Mobility-aware Vehicular Cloud Formation Mechanism for Vehicular Edge Computing Environments

Joahannes B. D. da Costa[a,b,*], Wellington Lobato[a], Allan M. de Souza[a], Eduardo Cerqueira[c], Denis Rosário[c], Christoph Sommer[b], Leandro A. Villas[a]

[a] *Institute of Computing, University of Campinas (UNICAMP), Campinas, Brazil*
[b] *TU Dresden, Faculty of Computer Science, Dresden, Germany*
[c] *Federal University of Pará (UFPA), Belém, Brazil*

Abstract

Rapid advancements in vehicular technology and increased vehicle modernization have led to the emergence of intelligent and interconnected entities. As a result, the Vehicular Edge Computing (VEC) paradigm has gained prominence. This paradigm enables the provision of cloud computing services close to vehicular users by utilizing the idle computational resources of vehicles to execute tasks that require computing power beyond what is available locally. Aggregating these computational resources in the vehicular context is known as Vehicular Cloud (VCloud) formation. However, leveraging and aggregating these resources poses several challenges due to the dynamic nature of the vehicular environment. One of the main challenges is the efficient selection of vehicles to assume management roles in the distribution of computational power within the group, often referred to as leading vehicles. This research presents a mobility-aware mechanism called PREDATOR to enhance the VCloud formation process. In this mechanism, the Roadside Unit (RSU) provides vehicular mobility predictions, enabling the selection of the most stable vehicles within the RSU coverage area to assume leadership roles in the VCloud. In this context, vehicle stability is associated with a vehicle's time within the RSU coverage area, known as dwell time. PREDATOR employs a microscopic perspective to select vehicles with the longest dwell time in the VCloud, allowing for efficient management of computational resource utilization. Simulation results have demonstrated that PREDATOR not only increases the VCloud lifetime but also minimizes leader changes, reduces network message exchange, mitigates packet collisions, and facilitates the effective utilization of aggregated vehicular resources compared to state-of-the-art approaches.

Keywords: Mobility Prediction, Vehicular Edge Computing, Vehicular Cloud formation, Clustering

1. Introduction

In recent years, vehicles have become increasingly more intelligent and connected [1, 2]. Furthermore, vehicles can exchange data with other entities through Vehicle-to-Everything (V2X) communication [3]. However, this evolution can dramatically increase bandwidth consumption and potentially create network congestion at the network core due to the growing number of connected vehicles and vehicular applications [4]. In this sense, the Vehicular Edge Computing (VEC) paradigm emerges to make it possible to provide cloud services closer to the vehicle users, which aims to group and use the computational resources of vehicles as a service to run tasks that require computing power above than supported locally [5]. In short, VEC is the union of Mobile Edge Computing (MEC) and cloud computing concepts, where vehicles become active in using and providing computing resources to the Vehicular Ad-Hoc Network (VANET) [6].

VEC considers a set of Vehicular Clouds (VClouds) to allow vehicular users to request resources to meet application demands [7, 8]. Specifically, VCloud groups up computational resources, such as processing units and storage capacity, available in a set of vehicles (*i.e.*, either moving or parked vehicles) and infrastructure (*i.e.*, Roadside Units (RSUs), 5G Base Stations (BSs), and Remote Servers (RS) in the Internet cloud) nodes to provide cloud services [9]. In other words, some vehicles increase their computing capacity by using VEC resources, while others lend their available resources to VCloud. Hence, VEC avoids the underutilization of vehicular computing resources by using idle resources to be managed and utilized by other vehicles. In this context, there is a need to rely on VCloud formation mechanisms for grouping these vehicles in VClouds, where VCloud formation occurs by creating groups with a set of vehicles in a geographic region sharing the same

*Corresponding author
Email addresses: joahannes.costa@ic.unicamp.br (Joahannes B. D. da Costa), wellington@lrc.ic.unicamp.br (Wellington Lobato), allanms@lrc.ic.unicamp.br (Allan M. de Souza), cerqueira@ufpa.br (Eduardo Cerqueira), denis@ufpa.br (Denis Rosário), leandro@ic.unicamp.br (Leandro A. Villas)
URL: www.ic.unicamp.br/~joahannes.costa (Joahannes B. D. da Costa), www.cms-labs.org/people/sommer/ (Christoph Sommer)

Elsevier Ad Hoc Networks (Volume 151, December 2023) Digital library entry → https://doi.org/10.1016/j.adhoc.2023.103300

preferences, such as direction, trajectory, path similarity, etc [10, 4].

The VCloud formation mechanisms rely on the communication infrastructures to assist the VClouds formation process [11]. In this sense, vehicles under infrastructure coverage are considered members of this VCloud. This is because, in a macroscopic view, these VClouds can be inter-communicable through infrastructures deployed in cities, allowing cooperation between them [12]. In this work, we consider moving vehicles in specific geographic regions in the city, *e.g.*, intersections, which are covered by RSUs. Specifically, intersections are an appropriate location for setting up VClouds as vehicles traveling on multiple road segments can meet at one place [13]. Unlike distributed approaches for VCloud formation, we consider that vehicles do not have autonomy in defining the roles of Vehicular Cloud Head (VCloudHead) or Vehicular Cloud Member (VCloudMember). In this case, the RSU performs the required inferences for these definitions and informs vehicles. This definition is essential to take advantage of the communication infrastructures naturally available in cities, with the broad expansion of 5G networks [14], and considerably reduce the number of maintenance messages that fully-distributed approaches need in their decision processes.

However, high vehicular mobility is the main challenge for proposing efficient VCloud formation mechanisms since vehicle mobility causes several variations in network topology and intermittent connections [15]. For example, it may happen that the connection between the client vehicle and the vehicles processing in the VCloud does not last until the scheduled tasks finish their execution, which generates rescheduling processes and increases the cost of using resources. Hence, vehicular mobility impacts the system efficiency by fluctuating the number of available resources in the VCloud, causing the loss of task processing results and impacting the number of tasks attended/scheduled [16]. In this context, existing works have considered mobility information in their decision-making processes to mitigate the impact of vehicular mobility [17, 18, 19]. Although mobility is considered in such works, they do not involve mobility in their optimization problems, and mobility is like a trigger condition for other processes [20]. Hence, providing a mobility-aware VCloud formation mechanism to select the most stable vehicles to lead each VCloud and make managing more stable and reliable is still an open issue [21].

In this context, this article describes a mobility-aware mechanism called PREDATOR that enhances the VCloud formation process. The mechanism leverages the vehicular mobility predictions provided by RSUs to select the most stable vehicles within the RSU coverage area to lead the VCloud, thereby increasing the VCloud's lifetime. Vehicle stability is measured by the dwell time, representing the duration a vehicle spends within the RSU coverage area [13]. To achieve this, RSUs receive contextual information from vehicles through the natural beacon exchange in VANET. This information is aggregated at the city intersections at certain intervals, and the RSU executes the VCloud formation process. The number of VClouds formed is proportional to the number of RSUs in the scenario. Additionally, a Network Controller at a higher level in the network can oversee these VClouds and manage their aggregated resources. Simulation results demonstrate the superior performance of PREDATOR compared to other VCloud formation mechanisms. The results highlight the benefits of PREDATOR, including a 42.15 % longer VCloud lifetime, 45.81 % lower number of VCloud leader changes, 48.51 % lower number of messages exchanged on the network, 43.55 % lower number of packet collisions, and the ability to serve 24.68 % more requests for vehicular computational resources.

This article extends our previous work [22] by introducing a detailed description of PREDATOR, including all algorithms, a new function to select stable vehicles, a detailed explanation of the mobility prediction aspects, and an evaluation of the impact of different mobility traces to choose more stable nodes in each VCloud. We also introduced a comprehensive assessment in a more challenging and realistic VANET scenario (*i.e.*, Luxembourg city [23]). Therefore, the contributions of this work can be summarized as follows:

- We describe an efficient mechanism for the VCloud formation assisted by communication infrastructures, which increases the VClouds lifetime and decreases the number of sent messages on the network.
- We consider vehicular mobility prediction information to enhance the decision-making process in vehicle selection to lead each VCloud, making VClouds more stable and lessening their management burden.
- In a detailed performance evaluation with different mobility traces (synthetic and realistic), we show the necessity and outline the benefits of mobility prediction information for the VCloud formation process compared to other state-of-the-art approaches.

This article is organized as follows. Section 2 presents and discusses the related works. Section 3 presents the system model and operation of PREDATOR. Section 4 discusses the performance evaluation and results obtained. Finally, Section 5 provides final remarks and discusses future work.

2. Related Work

This section presents state-of-the-art research on grouping network devices to use their computational resources for different purposes. We have analyzed these studies focusing on their applicability in VCloud formation scenarios, which typically utilize conventional clustering concepts in VANETs [24, 25]. Furthermore, we have categorized the related works into three perspectives based on

the level of infrastructure support, the grouping strategy employed, and the integration of mobility information into the decision-making process.

For instance, Zhao *et al.* [26] proposed an algorithm to form mobile device groups considering social factors, particularly the degree coefficient, to maximize network throughput among the grouped devices. The algorithm selects leaders and members for each group based on social attributes and physical factors such as community, connections, and geographic proximity. While this approach is generic, it can also be applied in vehicular environments. Similarly, Kamakshi *et al.* [27] introduced an algorithm based on graph modularity gain and the degree of cohesion between vehicles to form stable vehicle groups. In summary, the algorithm selects forwarders (*i.e.*, Vehicle-to-Vehicle (V2V) communication hubs) for safety messages by creating stable communities of vehicles, considering their relative mobility. The authors consider the relative mobility between communities during maintenance, specifically for community aggregation and separation.

Da Costa *et al.* [6] introduced a mechanism that considers the VCloud formation based on the Density-Based Spatial Clustering of Application with Noise (DBSCAN) clustering algorithm. In short, DBSCAN identifies groups based on the spatial density of individuals. The proposed mechanism centrally obtains the positioning of vehicles and executes DBSCAN to identify the VClouds. Besides, the choice of the VCloudHead in this approach is carried out by calculating the centroid in the spatial distribution of the VCloud and identifying the vehicle closest to this centroid. Peixoto *et al.* [28] introduced a data clustering framework to reduce traffic information at the edge of vehicular networks by exploiting fog computing. The proposed data clustering framework defines two methods for the reduction of the traffic data stream: The baseline method, which is an ordinary traffic congestion detection approach, and two adapted clustering methods for a data stream, namely, the Ordering Points to Identify the Clustering Structure (OPTICS) and the DBSCAN.

Bute *et al.* [11] proposed a cluster-based cooperative task offloading scheme for cellular-vehicle networks. In this case, a clustering of vehicles is achieved by employing the fuzzy logic algorithm. For cluster formation and VCloudHead selection, VCloudHead is chosen based on a distributed algorithm. Three metrics are considered to form VClouds: k-connectivity, link reliability, and relative distance. These metrics are to ensure stable connectivity between nodes for reliable communication. After receiving beacon messages, each vehicle evaluates its suitability value and each one-hop neighbor in the communication range. The vehicle with the highest suitability value declares itself the VCloudHead.

Abbasi *et al.* [29] presented a fuzzy logic-based vehicle weighting model for scheduling prioritized data in VANETs, called FWDP. FWDP employs RSUs to dominate the frequent topology changes and manage the data propagation. A Fuzzy C-Means Clustering (FCMC) is used for handling clustered vehicles that compete to utilize the shared channel. RSUs receive prioritized data from VCloudHeads in the proposed model, wherein VCloudHeads allocate scheduled service channels to weighted vehicles during an interval. FWDP is equipped with a Fuzzy Inference System (FIS) for vehicle weighting according to the velocity and inter-vehicle distance metrics. In addition, RSUs compute the signal-to-noise and the interference ratio (SINR) to solve the hidden terminal problem to prevent radio interference. Also, FWDP uses mobility information to estimate vehicular density and prioritize the vehicles that should propagate messages. Zhao *et al.* [19] proposed an adaptive vehicle clustering approach based on a fuzzy C-means algorithm to minimize vehicle power consumption. Specifically, the proposed algorithm dynamically allocates the computing resources of each virtual machine in the vehicle according to the popularity of different virtualized network functions. The optimal clustering number to minimize the total energy consumption of vehicles is determined using the fuzzy C-means algorithm, and the VCloudHead is selected based on a vehicle moving direction, weighted mobility, and entropy.

Hagenauer *et al.* [9] presented a map-based approach to VCloud formation that selects the most central vehicle (close to the centroid) in the region covered by an RSU. The formed clouds can provide services in their vicinity, and together they form larger VClouds, allowing for more complex services and covering entire cities. This work shows that an efficient VCloud formation enables vehicular applications to use aggregated computational resources efficiently. In this approach, the VCloud formation process is limited to intersections in urban environments and the VClouds size is limited to the communication radius of the selected VCloudHead. Also, the primary metric to determine VCloudHead is the vehicle position about the intersection centroid. Wang *et al.* [30] proposed a network representation learning to achieve accurate vehicle trajectory clustering. Specifically, the authors dynamically construct the K-Nearest Neighbor (KNN)-based vehicle groups. Then they discover the low-dimensional representations of vehicles by performing dynamic network representation learning on the constructed network. Finally, vehicle trajectories are clustered using Machine Learning (ML) methods using the learned vehicle vectors. Magaia *et al.* [18] introduced a vehicular clustering algorithm at the edge of the network and an efficient message routing approach, which is known as Group'n Route (GnR). Both mechanisms resort to machine learning and graph metrics reflecting the nodes' social relationships. The performance evaluation reveals that the clustering algorithm yields stable results with varying road scenarios.

Table 1 summarizes the main characteristics of reviewed studies regarding grouping strategy, the assistance provided by communication infrastructures, and the use of vehicular mobility information in the decision-making phase. Based on our state-of-the-art analysis, we conclude that it is essential to consider predicted mobility information in

Table 1: Summary of related works and comparison with our PREDATOR approach.

Work	Grouping strategy	Infrastructure-based	Mobility information
Zhao et al. [26]	Degree centrality	✓	
Hagenauer et al. [9]	Centroid	✓	
Da Costa et al. [6]	DBSCAN	✓	
Peixoto et al. [28]	OPTICS & DBSCAN	✓	
Bute et al. [11]	Fuzzy		
Kamakshi et al. [27]	Centrality metrics		Relative mobility
Abbasi et al. [29]	Fuzzy C-Means	✓	Vehicular density
Zhao et al. [19]	Fuzzy C-Means	✓	Moving direction
Wang et al. [30]	KNN + Machine Learning	✓	Trajectory
Magaia et al. [18]	Social + Machine Learning	✓	Social contacts
Da Costa et al. [22]	Dwell time	✓	Mobility prediction
PREDATOR	Dwell time + Distance	✓	Mobility prediction

the decision process. This is because mobility prediction provides a temporal layer for spatial data, making it possible further to explore the social relationships between vehicles in the network. In addition, considering communication infrastructures is essential to increase the reliability of control rules. In summary, our work can complement the others since it uses the vehicles' dwell time in VClouds for decision-making. This knowledge is essential to improve many processes, such as assisting in resource management, efficient data dissemination, cooperative data processing and perception, and location-based content aggregation. Compared to our previous work [22], called NEMESIS, the main improvement of this work is the inclusion of the distance factor in the decision-making equation. In this case, our new approach considers the balance between decision metrics (distance and dwell time), effectively reducing the likelihood of disconnections between the leading vehicle and the RSU. Additionally, this mechanism operates in city intersections, increasing the connection probability among VCloud members and enhancing resource management capacity [13].

3. PREDATOR

As discussed above, vehicular mobility directly impacts the approaches to forming VClouds due to the dynamic nature of moving vehicles. VCloud formation involves grouping and utilizing computational resources available in moving and parked vehicles to provide services and applications to users. However, vehicular mobility introduces significant challenges, such as variations in resource connectivity and availability and the possibility of disruptions in communication between vehicles. This makes the VCloud formation process more complex.

A solution aware of vehicular mobility can overcome the challenges of high mobility for VCloud formation. This approach should take real-time information about the location, speed, and availability of vehicles into account to perform efficient grouping strategies and task scheduling. Additionally, after the VClouds are formed, adaptive algorithms and load-balancing strategies can be implemented to optimize the task scheduling among vehicles grouped in these VClouds, ensuring that user demands are effectively met even in highly dynamic environments. The formation solution must rapidly adapt to changes in mobility conditions, dynamically and intelligently allocating resources to enhance the performance and reliability of the VCloud. Thus, vehicular mobility prediction information can be leveraged as an advantage for forming more robust and resilient VClouds.

With that in mind, and aiming to overcome the challenges posed by vehicular mobility, this section describes the details of PREDATOR, where we consider a scenario composed of multiple VClouds assisted by RSUs. PREDATOR considers a mobility prediction model for selecting the most stable vehicles to coordinate each VCloud. We present some essential assumptions, detail the system architecture used, and define the problem. Finally, we present the PREDATOR operation with a description of the algorithms that compose it.

3.1. Assumptions

The following assumptions are made for PREDATOR to work properly.

- Each vehicle can communicate with other devices through V2X communication.
- Each vehicle has limited computational resources, such as processing power and storage capacity, that can be shared with other interested entities.
- Every vehicle under RSU coverage participates in the VCloud formation process. That is, we do not consider selfish vehicles. Also, all vehicles are ready to offer their computing power and storage capacity to cloud services.

- Each vehicle is aware of its real-time location through the global navigation satellite system, such as Global Positioning System (GPS).
- A VCloud starts when an RSU selects a new VCloudHead, and this VCloudHead receives the warning message from an RSU.

3.2. Network and System Model

We consider a scenario composed of x vehicles, where each vehicle $u_i \in U$ has a unique individual identification ($i \in [1, x]$) and is equipped with an IEEE 802.11p compliant radio transceiver, which enables V2X communication. Also, we consider k RSUs deployed in some intersections in the city. Intersections are appropriate for setting up VClouds as vehicles traveling on multiple road segments can meet at one place [13]. Vehicles periodically send beacon messages on the network, and thus an RSU collects this information in real-time to build the knowledge necessary for its decision-making. It is important to highlight that Information such as position, speed, direction, and computational resources are added to the beacon message. Besides, we denote each VCloud as $v_j \in V = \{v_1, \ldots, v_m\}$, which consists of a subset of vehicles $V \subset U$ capable of sharing computational resources, such as processing power and storage capacity. An RSU assists a VCloud directly, so VClouds' number equals the number of RSUs.

In this scenario, vehicles can take on two roles. The first role is VCloudHead, representing the vehicle elected as the leader of VCloud by an RSU. The second role is VCloudMember, the vehicles that will share their computing resources on the network through a VCloud. Therefore, when a vehicle is associated with an RSU, it automatically becomes part of the VCloud assisted by that RSU. The primary function of an RSU is to select the vehicle leading this VCloud. Considering the communication range of RSUs is greater than that of vehicles, the lead vehicle can be n hops away from the other VCloud members. After the RSU selects and informs the VCloudHead of this VCloud, this vehicle notifies the other VCloudMembers through broadcast messages.

Finally, after an RSU selects its local VCloudHead, it provides this information to a network controller. The controller has a connection with all the RSUs, having a global view of the network and all VClouds. In this case, the controller does not actuate directly in the VCloud formation process. It is responsible only for information aggregation sent by RSUs after ending the VCloud formation process. So, VANET applications that need computational resources can request these resources on the network, and the controller decides which VCloud runs the tasks of these applications [6].

Figure 1 presents the system architecture. As in a modern scenario, some RSUs are deployed in the city. Each RSU is responsible for covering a specific area, which in this case is a road intersection. In this way, vehicles in the intersection are covered by the RSU, which aggregates and

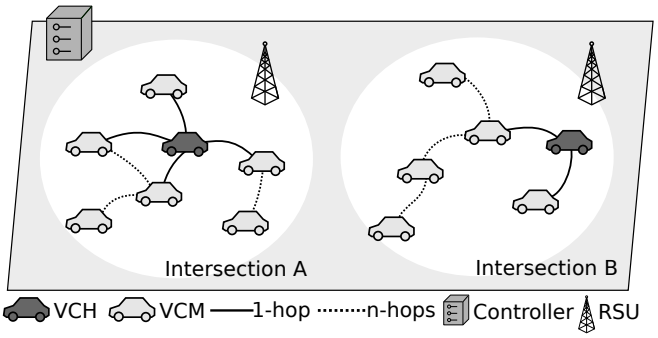


Figure 1: System architecture (VCH: Vehicular Cloud Head (VCloudHead); VCM: Vehicular Cloud Member (VCloudMember).

maintains knowledge of how many vehicles are under their coverage. Each RSU can also apply algorithms to predict vehicular mobility and estimate vehicles' dwell-time under its coverage, thus choosing the most stable vehicle to lead this VCloud (VCloudHead). Besides, it is considered a scenario where the RSU has a higher communication range than vehicles. This way, V2V communication within the VCloud can occur over more than one hop.

In summary, PREDATOR considers two phases in the VCloud formation process, namely communication and information. The communication phase receives and aggregates contextual information from neighboring vehicles. At this phase, the RSU maintains a neighborhood structure with all vehicles in its coverage and manages them based on the receiving signaling messages (beacons). Each beacon message contains the vehicle's information, such as position, speed, direction, and route. The information phase is responsible for informing the new VCloudHead about its role in the network. In this way, the RSU sends a message containing the VCloud identification, the VCloudHead identification, and the number of resources available in that VCloud. After that, the VCloudHead informs its neighboring vehicles, which now become VCloudMembers, and those VCloudMembers relay this warning message. With each message retransmission, the number of hops is incremented so that the receiving knows its distance to the VCloudHead.

3.3. Problem definition

In essence, mobility prediction algorithms estimate a given vehicle's position from current and/or past information. With the forecast information, it is possible to know if, in a future moment, the vehicle will be part of a specific VCloud. It is essential to treat mobility as a time series, where each measurement constitutes an input provided to the predictor engine to adjust the prediction model [31, 32]. Also, the forecast granularity can be defined based on the VCloud formation intervals.

The vehicular mobility pattern makes it possible to model a mathematical system to predict the future geographic positions of nodes [33]. Thus, consider $P_{u(t)} = (X_{u(t)}, Y_{u(t)})$ being the vehicle's position at the current

time t and $P_{u(t+1)} = (X_{u(t+1)}, Y_{u(t+1)})$ the vehicle's position at time $t+1$. As mentioned, geographic position data can be consulted through digital maps and GPS. Therefore, when vehicles associate with an RSU and provide information, such as current location, speed, direction, and available resources, the RSU can aggregate this information and create the VCloud for location-based services. However, an RSU maintains a macroscopic view of the VCloud it manages.

Building decision-making closer to these vehicular resources is necessary to guarantee real-time requests' fulfillment. In this way, the RSU selects the most stable vehicle (with higher dwell time) to lead the VCloud, and this vehicle is called VCloudHead. However, this VCloudHead definition is one of the critical points in the VCloud formation process. This VCloudHead will receive the rules from the controller and manage the task scheduling/resource allocation among the VCloudMembers. The system's overall efficiency can be degraded depending on the selection criteria. The high number of VCloudHead changes leads to an increased number of warning messages in the network and, consequently, a high number of packet collisions. In addition to degrading the efficiency of the VEC system, it will flood the network with unnecessary transmissions.

3.4. PREDATOR's operation

Based on the vehicle's predicted positions $P_{u(t+n)}$, considering a T time window, PREDATOR can check if each position $p \in P_{u(t+n)}$ is under the coverage of a given RSU. As the prediction returns the positions for a given time window T, the vehicle coverage time is obtained by incrementing it by a 1-time unit if and only if the position is under the coverage of an RSU. In short, an RSU checks its distance to each predicted position and computes a unit of time for coverage if the result is less than or equal to its communication range. It is important to note that PREDATOR can work with any model for mobility prediction existing in the literature, as long as it returns good prediction results. Yet, an RSU maintains a neighborhood structure to store information about vehicles in its coverage. An RSU checks if this vehicle is already in this structure with each beacon message receipt. Case negative, the vehicle is added, along with information present in the beacon.

After aggregating vehicle information, the RSU decides who will be the VCloudHead of its VCloud, applying the Equation (1) for each vehicle. The calculation is subtracted from 1 to keep it in the range $[0, 1]$. PREDATOR selects the vehicle with a maximum S_u in each VCloud.

$$S_{u_i} = 1 - \left(\alpha \times \bar{A} + \beta \times \bar{B}\right) \tag{1}$$

where α and β represent weights to define the importance of the metrics ($\bar{A}$ and $\bar{B}$) in the equation. That is, if $\alpha = 0.8$ and $\beta = 0.2$, it means that portion $\bar{A}$ has greater relevance than portion $\bar{B}$. The portion $\bar{A}$ represents the distance between the vehicle and the RSU, as shown in Equation (2). In this case, the vehicle closest to the RSU will be prioritized to guarantee greater stability in the connection. R_{max} represents the RSU's estimated communication range and $dist$ is the distance from the vehicle u_i to the RSU.

$$\bar{A} = \left(\frac{dist}{R_{max}}\right) \tag{2}$$

On the other hand, portion $\bar{B}$ of the Equation (1) defines how the vehicle's dwell time in the RSU's coverage is considered. In this case, the vehicle with the longest dwell time will be prioritized, hence the division between 1 and $DwellTime$ shown in Equation (3). $DwellTime$ is the dwell time calculated by checking the predicted positions.

$$\bar{B} = \left(\frac{1}{DwellTime}\right) \tag{3}$$

To illustrate the calculation of Equation (1), imagine that vehicles u_1 and u_2 are in the RSU's coverage. The RSU has a $R_{max} = 250$m. After the beacon exchange and the prediction process, RSU knows that vehicle u_1 has $dist = 50$m and $DwellTime = 10$s and the vehicle u_2 has $dist = 100$m and $DwellTime = 12$s. Also, parameters α and β are set to 0.5. In other words, both are equally important in this example.

Applying the equation, we have for vehicle u_1:

$$\begin{aligned} S_{u_1} &= 1 - \left(0.5 \times \left(\frac{50}{250}\right) + 0.5 \times \left(\frac{1}{10}\right)\right) \\ &= 1 - (0.5 \times 0.2 + 0.5 \times 0.1) \\ &= 1 - (0.1 + 0.05) \\ &= 1 - (0.15) \\ &= 0.85 \end{aligned}$$

Also, we have for vehicle u_2:

$$\begin{aligned} S_{u_2} &= 1 - \left(0.5 \times \left(\frac{100}{250}\right) + 0.5 \times \left(\frac{1}{12}\right)\right) \\ &= 1 - (0.5 \times 0.4 + 0.5 \times 0.084) \\ &= 1 - (0.2 + 0.042) \\ &= 1 - (0.242) \\ &= 0.758 \end{aligned}$$

Therefore, RSU selects vehicle u_1 to lead its VCloud because it has a higher S value. This means that even the vehicle having a lower $DwellTime$, the possibility of an intermittent connection between the lead vehicle and the RSU is minimized, also reducing the need for control messages on the network.

Figure 2 presents a visual example with more details of the proposal. The goal is to know the vehicles' dwell time under the RSU coverage and, consequently, its dwell time in the VCloud. In this example, the vehicle remains 25 s in the RSU of Intersection A. As some prediction methods require a base of past information to make future estimates and vehicles transmit beacons with a frequency of 1 Hz,

each RSU stores the mobility information of all vehicles in its coverage to build a dataset for the prediction method. That is, if the prediction is performed at the instant $t = 15$ with a time window of 10 s, the information for building the database will range from $t = 0$ to $t = 15$ and, thus, it will be identified that this vehicle will no longer be part of the VCloud of Intersection A at the instant $t = 25$. The RSU only has local knowledge of its coverage. However, the controller can aggregate the local knowledge of each RSU and build its global knowledge. In this VCloud formation stage, the controller has no role other than to keep the information of each RSU for eventual resource orchestrations.

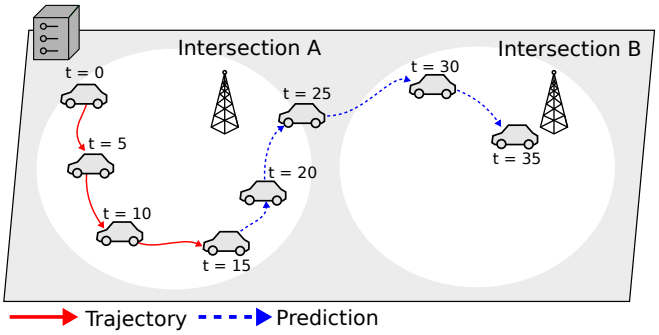


Figure 2: PREDATOR's microscopic vision.

In this way, Algorithm 1 shows an abstraction of PREDATOR. The set of vehicles N_k and the time window T for the forecast are provided to PREDATOR. Initially, the algorithm checks that the set N_k is not empty, which means some vehicles are in the RSU coverage. If N_k is empty, the resources of this VCloud are indicated as 0, and PREDATOR informs the controller about this information (Line 19). Otherwise, each vehicle is verified and its shared resources are aggregated for the VCloud (Line 7). The predicted positions are added to a temporary set *predPositions* that contains the vehicle identification and the list of estimated positions. In this step, we consider the ARIMA model with the vehicle's past positions dataset and time window T, which indicates how many time units the model will try to predict (Line 8). As mentioned earlier, the distance between the vehicle and the RSU is crucial in decision-making. In this way, this distance is calculated and stored (Line 9). All predicted positions are verified, and if one of these positions is within RSU coverage, a dwell time in the VCloud is incremented in a 1-time unit (Lines 11 and 13). With the distance *dist* and dwell time *DwellTime* calculated, the Equation (1) is applied, and the vehicle score is stored in a control list S (Lines 14 and 15). After all checks, PREDATOR selects the vehicle with the highest score to become VCloudHead of the respective RSU k (Line 16). Finally, with the VCloudHead identified, the RSU sends the message on the network and informs the controller about the created VCloud (Lines 17 and 18).

On the other hand, the Algorithm 2 presents the process when the VCloudHead receives the RSU warning message. In our system, vehicles can assume three states: (i) Undefined (UNDEF), which means undefined state; (ii) VCloudMember, which represents VCloud's member vehicles; and (iii) VCloudHead, which represents the leader of the VCloud. Before any process, all vehicles have been set to UNDEF. Therefore, when receiving a broadcast message from the RSU, the vehicle checks its current status, and if it is UNDEF, it means that the vehicle is not yet part of any VCloud (Line 2). The vehicle keeps the RSU's *id* received in the broadcast message (Line 3). Then, the vehicle checks if its identification number is included in this message. If so, its state changes to VCloudHead (Line 5) and starts the leadership in this VCloud (Line 6). The VCloudHead must inform its neighbors about its new role in the network. For this, it creates a VCloudHead message, adds its identification, the identification of the RSU that supports it, the number of hops to the VCloudHead (which in this case is 0 because it itself is the VCloudHead), and sends the message on the network (Lines 7 to 10). If, when receiving a broadcast message from the RSU, the vehicle's status is not UNDEF, it checks if it already acts in the role of VCloudHead (Line 12). The vehicle checks if it is still listed as a VCloudHead, and if not, it ends its leadership at this point (Line 13 to 15).

Additionally, to illustrate the entire message exchange

Algorithm 1: Abstraction of PREDATOR

```
   Input: vehicles set N_k, time window T
1  S ← ∅
2  rsu ← Roadside Unit (RSU) k's identifier
3  resource ← 0
4  VCloudHead ← NULL
5  if N_k ≠ ∅ then
6      foreach u ∈ N_k do
7          resource ← resource + u.resource
8          predPositions ← ARIMA(u.dataset, T)
           ▷ Current distance between u and rsu
9          dist ← distance(u.pos, rsu.pos)
10         DwellTime ← 0
11         foreach p ∈ predPositions do
12             if distance(p, rsu.pos) ≤ rsu.R_max
                then
13                 DwellTime ← DwellTime + 1
           ▷ Use DwellTime and dist in this point
14         u.score ← Equation (1)
           ▷ Store the vehicle's score
15         S.append(u.score)
16     VCloudHead ← max{S}
17     SendBroadcastMsg(rsu, VCloudHead)
18     SendStatusMsg(rsu, VCloudHead, resource)
19 else
20     SendStatusMsg(rsu, VCloudHead, resource)
```

Algorithm 2: VCloudHead receiving broadcast message from the RSU

```
   Input: msg
1  if broadcast message from the RSU then
2      if status = UNDEF then
           ▷ Checks based on RSU's id
3          myRSU ← msg.rsu
4          if myId = msg.VCloudHead then
5              status ← VCloudHead
6              VCloudStarted(currentTime())
               ▷ VCloudHead message
7              msg.VCloudHead ← myId
8              msg.RSU ← myRSU
9              msg.hops ← 0
               ▷ Send message to neighbors
10             SendVCloudHeadMessage(msg)
11     else
12         if status = VCloudHead then
13             if myId ≠ msg.VCloudHead then
14                 status ← UNDEF
15                 VCloudDied(currentTime())
```

Algorithm 3: Receiving relay message

```
   Input: msg
   ▷ Retransmissions just one hop beyond the RSU's
     R_max
1  if msg.hops ≤ 3 then
2      if status ≠ VCloudHead then
3          Create relay message msg
4          Update status for VCloudMember
5          Store my current VCloudHead
6          Update my current RSU
7          Increment number of hops by 1
           ▷ Send a message if you haven't sent it yet
8          if message not yet sent then
9              SendBroadcastMsg(msg)
```

process on the network, some actions must be taken when the VCloudHead informs its neighbors about its role in the VCloud. Algorithm 3 presents an abstraction of this phase. The receiving vehicle has two possibilities when receiving a message from another vehicle. The first is that this neighbor is a VCloudHead and the second is that it is a VCloudMember. However, the actions taken in both cases are the same. First, the number of hops in the message is verified to limit retransmissions in the information region of interest, which is the coverage of the RSU (Line 1). After that, the vehicle checks if its state is UNDEF or VCloudMember (Line 2). If so, the vehicle creates a relay message and updates some information, such as its status becomes VCloudMember, stores its current VCH, stores its current RSU, and increases the number of hops in the relay message (Lines 3 to 7). As the environment is distributed at this stage, the vehicle may receive this message several times from other neighbors. Hence, checking whether the vehicle has retransmitted this message at this interval is necessary to reduce the number of duplicate messages on the network (Lines 8 and 9).

Regarding the mobility prediction, we consider the ARIMA model for our evaluations. ARIMA is a statistical model for analyzing and predicting time series and works by taking series values and making them stationary if necessary. A stationary time series has no trend, and the amplitude of its variations around the mean is constant. Future series values are considered a linear combination of past values and past moving averages in the ARIMA model. ARIMA is described as a 3-tuple (p, d, q), where p is the number of past measurements weighted in the estimate, d is the number of differentiation series to make statistically stationary, and q is the number of previous moving averages. The basic formulation of the model is given by Equation (4). We denote past terms as l, past moving averages as μ, while θ and Φ are individual weights for each term and will be model trained.

$$l_t = \theta_0 + \Phi_1 l_{t-1} + \Phi_2 l_{t-2} + \cdots + \Phi_p l_{t-p} - \theta_1 \mu_{t-1} - \theta_2 \mu_{t-2} - \cdots - \theta_p \mu_{t-p} \quad (4)$$

In summary, the number of past value terms and moving averages depends on the series considered. Some series mainly depend on weighted past values and do not need moving average terms. The model can be represented by the ARIMA(3, 1, 0) notation, which means three previous terms are used, a differentiation is performed, and previous moving averages are not considered. ARIMA is used for single-variable time series forecasting, requiring a different latitude and longitude training step for vehicles [34]. Therefore, the model is trained for each vehicle and its respective geographic coordinates [31].

3.5. Computational complexity

The PREDATOR's time complexity is analyzed as follows. As shown in Algorithm 1, the time complexity is mainly dominated by the ARIMA method runtime, which is estimated as $\mathcal{O}(m)$ [35], where m is the number of observations in the input data. In summary, the time complexity of ARIMA depends on the specific method used to fit the ARIMA model to the input data. The most common method for fitting an ARIMA model is Maximum Likelihood Estimation (MLE), which is computationally efficient and typically considered to have polynomial time complexity. In our evaluation, we used the library `statsmodels.tsa.arima.ARIMA` from the Python language, which uses the MLE to estimate the parameters.

Additionally, with the verification of each vehicle in the RSU's coverage, we have a time complexity of $\mathcal{O}(n)$, where n is the number of vehicles within that RSU ($|N_k|$). Also,

the predicted positions of each vehicle are also checked, resulting in a time complexity of $\mathcal{O}(pred)$. However, *pred* takes a value equal to the prediction window size T, which is fixed and makes it a constant. The search for the maximum value of S performed in line 3 also employs a time complexity of $\mathcal{O}(n)$. Assignment and comparison operations do not directly influence the time complexity calculation. Therefore, it can be noted that PREDATOR operates with a time complexity of $\mathcal{O}(n \times m) + \mathcal{O}(n)$, which can be reduced to $\mathcal{O}(n \times m)$. In summary, PREDATOR is a mechanism that operates in polynomial time and is efficient enough to operate in practical scenarios.

4. Performance Evaluation

This section describes the methodology and metrics used to evaluate PREDATOR performance in a VEC environment. First, we show the simulation environment, including implementation, scenario parameters, and evaluation metrics. Second, we discuss the obtained results.

4.1. Experimental settings

The simulation platform to evaluate the performance of the designed mechanism is composed of the Simulator of Urban Mobility (SUMO) 1.11.0, the network simulator OMNeT++ 5.6.1, and the vehicular networking framework Veins 5.2 [36], which implements the IEEE 802.11p protocol stack for V2X communication and signal attenuation.

We consider two mobility traces to establish vehicular evaluation scenarios. The first trace considers a Manhattan Grid (Grid) scenario [37] with 1 km^2 area, as shown in Figure 3(a). In this case, the traffic behavior is configured to use the Krauss car-following model for its accuracy and simplicity [38]. Second, we consider the realistic mobility trace of the Luxembourg city (LuST) [23, 13], as shown in Figure 3(b). We selected a 2 km^2 area with 5 intersections connecting the city center of Luxembourg city to a freeway. The simulations in this scenario start at 08 am (28 800 s) because it represents one of the times with high vehicular traffic. Also, we conducted 33 and 5 simulations with different randomly generated seeds for the Grid and LuST scenarios, respectively. All results show the values with a confidence interval of 95 %.

In both scenarios, RSUs have a higher communication range than vehicles. For RSUs, we consider the transmission power equal 6.1 mW. Together with the *Two-Ray ground* propagation model, these parameters provide a communication range of 250 m. Also, we use the frequency band of 5.89 GHz, a bandwidth of 10 MHz, and a bit rate of 6 Mbit/s at the MAC layer [39]. For vehicles, we only change the transmission power to 2.2 mW to give a communication range equal 150 m. The beacon frequency was 1 Hz [15]. Simulation time was 200 s for Grid and 500 s for the LuST scenario. After the simulation gets stable (*i.e.*, after a warm-up period defined as 100 s), the VCloud formation process starts. We setup 5 RSUs at the main intersections in all scenarios, as indicated by red points in Figures 3(a) and 3(b).

We consider an ARIMA(2,2,1) configuration in these scenarios. This means that we consider two past values, the series is differenced twice to achieve stationarity, and there is one moving average term. We utilize a GridSearch estimator to find the optimal parameters for the model [40]. At each VCloud formation interval, set at 5 s, we save the past mobility data to use as input for future predictions. The 5-second interval is acceptable, as it allows analyzing changes in the network topology without drastically compromising the prediction errors imposed by more extended intervals [41, 42]. The prediction model is implemented in Python 3.8 and connected to Veins using the os.System() interface in the C++ language. We set $\alpha = 0.3$ and $\beta = 0.7$, meaning that the dwell time in RSU coverage significantly impacts the relative distance between the vehicle and RSU. All relevant simulation parameters were considered based on the state-of-the-art and are listed in Table 2.

The main goal of our simulation-based evaluations is to assess the performance of PREDATOR compared to other state-of-the-art approaches, namely DEGREE [26, 27], CENTROID [9], and SPATIAL [6, 28]. Additionally, we compare the performance of PREDATOR with its previous version, called NEMESIS [22]. We include an approach that utilizes actual knowledge about the vehicles' mobility, referred to as OPTIMAL. In line with this, we evaluate the proposed mechanism using seven metrics for performance assessment, which are categorized into four perspectives or assessments. The details of these assessments are described below.

1. Mobility prediction assessment:
   - *Root Mean Square Error (RMSE)* quantifies the difference between real data and predicted data. This metric is widely used to measure the performance of predictors.
   - *Prediction time* represents the time required for the model to return the result. Considers training, testing, and prediction time. In this case, we used an Intel(R) Xeon(R) CPU X5650 (24 × 2.67 GHz) with Linux architecture x86-64.
2. VCloud formation assessment:
   - *Lifetime* is the accounting of how many time units the VClouds lasted. A high VCloud lifetime value means that the leader selected for this VCloudHead is stable, and the VCloudHead has not been changed frequently.
   - *Leader changes* are the number of times the vehicle is no longer VCloudHead in the VCloud. The high number of leader changes implies that vehicles selected as VCloudHead were not the most stable.
3. Scalability assessment:

(a) Manhattan Grid scenario

(b) Luxembourg (LuST) scenario

Figure 3: Simulation scenarios considered.

- *Sent packets* shows the total number of transmitted messages by the vehicles in the network. This result must be interpreted with other metrics. For example, the approaches must send fewer packets in the network, maintaining their high performance [15].
- *Packet collision* shows the total number of packets lost during message transmission. That occurs due to the busy communication channel and bit errors in received packets.

4. Scheduling assessment:
   - *Scheduling success* shows the total number of tasks that were successfully serviced. This evaluation is essential because it shows the impact of the VClouds' stability in using the aggregated computing resources.

Table 2: Simulation parameters.

Parameter	**Value**
Channel	5.89 GHz
Bandwidth	10 MHz
Data rate	6 Mbit/s
Transmission power (RSU)	6.1 mW
Communication range (RSU)	250 m
Transmission power (Vehicle)	2.2 mW
Communication range (Vehicle)	150 m
Beacon transmission rate	1 Hz
VCloud formation interval	5 s
Number of RSUs	5
α, β	0.3, 0.7
ARIMA (p, d, q)	(2,2,1)
Vehicle speed limits (Grid)	15 m/s
Simulation area (Grid)	1 km^2
Simulation time (Grid)	200 s
Vehicle speed limits (LuST)	From the LuST [23]
Simulation area (LuST)	2 km^2
Simulation time (LuST)	500 s

4.2. Simulations results

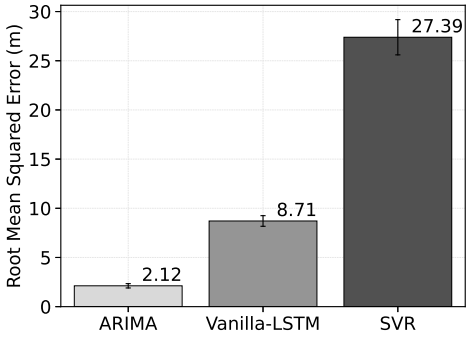
This subsection presents and discusses the simulation results, separated into four perspectives, namely: mobility prediction, VCloud metrics, scalability metrics, and task scheduling metrics.

4.2.1. Mobility prediction Assessment Perspective

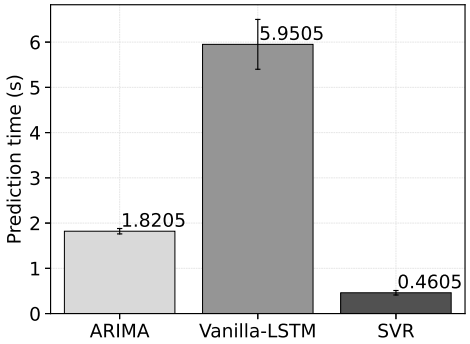
Figure 4(a) presents the RMSE results obtained with the considered models, namely ARIMA, Unmodified Long Short Term Memory (Vanilla-LSTM), and Support Vector Regression (SVR), in the prediction data. This assessment calculates the average RMSE among all vehicles in the scenario. However, please note that Figure 4(a) exemplifies the RMSE obtained for only one vehicle. The figure shows that the predictions provided by ARIMA achieve an average RMSE of approximately 2.12. In comparison, the predictions provided by Vanilla-LSTM and SVR experience more significant degradations, with RMSE values of 8.71 and 27.39, respectively. ARIMA performs better than other models when a smaller amount of past data is considered. This is because applying moving averages and differentiation becomes more straightforward and faster with a smaller dataset. Additionally, ARIMA utilizes predicted data to adjust subsequent predictions, serving as an error correction mechanism.

Figure 4(b) presents the prediction time results for the considered approaches. In this evaluation, SVR exhibits the shortest turnaround time for the prediction results, followed by ARIMA and Vanilla-LSTM. However, it is essential to compare this result with the RMSE results presented in Figure 4(a). Finding a trade-off between the two results is crucial. Based on the results, the ARIMA model demonstrates more robust performance in our mobility scenarios, achieving the best RMSE with a reasonable prediction time of under 2 s. Therefore, we utilized the ARIMA model to assist our VCloud formation process, setting the time window T to 5 s. Since it is an external process, the prediction time does not impact the dynamics

of the simulation. However, we consider running the predictions 2 s before each interval to account for the model's processing time. Afterward, we adjust the predicted value with the actual value received from the RSU.



(a) Root Mean Square Error (RMSE)



(b) Processing time

Figure 4: Prediction results on a single vehicle's data.

4.2.2. VCloud formation Assessment Perspective

Figure 6 shows the results obtained for VCloud metrics, such as Lifetime and Leader changes, considering the two different mobility traces. First, Figures 6(a) and 6(c) show the VClouds' lifetime results for both scenarios. It can be noted that VClouds formed with PREDATOR have longer lifetimes in all observed scenarios. That is, selecting the vehicle that will spend the most time in RSU coverage makes the VCloud exist during that vehicle's travel time, which is different from the other compared approaches. Also, the number of leader changes is decreased when PREDATOR is used in all scenarios. This result is expected because if the vehicle with the longest time in RSU coverage is selected, the leader change will only occur when that vehicle leaves the intersection covered by the RSU.

In addition, we compare all approaches with a solution that has actual knowledge of vehicle mobility, called OPTIMAL. The decision mechanism used in this approach is the same as the one used by PREDATOR. The difference is that the optimal strategy does not have prediction errors in vehicle mobility information. Thus, we can consider the OPTIMAL approach as the baseline. In the Grid scenario, PREDATOR maintains an average lifetime of 18.69 s compared to NEMESIS's 14.95 s and CENTROID's 13.57 s. Yet, the observed optimal lifetime is 24.3 s. Similarly, in the LuST scenario, PREDATOR maintains a lifetime of 29.33 s compared to NEMESIS's 24.56 s and the SPATIAL's 19.99 s. The OPTIMAL approach achieved 36.66 s of lifetime in this scenario. In this evaluation, followed by the result achieved by NEMESIS, SPATIAL is found to be superior to DEGREE and CENTROID approaches. This result can be justified by the vehicular dynamics of the realistic scenario, which exhibit fewer drastic changes in vehicle positioning, allowing the spatial information to remain valid for a longer duration. In summary, PREDATOR shows an overall improvement of 18.14 %, 40.40 %, 49.48 %, and 60.59 % in the VCloud lifetime metric over NEMESIS, SPATIAL, CENTROID, and DEGREE, respectively.

Complementarily to the experiment shown in Figure 6(a) and to reinforce the impact of vehicular mobility in the context of VCloud formation, we conducted an experiment where the vehicle speed was varied at 0 m/s (no mobility), 15 m/s (low mobility), and 25 m/s (high mobility) in the Manhattan Grid scenario. As the mobility of the Grid scenario is synthetic, changes in the vehicle speed are entirely possible. Therefore, Figure 5 displays the VClouds' lifetime using all VCloud formation approaches. Naturally, the performance of all approaches degrades as the vehicle speed increases since selecting vehicles under these conditions becomes much more challenging due to high topological changes. However, it can be observed that PREDATOR remains superior to the other approaches in all speed variations, except for the OPTIMAL approach, which contains vehicle mobility information without prediction errors. Furthermore, when mobility is static (vehicle speed equal to 0 m/s), all approaches perform similarly. This is because the initial vehicle chosen as the leader will retain its leadership throughout the simulation time. In other words, since mobility remains static, no other vehicle with better rates (degree centrality, closer to the centroid, greater spatial density, etc.) will be chosen as the new leader.

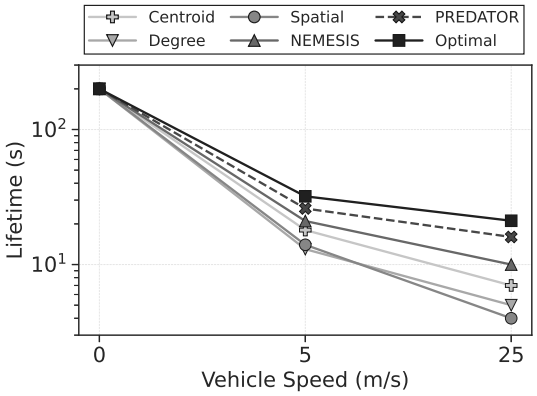
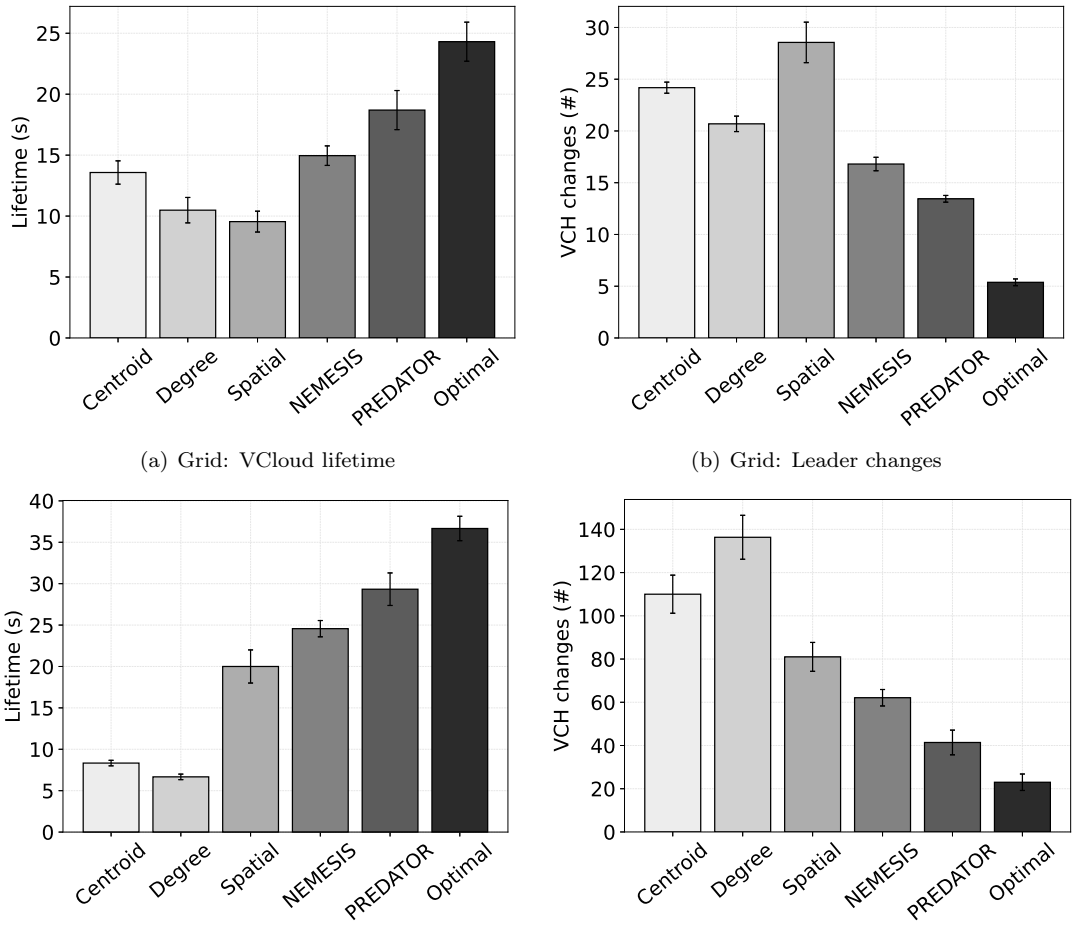


Figure 5: An example of how vehicular mobility impacts all VCloud formation solutions

In addition, Figures 6(b) and 6(d) show the general



(a) Grid: VCloud lifetime

(b) Grid: Leader changes

(c) LuST: VCloud lifetime

(d) LuST: Leader changes

Figure 6: VCloud formation results considering different mobility traces.

stability of the created VClouds. It can be observed that PREDATOR spends more time in the RSU coverage, which implies greater stability in VCloud management. SPATIAL performs the worst in the Grid scenario as it selects vehicles based on spatial information, and the synthetic mobility in this scenario can have drastic directional and positional changes. On the other hand, in the LuST scenario, the worst-performing approach is DEGREE. This result indicates a high rate of change in vehicles with the highest degree coefficient indices (largest neighborhood). Given its positioning changes, the neighborhood density of this vehicle also changes with more frequency. That is, specifically in this area considered, the flow of vehicles is high, and the vehicular social contacts accompany this dynamic. In the Grid scenario, PREDATOR experiences an average of 13 leader changes compared to NEMESIS's 16 changes and DEGREE's 21 changes. Similarly, in the LuST scenario, PREDATOR has 41 leader changes compared to NEMESIS's 62 changes and SPATIAL's 81 changes. The optimal approach has 6 leader changes in the Grid scenario and 23 changes in the LuST scenario. Overall, PREDATOR employs improvement in this metric by 26.64 %, 50.91 %, 52.31 %, and 53.38 % over NEMESIS, SPATIAL, CENTROID, and DEGREE, respectively.

4.2.3. Scalability Assessment Perspective

Regarding scalability metrics, Figures 7(a) and 7(c) present the results regarding the number of packets sent on the network by the evaluated approaches. In this evaluation, only the messages transmitted among vehicles are considered, as the number of sent packets by the RSU is the same in both approaches, given that the VCloud formation intervals are identical. However, the number of sent packets is related to leader changes. PREDATOR only sends a leader message if the VCloudHead changes from one interval to another (Algorithm 2). This strategy significantly reduces the number of sent packets. Thus, we observe that PREDATOR transmits fewer packets on the network than the other approaches. This result can be attributed to the findings shown in Figures 6(b) and 6(d), as a smaller number of VCloudHead changes leads to fewer warning messages being sent on the network. The optimal approach employs approximately 2100 sent packets in the Grid scenario and around 5640 sent packets in the LuST scenario. In summary, PREDATOR demonstrates improvements of 28.75 %, 53.96 %, 55.38 %, and 55.94 %

over NEMESIS, DEGREE, SPATIAL, and CENTROID, respectively.

Figures 7(b) and 7(d) show the number of packet collisions. It is important to note that all approaches utilize a Flooding algorithm to disseminate warning messages on the network [15]. We chose this approach due to its simplicity and generally good performance in terms of delivery rate on the network. However, maintaining stable leader selection already leads to significant improvements in network performance metrics. PREDATOR exhibits the lowest number of packet collisions on the network since it transmits fewer warning messages. The OPTIMAL approach results in approximately 618 collided packets in the Grid scenario and around 2000 in the LuST scenario. Therefore, PREDATOR demonstrates improvements of 30.58 %, 45.86 %, 47.48 %, and 50.28 % over NEMESIS, DEGREE, SPATIAL, and CENTROID, respectively.

4.2.4. Scheduling Assessment Perspective

Considering the created VClouds' stability, it was necessary to simulate a practical application of this scenario. The application concerns the use of vehicular computational resources that were aggregated in the VCloud Formation process. This use is called task scheduling and must consider the processing time constraints of each task. As there is no order restriction for executing the tasks that arrive at the system, abstractions of Bag-of-Tasks applications were considered for this evaluation.

In this sense, the scheduling evaluation was modeled as follows: there is a set of tasks where each task has a processing time necessary for its completion. When a task is scheduled, its scheduling success is only computed when the total processing time is reached. If the VCloudHead changes, the task is automatically canceled, and its interruption is computed. The controller present on the network is responsible for selecting which VClouds will process a given set of tasks. On the other hand, the VCloudHeads are responsible for managing the processing of these tasks between their VCloudMembers.

Algorithm 4 presents an abstraction of what happens in the VCloudHead vehicle when it receives the schedule message from the controller. VCloudHead starts a timer to control when the task completes its execution (Line 1). Therefore, VCloudHead adds a time unit to the timer (Line 3). As the VCloud formation process is independent of the scheduling process, verifying if the current vehicle is still VCloudHead of this VCloud at each instant is necessary. If the vehicle is VCloudHead, it checks if the control timer is equal to the task's processing time and, if so, the task has been completely executed in the VCloud (Line 5). VCloudHead informs the controller about the execution of the task and computes the scheduling success. However, if the vehicle is no longer VCloudHead during the task execution process, it is checked if its control timer is less than the task processing time and, if so, the task was interrupted (Line 7). The vehicle informs the controller and accounts for the schedule failure.

Algorithm 4: Task scheduling control in VCloudHead

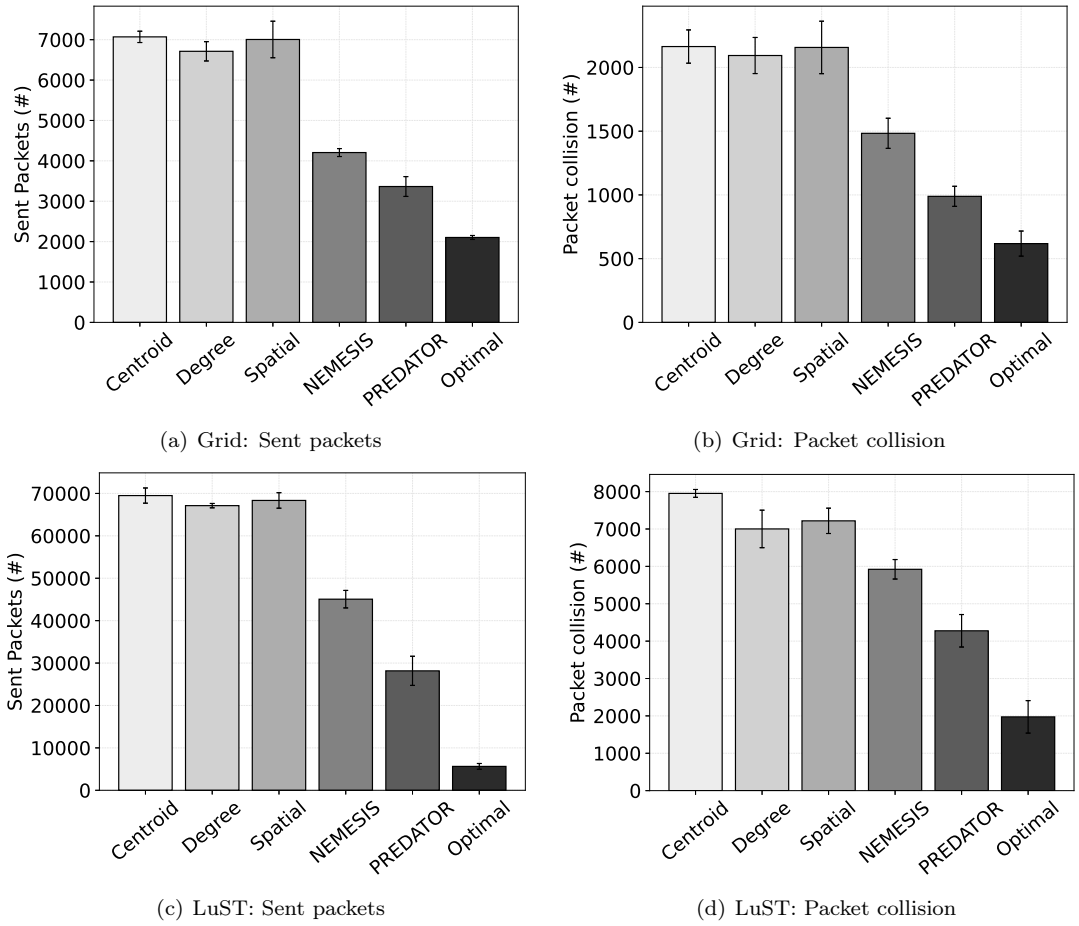
```
   Input: schedule message msg with VCloud
          information, such as computational
          resources and number of VCloudMembers
   ▷ VCloudHead receives scheduling message
   ▷ VCloudHead starts a control timer
1  scheduleTimer ← 0
2  foreach time slot do
3      scheduleTimer ← scheduleTimer + 1
4      if status = VCloudHead then
5          if scheduleTimer = msg.time then
               ▷ Complete task execution
               ▷ Informs the controller through RSU
               ▷ Scheduling success
6      else
7          if scheduleTimer ≠ msg.time then
               ▷ Stop task execution
               ▷ Informs the controller through RSU
8          scheduleTimer ← 0
```

The simulation settings were the same used in the evaluation of VCloud formation presented in the previous section. However, to assess the efficiency of VClouds, we consider applications with processing times ranging from 5 s to 10 s. As the objective is to evaluate the efficiency of VClouds about stability, other metrics besides the time of tasks and VClouds were not considered. The number of tasks was varied by 10, 15, 20, 25, and 30.

Figure 8 presents the results for task scheduling assessment. We can see that the results obtained corroborate the results obtained in evaluating the VClouds formation about the average lifetime of the VClouds. That is, the longer the lifetime of the VCloud, the greater the chance of it serving a greater number of tasks in the system. Not only that, but VCloud stability is crucial to increasing the percentage of successfully processed tasks. As expected, as the number of tasks grows in the system, the chance that tasks fail to be scheduled increases. As we can see, the number of tasks scheduled in the LuST scenario is higher than in the Grid scenario due to the greater stability (lifetime and leader changes) of the VClouds created in this scenario. In all observed scenarios (Grid and LuST), PREDATOR manages to stay superior in 10.54 %, 25.87 %, 27.68 %, and 34.65 % more tasks than the NEMESIS, CENTROID, SPATIAL, and DEGREE, respectively. Also, PREDATOR is 89.13 % and 86.87 % closer to the optimal approach in the Grid and LuST scenarios, respectively.

5. Conclusion

The Vehicular Edge Computing (VEC) paradigm emerged and enabled vehicles to actively act in the consumption



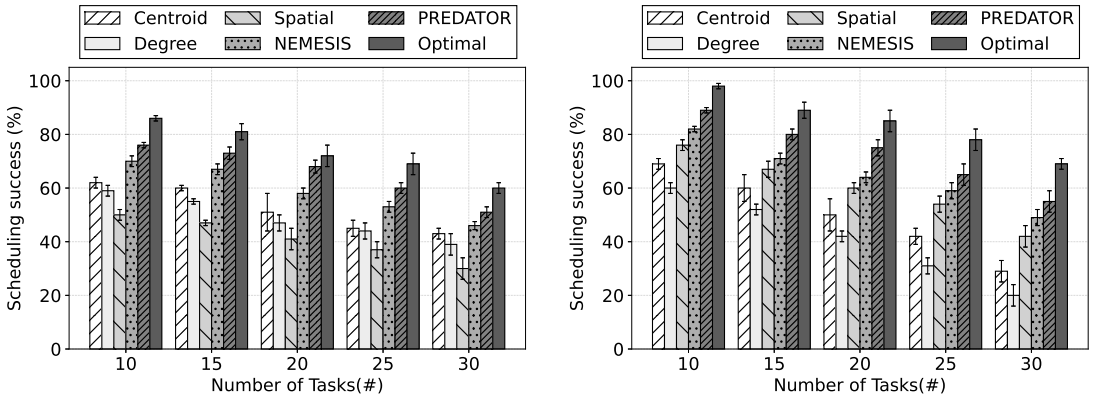
(a) Grid: Sent packets

(b) Grid: Packet collision

(c) LuST: Sent packets

(d) LuST: Packet collision

Figure 7: Networks metrics considering different mobility traces.



(a) Task scheduling in the Grid scenario

(b) Task scheduling in the LuST scenario

Figure 8: Task scheduling results considering computational tasks with different requirements.

and supply of computational power to the Vehicular Ad-Hoc Network (VANET). However, due to the dynamic nature of vehicular mobility, proposing mechanisms that efficiently aggregate vehicular resources is not a trivial task. This aggregation of vehicle resources is referred to as Vehicular Cloud (VCloud) formation. In this context, we introduced a mobility-aware VCloud formation mechanism called PREDATOR, which selects the most stable nodes in the network to perform cloud leadership activities. By leveraging this mechanism, VEC applications can be allocated to VClouds with the assurance that their processes will be successfully completed within the required time frame.

As demonstrated in the experiments, vehicular mobility directly impacts the efficiency of VCloud formation approaches that do not consider it in their decision-making

process. For instance, approaches relying solely on spatial information exhibit the poorest performance in all analyses. In other words, selecting a vehicle to lead a VC without considering its current or future mobility does not ensure the necessary stability for applications utilizing the computational power of VClouds, with no estimate of the resource availability duration. On the other hand, PREDATOR outperformed all evaluations precisely because it considers future mobility when deciding VCloud leadership, thereby enabling a future estimation of the duration of these VClouds. The results demonstrate that PREDATOR can increase the average lifetime of VClouds, reduce the number of leader changes within these clouds, and minimize network message exchanges compared to other approaches in the literature, resulting in fewer packet collisions. Furthermore, PREDATOR provides enhanced stability to VEC applications with stringent deadline constraints.

In future work, we intend to explore additional techniques for mobility prediction and incorporate other metrics to evaluate task scheduling. Additionally, we aim to conduct an exploratory assessment of factors such as route information, which could improve dwell time estimates within VClouds. We also plan to investigate changes in the weights assigned to the relative distance between the vehicle and Roadside Unit (RSU) α and dwell time β to further enhance the system's performance.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the grants #2018/16703-4 and #2021/13780-0 of the São Paulo Research Foundation (FAPESP). Wellington Lobato would like to acknowledge the financial support granted by FAPESP, process #2019/19105-3, in his research. Eduardo Cerqueira would like to acknowledge the financial support granted by CNPq.

Data availability

Data will be made available on request.

References

[1] S. Feng, X. Yan, H. Sun, Y. Feng, H. X. Liu, Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment, Nature Communications 12 (1) (Feb. 2021). doi:10.1038/s41467-021-21007-8.

[2] L. Le Mero, D. Yi, M. Dianati, A. Mouzakitis, A Survey on Imitation Learning Techniques for End-to-End Autonomous Vehicles, IEEE Transactions on Intelligent Transportation Systems (2022). doi:10.1109/TITS.2022.3144867.

[3] T. Yoshizawa, D. Singelée, J. T. Mühlberg, S. Delbruel, A. Taherkordi, D. Hughes, B. Preneel, A survey of security and privacy issues in v2x communication systems, ACM Computing Surveys (CSUR) (Aug 2022). doi:10.1145/3558052.

[4] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, E. Madeira, Vehicular Edge Computing: Architecture, Resource Management, Security, and Challenges, ACM Computing Surveys (CSUR) 55 (1) (Jan. 2023). doi:10.1145/3485129.

[5] G. S. Pannu, S. Dunkel, S. Ucar, T. Higuchi, O. Altintas, F. Dressler, Improving Data Consistency in Vehicular Micro Clouds, in: 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2022, pp. 489–490. doi:10.1109/CCNC49033.2022.9700653.

[6] J. B. D. da Costa, R. I. Meneguette, D. Rosário, L. A. Villas, Combinatorial Optimization-based Task Allocation Mechanism for Vehicular Clouds, in: IEEE 91st Vehicular Technology Conference (VTC Spring), IEEE, 2020, pp. 1–5. doi:10.1109/VTC2020-Spring48590.2020.9128834.

[7] G. S. Pannu, S. Ucar, T. Higuchi, O. Altintas, F. Dressler, Vehicular Virtual Edge Computing using Heterogeneous V2V and V2C Communication, in: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2022, pp. 1–2. doi:10.1109/infocomwkshps54753.2022.9798362.

[8] H. Choi, Y. Nam, Y. Shin, E. Lee, The partial cloud member replacement for reconstructing vehicular clouds in VANETs: Reactive and proactive schemes, Ad Hoc Networks 136 (2022) 102959. doi:10.1016/j.adhoc.2022.102959.

[9] F. Hagenauer, T. Higuchi, O. Altintas, F. Dressler, Efficient data handling in vehicular micro clouds, Ad Hoc Networks 91 (2019) 101871. doi:10.1016/j.adhoc.2019.101871.

[10] A. Boukerche, N. Aljeri, Design Guidelines for Topology Management in Software-Defined Vehicular Networks, IEEE Network 35 (2) (2021) 120–126. doi:10.1109/MNET.011.2000369.

[11] M. S. Bute, P. Fan, G. Liu, F. Abbas, Z. Ding, A cluster-based cooperative computation offloading scheme for C-V2X networks, Ad Hoc Networks 132 (2022) 102862. doi:10.1016/j.adhoc.2022.102862.

[12] A. Boukerche, V. Soto, Computation Offloading and Retrieval for Vehicular Edge Computing: Algorithms, Models, and Classification, ACM Computing Surveys (CSUR) 53 (4) (Jul. 2021). doi:10.1145/3392064.

[13] G. S. Pannu, S. Ucar, T. Higuchi, O. Altintas, F. Dressler, Dwell time estimation at intersections for improved vehicular micro cloud operations, Ad Hoc Networks 122 (2021) 102606. doi:10.1016/j.adhoc.2021.102606.

[14] X. Li, A. Garcia-Saavedra, X. Costa-Perez, C. J. Bernardos, C. Guimarães, K. Antevski, J. Mangues-Bafalluy, J. Baranda, E. Zeydan, D. Corujo, P. Iovanna, G. Landi, J. Alonso, P. Paixao, H. Martins, M. Lorenzo, J. Ordonez-Lucena, D. R. Lopez, 5Growth: An End-to-End Service Platform for Automated Deployment and Management of Vertical Services over 5G Networks, IEEE Communications Magazine 59 (3) (2021) 84–90. doi:10.1109/MCOM.001.2000730.

[15] J. B. D. da Costa, A. M. de Souza, D. Rosário, E. Cerqueira, L. A. Villas, Efficient data dissemination protocol based on complex networks' metrics for urban vehicular networks, Journal of Internet Services and Applications 10 (1) (Aug. 2019). doi:10.1186/s13174-019-0114-y.

[16] Z. Rejiba, X. Masip-Bruin, E. Marín-Tordera, A Survey on Mobility-Induced Service Migration in the Fog, Edge, and Related Computing Paradigms, ACM Computing Surveys (CSUR) 52 (5) (Sep. 2020). doi:10.1145/3326540.

[17] W. Long, T. Li, Z. Xiao, D. Wang, R. Zhang, A. C. Regan, H. Chen, Y. Zhu, Location Prediction for Individual Vehicles via Exploiting Travel Regularity and Preference, IEEE Transactions on Vehicular Technology 71 (5) (2022) 4718–4732. doi:10.1109/TVT.2022.3151762.

[18] N. Magaia, P. Ferreira, P. R. Pereira, K. Muhammad, J. Del Ser, V. H. C. de Albuquerque, Group'n Route: An Edge Learning-Based Clustering and Efficient Routing Scheme

Leveraging Social Strength for the Internet of Vehicles, IEEE Transactions on Intelligent Transportation Systems (2022). doi:10.1109/TITS.2022.3171978.

[19] H. Zhao, J. Tang, B. Adebisi, T. Ohtsuki, G. Gui, H. Zhu, An Adaptive Vehicle Clustering Algorithm Based on Power Minimization in Vehicular Ad-Hoc Networks, IEEE Transactions on Vehicular Technology 71 (3) (2022) 2939–2948. doi:10.1109/TVT.2021.3140085.

[20] X. Wu, S. Zhao, R. Zhang, L. Yang, Mobility Prediction-Based Joint Task Assignment and Resource Allocation in Vehicular Fog Computing, in: IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2020, pp. 1–6. doi:10.1109/WCNC45663.2020.9120524.

[21] J. Liu, M. Ahmed, M. A. Mirza, W. U. Khan, D. Xu, J. Li, A. Aziz, Z. Han, Rl/drl meets vehicular task offloading using edge and vehicular cloudlet: A survey, IEEE Internet of Things Journal 9 (11) (2022) 8315–8338. doi:10.1109/JIOT.2022.3155667.

[22] J. B. D. da Costa, W. V. Lobato J., A. M. de Souza, E. Cerqueira, D. Rosário, L. A. Villas, NEMESIS: Mecanismo para Formação de Nuvens Veiculares Baseado em Previsão de Mobilidade, in: XL Brazilian Symposium on Computer Networks and Distributed Systems (SBRC), Sociedade Brasileira de Computação, 2022, pp. 280–293. doi:10.5753/sbrc.2022.222309.

[23] L. Codecá, R. Frank, S. Faye, T. Engel, Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation, IEEE Intelligent Transportation Systems Magazine 9 (2) (2017) 52–63. doi:10.1109/MITS.2017.2666585.

[24] C. Cooper, D. Franklin, M. Ros, F. Safaei, M. Abolhasan, A Comparative Survey of VANET Clustering Techniques, IEEE Communications Surveys & Tutorials 19 (1) (2017) 657–681. doi:10.1109/COMST.2016.2611524.

[25] M. Ayyub, A. Oracevic, R. Hussain, A. A. Khan, Z. Zhang, A comprehensive survey on clustering in vehicular networks: Current solutions and future challenges, Ad Hoc Networks 124 (2022) 102729. doi:10.1016/j.adhoc.2021.102729.

[26] P. Zhao, L. Feng, P. Yu, W. Li, X. Qiu, A Social-Aware Resource Allocation for 5G Device-to-Device Multicast Communication, IEEE Access 5 (2017) 15717–15730. doi:10.1109/ACCESS.2017.2731805.

[27] S. Kamakshi, V. S. S. Sriram, Modularity based mobility aware community detection algorithm for broadcast storm mitigation in VANETs, Ad Hoc Networks 104 (2020) 102161. doi:10.1016/j.adhoc.2020.102161.

[28] M. L. M. Peixoto, A. H. Maia, E. Mota, E. Rangel, D. G. Costa, D. Turgut, L. A. Villas, A traffic data clustering framework based on fog computing for VANETs, Vehicular Communications 31 (2021) 100370. doi:10.1016/j.vehcom.2021.100370.

[29] F. Abbasi, M. Zarei, A. M. Rahmani, FWDP: A fuzzy logic-based vehicle weighting model for data prioritization in vehicular ad hoc networks, Vehicular Communications 33 (2022) 100413. doi:10.1016/j.vehcom.2021.100413.

[30] W. Wang, F. Xia, H. Nie, Z. Chen, Z. Gong, X. Kong, W. Wei, Vehicle Trajectory Clustering Based on Dynamic Representation Learning of Internet of Vehicles, IEEE Transactions on Intelligent Transportation Systems 22 (6) (2021) 3567–3576. doi:10.1109/TITS.2020.2995856.

[31] A. Costa, L. Pacheco, D. Rosário, L. Villas, A. A. F. Loureiro, S. Sargento, E. Cerqueira, Skipping-based Handover Algorithm for Video Distribution Over Ultra-Dense VANET, Computer Networks 176 (2020) 107252. doi:10.1016/j.comnet.2020.107252.

[32] F. Tang, B. Mao, N. Kato, G. Gui, Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges, IEEE Communications Surveys & Tutorials 23 (3) (2021) 2027–2057. doi:10.1109/COMST.2021.3089688.

[33] L. N. Balico, A. A. F. Loureiro, E. F. Nakamura, R. S. Barreto, R. W. Pazzi, H. A. B. F. Oliveira, Localization Prediction in Vehicular Ad Hoc Networks, IEEE Communications Surveys & Tutorials 20 (4) (2018) 2784–2803. doi:10.1109/COMST.2018.2841901.

[34] G. Sun, L. Song, H. Yu, V. Chang, X. Du, M. Guizani, V2V Routing in a VANET Based on the Autoregressive Integrated Moving Average Model, IEEE Transactions on Vehicular Technology 68 (1) (2019) 908–922. doi:10.1109/TVT.2018.2884525.

[35] A. Gupta, H. P. Gupta, B. Biswas, T. Dutta, A fault-tolerant early classification approach for human activities using multivariate time series, IEEE Transactions on Mobile Computing 20 (5) (2020) 1747–1760. doi:10.1109/TMC.2020.2973616.

[36] C. Sommer, R. German, F. Dressler, Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis, IEEE Transactions on Mobile Computing (TMC) 10 (1) (2011) 3–15. doi:10.1109/TMC.2010.133.

[37] O. Alzamzami, I. Mahgoub, Link utility aware geographic routing for urban vanets using two-hop neighbor information, Ad Hoc Networks 106 (2020) 102213. doi:https://doi.org/10.1016/j.adhoc.2020.102213.

[38] S. Krauß, P. Wagner, C. Gawron, Metastable states in a microscopic model of traffic flow, Physical Review E 55 (5) (1997) 5597.

[39] Y. A. Debalki, J. Hou, H. Ullah, B. Y. Adane, Multi-hop data dissemination using a multi-metric contention-based broadcast suppression strategy in vanets, Ad Hoc Networks 140 (2023) 103070. doi:https://doi.org/10.1016/j.adhoc.2022.103070.

[40] E. Ndiaye, T. Le, O. Fercoq, J. Salmon, I. Takeuchi, Safe grid search with optimal complexity, in: Proceedings of the 36th International Conference on Machine Learning, PMLR, 2019, pp. 4771–4780.

[41] N. Dasanayaka, Y. Feng, Analysis of vehicle location prediction errors for safety applications in cooperative-intelligent transportation systems, IEEE Transactions on Intelligent Transportation Systems 23 (9) (2022) 15512–15521.

[42] Y. Liu, L. Huo, J. Wu, A. K. Bashir, Swarm learning-based dynamic optimal management for traffic congestion in 6g-driven intelligent transportation system, IEEE Transactions on Intelligent Transportation Systems (2023).