus, to be able to work under the restriction of 120 ms, it is imperative that the MEC networking latency that is added to the 5G wireless communication latency is as low as possible. This is of critical importance for special emergency use-cases example, where the service level requirement for latency can drop down to a maximum of 10 ms for the overall communication.

A large-scale urban simulation environment was created using the CloudSim simulator for the MEC infrastructure and SUMO for the simulation of the vehicular network. The urban simulation scenario was defined over a geographical coverage area of $1.8 \times 2$ km$^2$, which corresponds to the city centre of Alicante, Spain. The real OpenStreetMaps information, which includes traffic lights, roundabouts, pedestrian crossings, speed limits, street priority, maximum speed, and all the necessary information of traffic regulations, was used to describe the scenario in SUMO. Different traffic densities were considered when configuring the SUMO simulation in order to study the performance of our proactive MEC resource management approach with different congestion levels, which range from light-density traffic to dense traffic without jams. In Table 1, we included the numerical input data that were configured in the SUMO simulations.

**Table 1.** SUMO simulation parameters.

| Parameter | Value |
|---|---|
| OpenStreetMap area | $1.8 \times 2$ km$^2$ |
| Simulation time | 10,000 s |
| Min path length | 500 m |
| max velocity | 50 km/h |
| Vehicle density ranges | (1) [4900, 5000] vehicles $\sim$ 1375 vehicles/km$^2$<br>(2) [5700, 5800] vehicles $\sim$ 1600 vehicles/km$^2$<br>(3) [6900, 7000] vehicles $\sim$ 1930 vehicles/km$^2$<br>(4) [8600, 8700] vehicles $\sim$ 2400 vehicles/km$^2$ |
| MEC services requested per vehicle | 1 |
| Vehicle type | passenger car |
| Pause time | 0 |
| Range | 300 m |
| Number of base stations | 9 |
| Number of hosts per base station | 5, 7, 9 11, or 13 |

Vehicles randomly show up in the simulation area during the simulation time of 10,000 s, and each car must go through at least 500 m before it randomly leaves the simulation area. An MEC service is requested at the moment the vehicle appears and is destroyed when it exits the simulation. During the simulation, the density of the vehicles in the urban area remains constant. All simulations were repeated with different seeds to increase the quality of the sample's representation. The output of SUMO was post-processed by a transformation engine as an intermediate step (please see Figure 4) in order to add information about the 5G cells defined by the Voronoi tessellation of the coverage area and, hence, to associate the active base station that covers the area wherein each vehicle is located. This information entered CloudSim, which showed a high-level design of the integrated simulation environment with a description of the interface connections between both simulators and the transformation engine. This process is described in detail in [20].

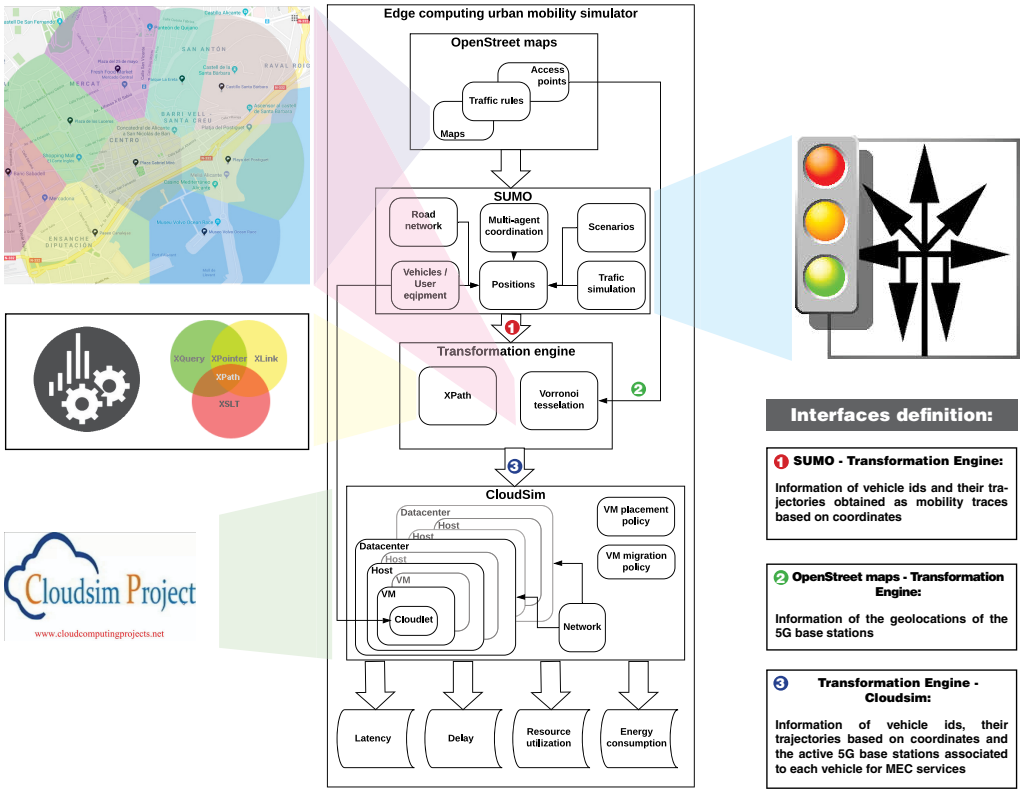



**Figure 4.** Integration process of the urban simulation environment.

Regarding the underlying network topology that was configured, our scenario set up consisted of a 5G telecommunications network with nine base stations connected through a 1 Gb network supported by a hierarchical fat tree topology with three layers, which were: edge, distribution, and core. Depending on where the service was allocated in the MEC system, the MEC delay between the base station and the service provided would include more or fewer hops through the network topology, as the service data might have to go through the distribution layer or even through the core layer. We used the standard delays for edge, distribution, and core layer links that were provided by CloudSim. Each base station had a homogeneous MEC datacentre connected with some MEC hosts. The number of hosts per datacentre was selected to provide the simulation scenario with a resource pool that, on one hand, covers the service demands of the least dense scenario and, on the other hand, provides enough capacity to deal with the most densely configured traffic workload without exceeding the road capacity. Therefore, we configured each MEC datacentre with 5, 7, 9, 11, or 13 MEC hosts, making a global number of 45, 63, 81, 99, or 113 hosts, respectively. These values were chosen to represent all possible scenarios with enough capacity to deal with the urban traffic workload considered. The simulations parameters defined in CloudSim and more information related to the numerical input of the simulation scenarios are detailed in Table 2.

MEC applications and services are normally deployed on edge datacentres as virtualised entities; we then defined them as virtual machines (VMs) in CloudSim that are managed by the MEC orchestrator. The resource management process runs, firstly, a VM placement step that defines where to allocate each VM in the associated MEC datacentre, and then it goes through several VM migration steps in which the allocation can be optimised to reduce the MEC delay as the vehicle moves through the coverage area.

**Table 2.** CloudSim simulation parameters.

| Parameter | Value |
|---|---|
| Virtual resource usage | space sharing |
| MEC network | 3 layer 1 Gb fat tree |
| MEC network switches | 9 edge, 3 access, 2 core |
| MEC network latencies | 1.57 ms, 2.45 ms, 2.85 ms |
| Number of server hosts | 45, 63, 81, 99, and 113 hosts |
| Server host type: | 2–6 cores, 12 GB RAM, 100 Mbps |
| Initial placement policy | hierarchical communities |
| Migration policy | follow-me |
| VM characteristics | 2 cores, 2 GB RAM, 100 Mbps |
| Cloudlet complexity | 100% of CPU time used |

## 5. Results and Discussion

As the large amount of data obtained by the simulations can be depicted in different ways for better understanding, we analysed the performance of our predictive approach from several points of view: edge service allocation performance, edge service migration performance, edge service delay analysis, and the study of the relation between delay and prediction accuracy. We chose to show, on one hand, the number of edge-service-first allocations, and then migrations performed while vehicles moved from one coverage area to another one during the simulation time. On the other hand, we selected two different metrics to show in two sets of images, i.e., (a) average delay versus number of vehicles, which gives us a general overview of delay results in the complete set of simulations, and (b) delay variation versus prediction error, which marks how the accuracy in the prediction affects the latency in edge services.

Let us divide the result analysis and discussion into three sections that consider different metrics.

### *5.1. Edge Service Allocation Performance*

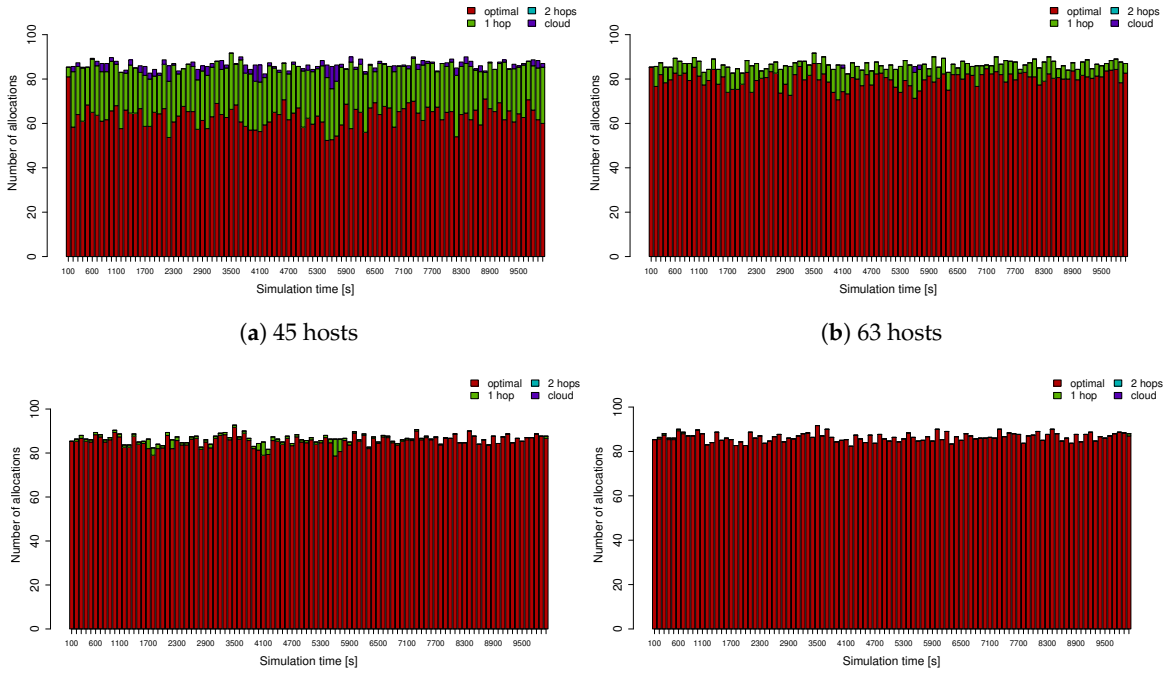
The placement process is activated when a vehicle enters an MEC coverage area and it establishes a connection with a base station of the 5G network topology. The searching process for a host to which a client's service can be allocated is then started. This allocation can be done optimally if the selected host is located in the MEC datacentre that is directly linked to the base station. It might also happen that all the resources of the closest datacentre are busy, and then the host search continues in contiguous datacentres, leading to longer propagation times, as one or more hops throughout the network topology are then required.

Figure 5 represents the number of allocations during the simulation time, showing the distance between the allocated service and the MEC host in terms of the "optimal", "1 hop", "2 hops", or "cloud" distance for the highest workload we consider in our study based on the number of hops required to send the information between the different parts of the underlying three-layer fat tree network topology. It is clear that the datacentre infrastructure determines the impact of the latency, as in the scenario of 45 hosts (see in Figure 5a), where some of the services end up being placed in the cloud, as there are not enough resources in either the local MEC datacentre or the contiguous ones. This situation is solved by increasing the number of computing elements, as shown in Figure 5d, which considers 117 hosts, and it can be observed that almost all services are optimally allocated.

### *5.2. Edge Service Migration Performance*

When the MEC service starts in an MEC coverage area and the vehicle moves according to a specific trajectory through the geographical area, changing its connection from one base station to another, the edge service that is firstly allocated in the MEC system is, then, migrated to a host of the MEC datacentre that is attached to the new base station, i.e., local MEC datacentre. This migration can be considered as an "optimal" association, as the

delay obtained is the smallest possible delay. In case the service migration to the local MEC datacentre cannot be performed because there are not enough resources, then the MEC service remains behind, so we consider this situation as a "1 hop" or "2 hops" connection based on the number of hops of the fat tree network topology. The failed migration process is reattempted every time the "keep-alive" service poll runs, so it might happen that the service is finally migrated to the local MEC datacentre at some point. If the service has just started or the vehicle enters the MEC coverage area and there are not available resources in the local MEC datacentre, then the service is allocated to the cloud, and the migration is periodically reattempted.



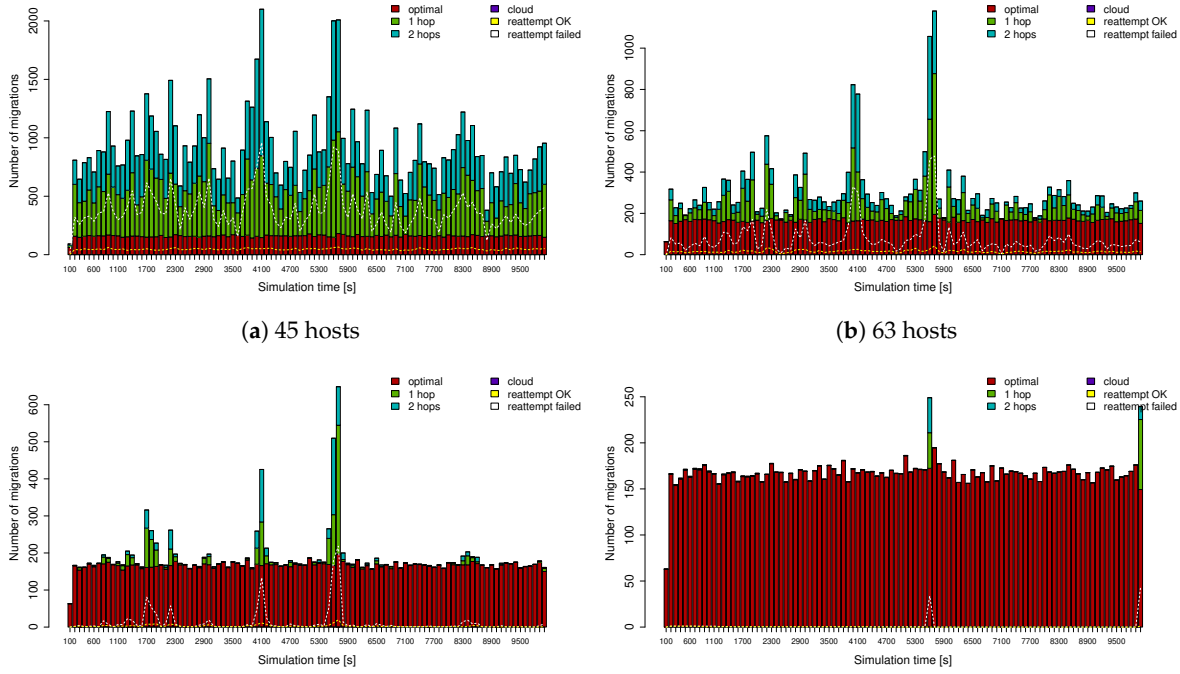

(**a**) 45 hosts (**b**) 63 hosts

(**c**) 81 hosts (**d**) 117 hosts

**Figure 5.** Number of allocations of edge services considering their distance to the MEC host that is providing the service.

In Figure 6, we show the average number of migrations performed throughout the simulation time, considering 5, 7, 9, 11, or 13 MEC hosts per MEC datacentre and, thus, a pool of 45, 63, 81, or 117 hosts in the whole MEC system. The number of services run in the simulation is in the range of 8600 and 8700 (they vary depending on the simulation run); therefore, we can observe that the migration information presented in Figure 6 is linked to the maximum workload considered in this study.

The graphical analysis shows how, with 45 hosts (see Figure 6a), the number of failed migrations, which are represented with the "1 hop" and "2 hops" metrics, is significantly high compared to the number of successful migrations, which are represented as the "optimal" migrations in dark red. We added a yellow dotted line showing the migrations that are reattempted and end up being successful in the next service poll. We also show the reattempts that fail to be optimal with a white dotted line. When increasing the number of MEC hosts to 63 (see Figure 6b), the number of failed migrations decreases, and this trend continues downward, as it is inversely proportional to the MEC host pool increase (Figure 6c,d). The results show that it is imperative to have enough resources in the MEC infrastructure to be able to support the projected number of services and their dynamics. The peaks that are present in the middle of the simulation are a representation of a situation wherein a large number of vehicles change their current base stations, requiring a corresponding MEC service migration. This is a situation where the MEC resource usage hits a peak because of the double resource reservation during the migration process.




(**a**) 45 hosts (**b**) 63 hosts

(**c**) 81 hosts (**d**) 117 hosts

**Figure 6.** Number of migrations of edge services considering their distance to the MEC host that is providing the service.

### *5.3. Edge Service Delay Analysis*

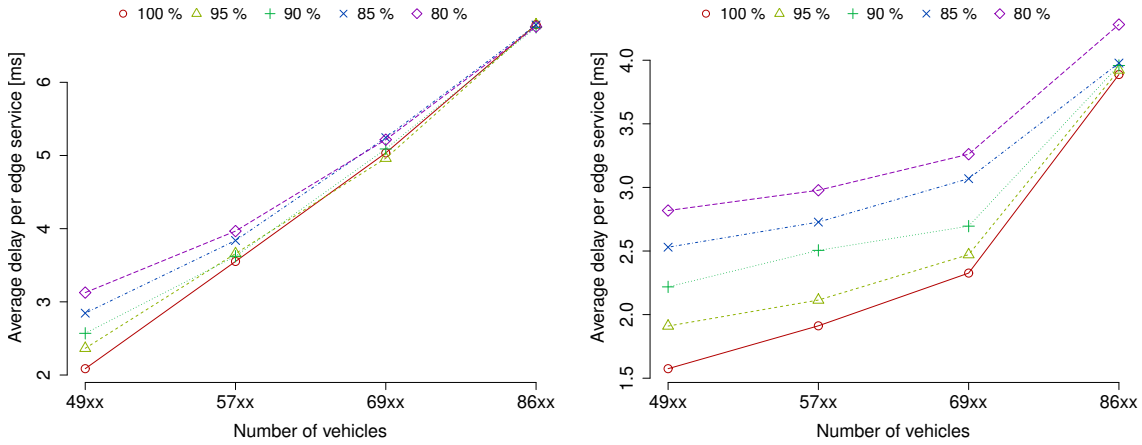
The service delay experienced by each vehicle when accessing the application in the MEC datacentre can be computed in many different ways that consider all the possible latencies that affect the final response time. In this paper, we consider the delay as the network time required to communicate with the MEC host where the virtual service is allocated with the network connection of the access point. The values computed based on the fat tree network delay available in CloudSim are: 1.57 ms for a communication through the edge layer, 2.45 ms for a communication through the aggregation layer, and 2.85 ms for a communication through the fat tree core layer.

In order to study the impact of prediction accuracy impact on the service delay, we depicted the relation of the average delay per edge service in contrast to the workload expressed as the number of vehicles in Figure 7. We can observe in Figure 7a that the prediction accuracy does not have much of an effect when MEC resources are scarce and especially when the service demand is high enough to leave a small idle margin after the consolidation process. Meanwhile, the delay differentiation is clearer in less dense scenarios. Improving the hardware infrastructure at the MEC layer allows us to clearly distinguish the effects on latency of having a range of reliability in prediction. Figure 7e shows that there can be more than 1.2 ms difference in delay between having prediction accuracies of 80% and 100%. In other words, when we compare it with the value of a delay per hop, this additional delay can be considered as adding an additional hop in the network.

### *5.4. Study of the Relation between Delay and Prediction Accuracy*

We can have a closer look at the effects of the accuracy level of the prediction in the resulting MEC service delay by showing the difference in percentage with the correct prediction delay versus the inaccuracy level in another set of images. In Figure 8, the relation of both errors is depicted; in the $y$ axis, the difference in percentage between the delay obtained with 100% correct prediction and the delay obtained with 95%, 90%, 85%, and 80% prediction accuracy is represented, and the $x$ axis represents the inaccuracy percentage of the prediction. We can observe that the relation of both metrics is almost linear when the vehicle density in the scenario is moderate and there are enough resources at the MEC

infrastructure. This relation is consistent and shows that when the infrastructure resources are not congested, an improvement in delays of almost 80% is obtained when the prediction inaccuracy is non-existent. Therefore, we can confirm that a good prediction related to service migration would have a consistent effect on the MEC service delay. However, we can also observe that even the best prediction cannot help much in reducing the delay when there is a lack of infrastructure resources for a defined workload.

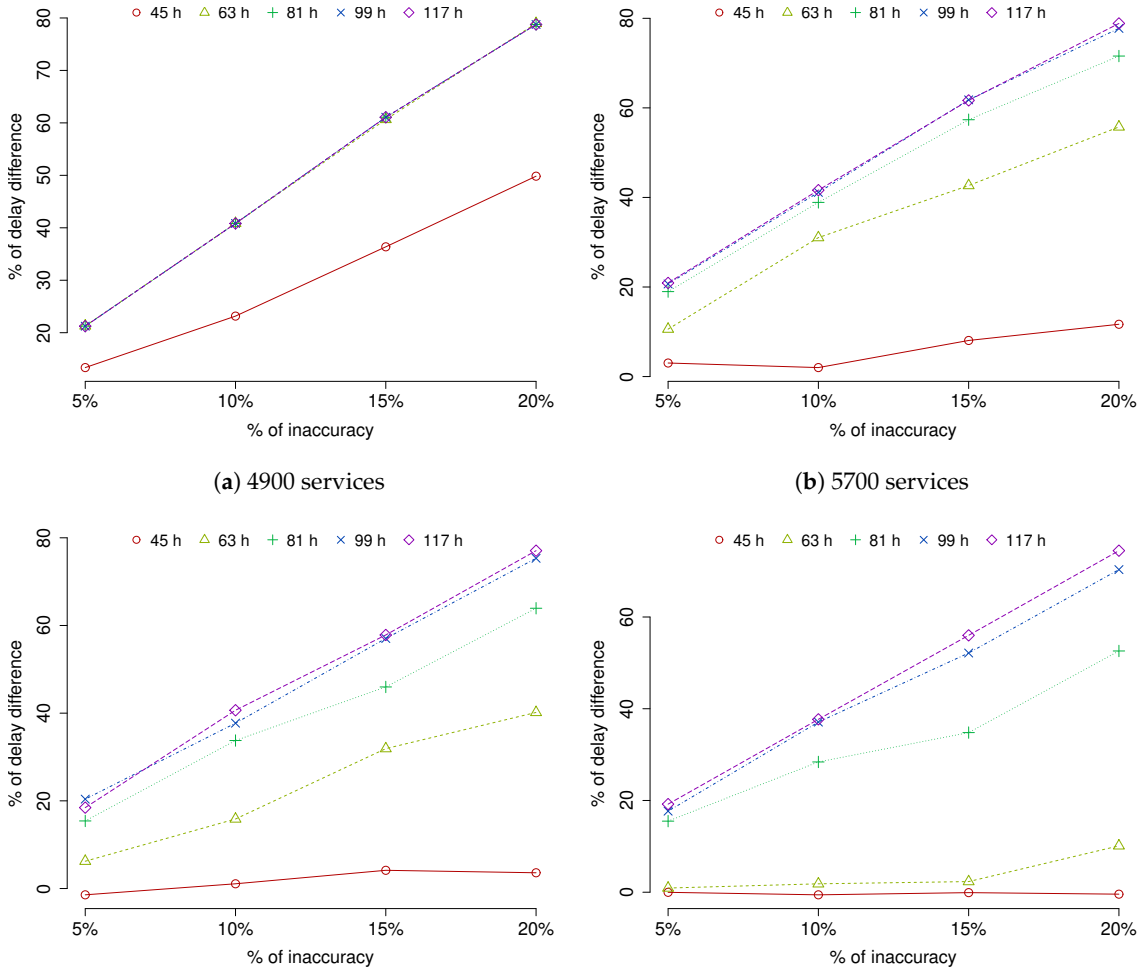



(**a**) 45 hosts (**b**) 63 hosts

(**c**) 81 hosts (**d**) 99 hosts

(**e**) 113 hosts

**Figure 7.** Average delay per MEC service considering the prediction accuracy.




(**a**) 4900 services

(**b**) 5700 services

(**c**) 6900 services

(**d**) 8600 services

**Figure 8.** Percentage of delay degradation compared to several levels of prediction accuracy.

## 6. Conclusions

Autonomous driving, automated traffic management, and other advanced vehicular services put a tremendous strain on the onboard computing power in vehicles. The use of a 5G network augmented with mobile edge computing servers can help alleviate this problem by providing additional virtual resources distributed in MEC server hosts located next to the 5G base stations. However, due to the high mobility of vehicles, intelligent resource management is needed to implement an efficient service migration in order to continuously maintain low latency while accessing the MEC services. For these purposes, a proactive approach to service migration based on employing predictive algorithms is necessary. In this paper, we have described how a hierarchical community-based resource management technique can be transformed from reactive to proactive behaviour by introducing a mobility prediction module. We then used this new implementation to analyse the performances of the MEC services in terms of latency experienced in a realistic urban vehicular scenario. The analysis, which was focused on determining how different accuracy rates of the prediction influence the average service latency, showed that the prediction errors induce additional latency, which increases linearly with the error independently of the number of active services.

**Author Contributions:** Conceptualisation, K.G. and S.F.; methodology, K.G., S.F., and S.A.; software, S.F.; validation, K.G. and S.A.; formal analysis, K.G.; investigation, K.G. and S.F.; resources, S.A.; data curation, K.G.; writing—original draft preparation, K.G. and S.F.; writing—review and editing, K.G.; visualisation, K.G. and S.A.; supervision, S.A.; project administration, S.A.; funding acquisition, S.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Shah, S.A.; Ahmed, E.; Imran, M.; Zeadally, S. 5G for vehicular communications. *IEEE Commun. Mag.* **2018**, *56*, 111–117. [CrossRef]
2. Zhang, H.; Liu, N.; Chu, X.; Long, K.; Aghvami, A.H.; Leung, V.C. Network slicing based 5G and future mobile networks: Mobility, resource management, and challenges. *IEEE Commun. Mag.* **2017**, *55*, 138–145. [CrossRef]
3. 3rd Generation Partnership Project. *Release Description; Release 16,* Technical Report 21.916, v. 0.60; 3GPP: Valbonne, France, 2020.
4. 5G Automotive Association. *A Visionary Roadmap for Advanced Use Cases, Connectivity Technologies, and Radio Spectrum Needs*; White Paper. v1.0; 5G Automotive Association: Munich, Germany, 2020.
5. Liu, J.; Wan, J.; Zeng, B.; Wang, Q.; Song, H.; Qiu, M. A scalable and quick-response software defined vehicular network assisted by mobile edge computing. *IEEE Commun. Mag.* **2017**, *55*, 94–100. [CrossRef]
6. Liu, L.; Chen, C.; Pei, Q.; Maharjan, S.; Zhang, Y. Vehicular edge computing and networking: A survey. *Mob. Netw. Appl.* **2020**, *25*, 1–24. [CrossRef]
7. Slamnik-Krijestorac, N.; Municio, E.; Resende, H.C.; Hadiwardoyo, S.A.; Marquez-Barja, J.M. Network Service and Resource Orchestration: A Feature and Performance Analysis within the MEC-Enhanced Vehicular Network Context. *Sensors* **2020**, *20*, 3852. [CrossRef] [PubMed]
8. Peng, H.; Ye, Q.; Shen, X.S. SDN-based resource management for autonomous vehicular networks: A multi-access edge computing approach. *IEEE Wirel. Commun.* **2019**, *26*, 156–162. [CrossRef]
9. Zhao, J.; Li, Q.; Gong, Y.; Zhang, K. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7944–7956. [CrossRef]
10. Hong, C.H.; Varghese, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–37. [CrossRef]
11. Carvalho, G.; Cabral, B.; Pereira, V.; Bernardino, J. Computation offloading in Edge Computing environments using Artificial Intelligence techniques. *Eng. Appl. Artif. Intell.* **2020**, *95*, 103840. [CrossRef]
12. Zhang, C.; Zheng, Z. Task migration for mobile edge computing using deep reinforcement learning. *Future Gener. Comput. Syst.* **2019**, *96*, 111–118. [CrossRef]
13. Miao, Y.; Wu, G.; Li, M.; Ghoneim, A.; Al-Rakhami, M.; Hossain, M.S. Intelligent task prediction and computation offloading based on mobile-edge cloud computing. *Future Gener. Comput. Syst.* **2020**, *102*, 925–931. [CrossRef]
14. Wu, C.L.; Chiu, T.C.; Wang, C.Y.; Pang, A.C. Mobility-Aware Deep Reinforcement Learning with Glimpse Mobility Prediction in Edge Computing. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–7.
15. Abdah, H.; Barraca, J.P.; Aguiar, R.L. Handover Prediction Integrated with Service Migration in 5G Systems. In Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–7.
16. ETSI Group Specification. *ETSI GS MEC 003 v2.1.1 Multi-Access Edge Computing (MEC): Framework and Reference Architecture*; ETSI Group Specification: Valbonne, France, 2019.
17. 5G Automotive Association. *C-V2X Use Cases Methodology, Examples, and Service Level Requirements*; White Paper; 5G Automotive Association: Munich, Germany, 2019.
18. Filiposka, S.; Mishev, A.; Gilly, K. Mobile-aware dynamic resource management for edge computing. *Trans. Emerg. Telecommun. Technol.* **2019**, *30*, e3626. [CrossRef]
19. Girvan, M.; Newman, M.E. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7821–7826. [CrossRef]
20. Gilly, K.; Alcaraz, S.; Aknin, N.; Filiposka, S.; Mishev, A. Modelling Edge Computing in Urban Mobility Simulation Scenarios. In Proceedings of the 2020 IFIP Networking Conference (Networking), Paris, France, 22–26 June 2020; pp. 539–543.