Universidade Estadual de Campinas
Instituto de Computação

Joahannes Bruno Dias da Costa

Mobility-aware Resource Management for Vehicular Edge Computing

Gerenciamento de Recursos Ciente de Mobilidade para Computação de Borda Veicular

CAMPINAS
2023

Joahannes Bruno Dias da Costa

Mobility-aware Resource Management for Vehicular Edge Computing

Gerenciamento de Recursos Ciente de Mobilidade para Computação de Borda Veicular

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Leandro Aparecido Villas
Co-supervisor/Coorientador: Prof. Dr. Denis Lima do Rosário

Este exemplar corresponde à versão final da Tese defendida por Joahannes Bruno Dias da Costa e orientada pelo Prof. Dr. Leandro Aparecido Villas.

CAMPINAS
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

C823m

Costa, Joahannes Bruno Dias da, 1994-
Mobility-aware resource management for vehicular edge computing / Joahannes Bruno Dias da Costa. – Campinas, SP : [s.n.], 2023.

Orientador: Leandro Aparecido Villas.
Coorientador: Denis Lima do Rosário.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Escalonamento de tarefas. 2. Computação de borda veicular. 3. Gerenciamento de recursos de informação. 4. Redes ad hoc veiculares (Redes de computadores). I. Villas, Leandro Aparecido, 1983-. II. Rosário, Denis Lima do. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações Complementares

Título em outro idioma: Gerenciamento de recursos ciente de mobilidade para computação de borda veicular
Palavras-chave em inglês:
Task scheduling
Vehicular edge computing
Information resources management
Vehicular ad hoc networks (Computer networks)
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Leandro Aparecido Villas [Orientador]
Fabrício Aguiar Silva
Lourenço Alves Pereira Júnior
Juliana Freitag Borin
Carlos Alberto Astudillo Trujillo
Data de defesa: 05-09-2023
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-9973-2479
- Currículo Lattes do autor: http://lattes.cnpq.br/4761632587625158

Universidade Estadual de Campinas
Instituto de Computação

Joahannes Bruno Dias da Costa

Mobility-aware Resource Management for Vehicular Edge Computing

Gerenciamento de Recursos Ciente de Mobilidade para Computação de Borda Veicular

Banca Examinadora:

- Prof. Dr. Leandro Aparecido Villas
  Instituto de Computação - UNICAMP
- Prof. Dr. Fabrício Aguiar Silva
  Universidade Federal de Viçosa - UFV
- Prof. Dr. Lourenço Alves Pereira Júnior
  Instituto Tecnológico de Aeronáutica - ITA
- Profa. Dra. Juliana Freitag Borin
  Instituto de Computação - UNICAMP
- Prof. Dr. Carlos Alberto Astudillo Trujillo
  Instituto de Computação - UNICAMP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 05 de setembro de 2023

To my family and friends.

Agradecimentos

Aos meus pais João e Lenita, pelo amor, paciência e por sempre vibrarem com minhas conquistas. E às minhas irmãs queridas Milla e Anne. Amo vocês.

À minha noiva Brenda, por todo amor, carinho, paciência e apoio em todos esses anos de doutorado. Obrigado por ser minha companheira, por me acalmar nos momentos tristes e festejar ao meu lado nos momentos de alegria. Você contribuiu para que essa jornada fosse mais leve. Obrigado, meu amor.

Ao meu orientador Prof. Leandro Villas, pela confiança, conversas, direcionamentos e discussões sempre produtivas e descontraídas. Ao meu coorientador Prof. Denis Rosário, também pela confiança e direcionamentos. Pude aprender muito com vocês durante todo esse período.

I thank Professor Christoph Sommer, who welcomed me to Dresden, Germany, in his laboratory. The 12 months I lived in Dresden were very important for my research and personal growth.

Aos colegas e amigos que fiz dentro e fora da universidade, obrigado pelos momentos sempre divertidos ao lado de vocês. O acolhimento desde os meus primeiros dias em Campinas foi essencial para eu me sentir confortável mesmo estando tão longe de casa. Muito obrigado.

Aos docentes e funcionários do Instituto de Computação da UNICAMP, pelo conhecimento transmitido e pela infraestrutura disponibilizada.

A Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) pelo apoio financeiro durante esta pesquisa de doutorado através do processo #2018/16703-4. Ainda, agradeço a FAPESP pela possibilidade de realização do doutorado sanduíche (processo #2021/13780-0) na *Technische Universität Dresden* - Alemanha, uma etapa fundamental para o desenvolvimento desta pesquisa.

Resumo

A indústria automobilística vem investindo continuamente na modernização dos veículos, aprimorando suas capacidades de comunicação e processamento de dados. Seguindo essa evolução, o paradigma de Computação de Borda Veicular (VEC) surge com a finalidade de prover serviços de computação em nuvem próximo aos usuários veiculares, utilizando os recursos computacionais dos próprios veículos. Nesse cenário, veículos e infraestruturas de comunicação podem cooperativamente atender serviços/aplicações veiculares, agregando seus recursos e disponibilizando-os por meio das Nuvens Veiculares (VCs). Para que essa disponibilização aconteça, os seguintes processos devem ser realizados: *(i)* Formação de VCs, que é o agrupamento dos veículos e seus recursos computacionais disponíveis; e *(ii)* Escalonamento de Tarefas, que tem como finalidade decidir em qual das VCs um determinado conjunto de tarefas será processado. Além disso, realizar balanceamento de carga entre as VCs é fundamental para aumentar a justiça na utilização dos recursos e tornar a distribuição de carga na rede mais homogênea. No entanto, a mobilidade é um dos principais desafios na proposição de soluções nesses cenários, uma vez que a mobilidade veicular provoca diversas mudanças na topologia da rede e conexões intermitentes.

Nesse contexto, esta tese apresenta um estudo de como se dá a formação das VCs e como as aplicações podem utilizar os recursos dessas nuvens de forma eficiente para processamento de dados e, com isso, auxiliar em tomadas de decisão que exigem baixa latência e tempo de processamento restrito. Além disso, esta tese propõe uma série de mecanismos para lidar com diferentes aspectos da mobilidade na borda da rede veicular. A primeira contribuição desta tese reside em uma solução ciente de mobilidade para estimar o tempo de permanência dos veículos em uma determinada região e, assim, mitigar os impactos da mobilidade na formação de VCs. Como segunda contribuição, foi proposto um mecanismo de escalonamento de tarefas que utiliza uma arquitetura de Rede Neural Recorrente (RNN) para estimar os recursos computacionais nas VCs e garantir que as demandas dos usuários sejam atendidas. Essa abordagem consegue aumentar o número de tarefas escalonadas, diminuir a latência geral do sistema e diminuir os custos monetários pela utilização dos recursos computacionais. A terceira contribuição se volta no aumento da justiça e balanceamento de carga na utilização dos recursos das VCs. As principais vantagens desse último mecanismo incluem: *(i)* um escalonamento de tarefas que maximiza o número de tarefas escalonadas e processadas com sucesso enquanto mantém um balanceamento de carga justo no uso de recursos computacionais e *(ii)* o uso de *multithreading* para resolução paralela de subproblemas de escalonamento, visando reduzir a latência do sistema sem comprometer o desempenho geral da solução.

As soluções propostas foram amplamente comparadas com outras soluções da literatura em diferentes métricas de avaliação de desempenho e considerando cenários de mobilidade realísticos. Os resultados mostram que as abordagens propostas são eficientes e escaláveis, nas quais podem ser boas alternativas para mitigar os desafios impostos pela dinamicidade da mobilidade veicular nos ambientes de VEC.

Abstract

The automobile industry has being continuously investing in the modernization of vehicles, improving their communication and data processing capacities. Following this evolution, the Vehicular Edge Computing (VEC) paradigm emerged to provide computing power and storage capability close to vehicular users. In this scenario, vehicles and communication infrastructures can cooperatively attend vehicular services/applications, aggregating their resources and making them available through Vehicular Clouds (VCs). For this availability to happen, the following processes must be carried out: *(i)* VC Formation, which is the grouping of vehicles and their available computational resources; and *(ii)* Task Scheduling, which aims to decide which of the VCs a given set of tasks will be processed. Also, carrying out load balancing between the VCs is essential to increase fairness in the use of resources and make the load distribution in the network more homogeneous. However, mobility is one of the main challenges in proposing solutions in these scenarios, since vehicular mobility causes several changes in the network topology and intermittent connections.

In this context, this thesis presents a study of how VCs are formed and how applications can use the resources of these clouds efficiently for data processing and, with that, help in decision making that requires low latency and restricted processing time. Furthermore, this thesis proposes a series of mechanisms to deal with different aspects of mobility at the edge of the vehicular network. The first contribution of this thesis lies in a mobility-aware solution to estimate the dwell-time of vehicles in a given region and thus mitigate the impacts of mobility on the VC formation process. Secondly, a task scheduling mechanism was proposed that uses a Recurrent Neural Network (RNN) architecture to estimate computational resources in VCs and ensure that user demands are met. This approach manages to increase the number of scheduled tasks, decrease the overall system latency, and reduce the monetary costs for using computational resources. The third contribution focuses on increasing fairness and load balancing in the use of VC's resources. The main advantages of the latter mechanism include: *(i)* a task scheduler that maximizes the number of tasks successfully scheduled and processed while maintaining fair load balancing in the use of computational resources and *(ii)* the use of *multithreading* for parallel solving of scheduling subproblems, aiming to reduce system latency without compromising the overall performance of the solution.

The proposed solutions were widely compared with other state-of-the-art solutions in different performance evaluation metrics and considering realistic mobility scenarios. The results show that the proposed approaches are efficient, scalable, and cost-effective, which can be good alternatives to mitigate the challenges imposed by the dynamics of vehicular mobility in the VEC environments.

List of Figures

1.1 Circulating fleet of vehicles in Brazil between the years 2010 and 2022. . . 18

2.1 Different types of communication in Vehicular Ad-Hoc Networks (VANETs). 25
2.2 A traditional VEC system architecture, presenting its main components, such as vehicles, Base Stations (BSs), Remote Server (RS), Vehicular Clouds (VCs), and Vehicular Edge Computing (VEC) Controllers. 29

4.1 System architecture (VCH: Vehicular Cloud Head (VCloudHead); VCM: Vehicular Cloud Member (VCloudMember). 44
4.2 PREDATOR's microscopic vision. 47
4.3 Simulation scenarios considered. 52
4.4 Prediction results on a single vehicle's data. 55
4.5 An example of how vehicular mobility impacts all VC formation solutions . 56
4.6 VC formation results considering different mobility traces. 57
4.7 Networks metrics considering different mobility traces. 58
4.8 Task scheduling results considering computational tasks with different requirements. 60

5.1 The architecture employed by MARINA, presenting its main components; in particular, Vehicular Clouds (VCs) and Vehicular Edge Computing (VEC) controllers. 64
5.2 Vehicular Cloud (VC) formation process – from Base Station (BS) association to scheduling – based on a prediction approach using Long Short-Term Memory (LSTM). 65
5.3 Time series created by the system for a Vehicular Cloud (VC). 66
5.4 Recurrent Neural Network (RNN) employed by MARINA. 67
5.5 Pareto set example with 24 tasks. 69
5.6 Simulation scenario. 73
5.7 Example of the data used for the training and prediction processes. 75
5.8 Results of resource predictions by Long Short-Term Memory (LSTM), Dense Neural Network (DNN), and Support Vector Regression (SVR). . . . 76
5.9 Task arrival rate example and total sent packets on the network. 77
5.10 Results of the scheduled tasks with different maximum deadline constraints and task arrival rates. 78
5.11 Results of the monetary cost with different maximum deadline constraints and task arrival rates. 79
5.12 Results of the system latency with different maximum deadline constraints and task arrival rates. 80
5.13 Results of the Central Processing Unit (CPU) time with different maximum deadline constraints and task arrival rates. 81

6.1 A system architecture employed by FARID, presenting its main components, such as vehicles, Base Stations (BSs), Remote Server (RS), Vehicular Clouds (VCs), and Vehicular Edge Computing (VEC) Controllers. 83
6.2 The task scheduling pipeline with a load-balancing function integrated into the scheduler. 85
6.3 Simulation results considering different deadline constraints. 89
6.4 Jain's fairness index considering different deadline constraints. 91
6.5 Spatial distribution of tasks processed with deadline equal to $[0.5, 0.8)$. . . 92

List of Tables

2.1 Comparison between Vehicular Edge Computing (VEC) and Vehicular Cloud Computing (VCC) [12, 123] 28

3.1 Summary of related work referring to the Vehicular Cloud (VC) formation, highlighting their characteristics and limitations. 34
3.2 Summary of related work referring task scheduling, highlighting their characteristics and limitations. 37

4.1 Simulation parameters for Vehicular Cloud (VC) formation assessments. . . 54

5.1 Summary of key notations for the task scheduling study. 63
5.2 Simulation parameters for task scheduling assessments. 74

6.1 Simulation parameters for fairness and load balancing assessments. 88

List of Algorithms

1 PREDATOR mechanism overview 48
2 Vehicular Cloud Head (VCloudHead) receiving broadcast message from the Roadside Unit (RSU) 49
3 Receiving relay message 50
4 Task scheduling control in VCloudHead 59

5 Vehicular Edge Computing (VEC) environment 71
6 MARINA mechanism overview 72

7 FARID mechanism overview 87

List of Acronyms

AHP Analytic Hierarchy Process

ARIMA Autoregressive Integrated Moving Average

BCP Bin Covering Problem

BoT Bag-of-Tasks

BS Base Station

CH Cluster Head

CPU Central Processing Unit

DBSCAN Density-Based Spatial Clustering of Application with Noise

DNN Dense Neural Network

DQN Deep Q-Network

DRL Deep Reinforcement Learning

DSRC Dedicated Short-Range Communication

EKF Extended Kalman Filter

FCFS First-Come, First-Served

FCMC Fuzzy C-Means Clustering

FFD First-Fit Decreasing

FIS Fuzzy Inference System

GD Gradient Descent

GPS Global Positioning System

GPU Graphics Processing Unit

IaaS Infrastructure-as-a-Service

ILP Integer Linear Program

ITS Intelligent Transportation System

KNN K-Nearest Neighbor

LEO Low Earth Orbit

LiDAR Light Detection and Ranging

LSTM Long Short-Term Memory

LTE Long Term Evolution

MCDM Multi-Criteria Decision Making

MEC Multi-Access Edge Computing

MIPS Million of Instructions Per Second

ML Machine Learning

OBU On-Board Unit

OPTICS Ordering Points to Identify the Clustering Structure

PSO Particle Swarm Optimization

QoS Quality of Service

RL Reinforcement Learning

RMSE Root Mean Square Error

RNN Recurrent Neural Network

RS Remote Server

RSU Roadside Unit

SINR signal to interference plus noise ratio

SUMO Simulator of Urban Mobility

SVR Support Vector Regression

UAV Unmanned Aerial Vehicle

UNDEF Undefined

V2I Vehicle-to-Infrastructure

V2V Vehicle-to-Vehicle

V2X Vehicle-to-Everything

VANET Vehicular Ad-Hoc Network

Vanilla-LSTM Unmodified Long Short Term Memory

VC Vehicular Cloud

VCC Vehicular Cloud Computing

VCloudHead Vehicular Cloud Head

VCloudMember Vehicular Cloud Member

VEC Vehicular Edge Computing

WAVE Wireless Access in a Vehicular Environment

Contents

List of Figures **10**

List of Tables **11**

List of Algorithms **12**

List of Acronyms **15**

1 Introduction **18**
1.1 Motivation . . . 18
1.2 Objective . . . 20
1.3 Main Contributions . . . 21
1.4 Thesis Outline . . . 23

2 Background and Key Concepts **24**
2.1 Vehicular Ad Hoc Networks . . . 24
2.2 Cloud and Edge Computing . . . 25
2.3 Vehicular Edge Computing and Vehicular Clouds . . . 27
2.4 Chapter Conclusions . . . 30

3 Related Work **31**
3.1 Vehicular Cloud Formation . . . 31
3.2 Task Scheduling . . . 34
3.3 Fairness and Load Balancing . . . 38
3.4 Chapter Conclusions . . . 39

4 Vehicular Cloud Formation using Mobility Prediction **41**
4.1 PREDATOR . . . 42
4.2 Problem Definition . . . 45
4.3 Performance Evaluation . . . 51
4.4 Results . . . 53
4.5 Chapter Conclusions . . . 60

5 Mobility- and Deadline-aware Task Scheduling Mechanism **61**
5.1 MARINA . . . 63
5.2 Problem Definition . . . 67
5.3 Performance Evaluation . . . 72
5.4 Results . . . 75
5.5 Chapter Conclusions . . . 81

6 Fairness and Load Balancing in Vehicular Clouds **82**
6.1 FARID . . . 83
6.2 Problem Definition . . . 84
6.3 Performance Evaluation . . . 86
6.4 Results . . . 89
6.5 Chapter Conclusions . . . 91

7 Final Remarks **93**
7.1 Conclusions . . . 93
7.2 Future Work . . . 94
7.3 Scientific Production . . . 96
7.3.1 Work Published in Journals . . . 96
7.3.2 Work Published in Conferences . . . 96
7.3.3 Research Collaborations . . . 97

Bibliography **99**

Chapter 1

Introduction

This chapter presents the motivation, objectives, and main contributions of the thesis, as well as the thesis outline.

1.1 Motivation

In recent years, the number of vehicles has grown significantly worldwide. With more than 1 billion vehicles in circulation, some estimates point out that the number of vehicles growth exceeding 1.2 billion by 2050 [24, 98, 96]. Furthermore, the number of connected vehicles globally is expected to grow from 192 million this year to over 367 million in 2027 [149]. Especially in Brazil, the vehicle fleet has been growing every year since 2010, according to a recent report presented by the National Union of the Component Industry for Motor Vehicles [133], as can be seen in Figure 1.1.

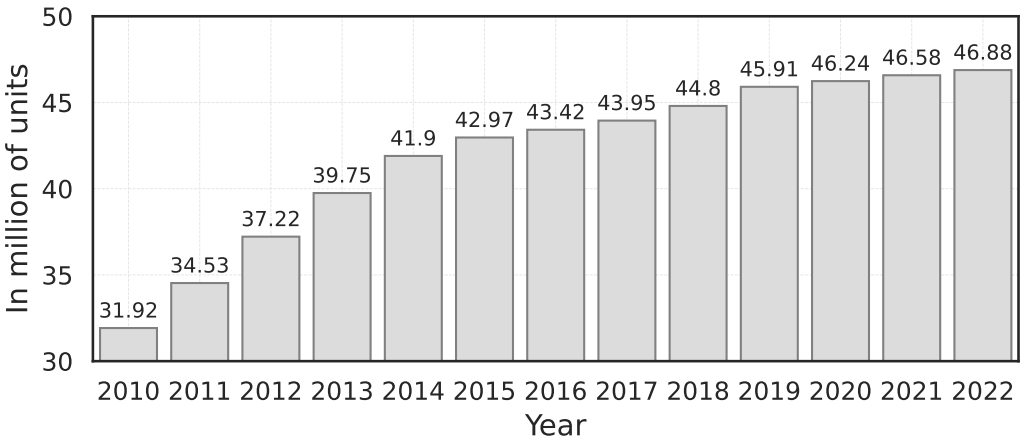


Figure 1.1: Circulating fleet of vehicles in Brazil between the years 2010 and 2022.

In this sense, the growing number of vehicles has resulted in frequent traffic jams and accidents on the roads, in addition to causing an increase in the amount of data to be transferred to cloud servers [44, 14]. However, traffic jams and accidents can be mitigated by providing drivers with adequate safety information about road conditions, surrounding environments, and critical driving situations [98]. Thus, in addition to meeting the growing demand in the automotive sector, there is also an effort by academia and industry to

enable vehicles to become increasingly intelligent, reliable, safe, and, thus, to be able to act as active agents in the processes that they involve, whether on highways or in urban centers [105, 42, 120, 141].

Vehicles today have a variety of computing resources to provide maximum comfort and safety to drivers and passengers [42, 153]. The vehicle modernization process concerns both the increase in wireless communication capacity and the addition of computational and sensing power, directing them towards increasingly intelligent and connected devices [48, 82]. For example, the Tesla Inc.'s Model X electric vehicle has connectivity to the Long Term Evolution (LTE) network, frontal radar for identifying objects up to 250 meters away, 8 cameras, and 12 ultrasonic sensors for detecting track lanes and surrounding objects, providing 360 degrees of visibility, speed adjustment based on visual identification of traffic signs and overtaking management in a totally autonomous way [66].

In the same direction, Vehicular Ad-Hoc Networks (VANETs) emerged as a wireless communication paradigm that allows direct communication between vehicles and other connected devices, through Vehicle-to-Everything (V2X) communication [31, 146]. With VANETs, vehicles can collect and share data to assist other entities in their decision-making processes, whether locally disposed entities or remote ones on the Internet cloud. In this context, several applications can be proposed for vehicular problems, such as detection and control of traffic jams, collision prevention, road warnings, and other applications of Intelligent Transportation Systems (ITSs) [12, 110].

A consolidated research area that is strongly related to VANETs is Cloud Computing, which by definition emerged to relieve the processing that was previously executed locally and is now executed in instances over the Internet [123, 79]. However, the rapid evolution of mobile devices, their applications, and the amount of data generated by them cause a significant increase in bandwidth consumption and congestion in the Internet network core [14]. In this scenario, Edge Computing emerges, making cloud computing services closer to end users [20]. Edge is an extension of Cloud Computing that brings resources to the network edge, aiming to reduce latency in serving user demands and reduce traffic in the network core. This paradigm was conceived to support applications and services that do not fit well in the traditional Cloud Computing paradigm, that is, that require characteristics such as low latency, geographic distribution, mobility, high resiliency, and large-scale distributed systems [23, 125] .

Given this evolutionary scenario, the Vehicular Edge Computing (VEC) paradigm arises, primarily aggregating and utilizing the vehicles' computational resources available. In other words, VEC merges the operational principles of VANETs and Cloud Computing services, emerging as a non-trivial variant of traditional Cloud Computing [111]. The computational resources available in vehicles include processing, communication, storage, and sensing capabilities, which can be dynamically collected and grouped into vehicle groups, otherwise known as Vehicular Clouds (VCs). The cloud, drivers, passengers, and users all benefit from the aggregation of these resources and their efficient distribution as cloud computing services closer to vehicular users [12, 4]. This assumption serves as the basis for many researchers in the VEC field, a relatively new and promising area for enhancing the computational resources available in the vehicular environment [50, 59, 16, 159, 19, 29, 14].

The use and management of VC resources can be divided into two main steps: *(i)* VC Formation, which aims to group the vehicles based on some characteristic, and with that, the computational resources available in these vehicles are grouped; and *(ii)* Task Scheduling, whose purpose is to decide which VC a particular application will run on, as well as manage in real time the resources available in that cloud after the scheduling process. However, due to the high mobility of vehicles, some challenges must be overcome. The VC formation is directly compromised by vehicular mobility, since the stability and maintenance of these clouds depends on robust algorithms that somehow mitigate the impact of mobility. Furthermore, if the VCs formed are not stable, the task scheduling process becomes imprecise, since it is important to ensure that the processing times (*deadlines*) of the tasks are respected. Another important point is the fair use of aggregate computational resources. That is, the scheduler must apply, whenever possible, a load balance between the available VC and avoid processing overhead on specific VCs.

Considering all points mentioned above, it is important to provide solutions that use the computational resources of the VCs and the constructed knowledge of the scenario for decision-making regarding the storage and processing data. Besides, the type of data collected and the service that will be offered to the user to meet the most diverse operating scenarios must be taken into account. In other words, it is essential to seek solutions for creating stable VCs that allow maximum use of the resources present in vehicles. In addition, it is important to propose solutions that consider the VCs formed to make such added resources available to vehicle users who need processing power and/or storage to execute one or more services. This resources availability must be fair and balanced to avoid overloading and underutilizing the available VCs.

1.2 Objective

The main objective of this thesis is the efficient resource management in VEC systems through the design, implementation, and evaluation of collaborative systems for VC formation and fair and balanced task scheduling. To achieve this goal, we need to answer the following questions:

- Many approaches for VC formation have been proposed in the literature with the aim of selecting the most stable vehicles in a given region, such as Roadside Unit (RSU)/Base Station (BS)' coverage radius, and making it the leader of this group. This problem revisits the classic Cluster Head (CH) definition problem in computational clusters. In the case of VCs, mobility must be taken into account when choosing these leading vehicles. If the leader selection is not efficient, the VCs will not be stable enough to process tasks that require a non-trivial processing time.

  Research Question 1: *How to form more stable VCs and ensure longer lifetimes?*

- High mobility of nodes is the main characteristic of VANETs. In this sense, strategies must be employed so that, when using the computational resources of the VCs, which are mostly composed of vehicular resources, vehicular mobility is not a factor that directly impacts the performance of scheduling solutions. Many works try to

mitigate the impacts of mobility using techniques to estimate the future positioning of vehicles and make decisions based on this information. Some of these approaches need a priori knowledge of vehicle dynamics, which makes these solutions dependent on accurate vehicle mobility information.

Research Question 2: *How to ensure that the topological dynamics of the Vehicular Ad-Hoc Networks (VANETs) does not negatively influence task scheduling in Vehicular Clouds (VCs)?*

- Numerous works consider VC formation processes aided by communication infrastructures, such as RSUs and BSs. Vehicles have their resources added while they are covered by these infrastructures. If several VCs are formed in a city and the scheduler identifies certain regions with high resource availability, it is natural for the scheduler to prioritize these VCs in its decision-making process. However, if we look at it from a load balancing and fairness perspective, this scheduler is concentrating in a specific region all the processing workload of the VEC environment. This unbalanced decision can overload infrastructures and negatively impact the Quality of Service (QoS) offered by this infrastructure to other applications.

  Research Question 3: *How to use the computational resources of Vehicular Clouds (VCs) in a fair and balanced way without degrading the system's overall efficiency in task scheduling?*

1.3 Main Contributions

The main thesis contributions are a mobility-aware vehicular cloud formation mechanism for VEC environments, a mobility- and deadline-aware task scheduling approach, and a fairness task scheduling study on the multiple available VCs. In summary, we have:

1. **Vehicular Cloud Formation Using Mobility Prediction Information**

   This contribution concerns the proposal of a mobility-aware mechanism that enhances the VC 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 VC, thereby increasing the VC's lifetime. Vehicle stability is measured by the dwell time, representing the duration a vehicle spends within the RSU coverage area. 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 VC formation process. The number of VCs formed is proportional to the number of RSUs in the scenario. Additionally, a VEC Controller at a higher level in the network can oversee these VCs and manage their aggregated resources. Simulation results (presented in Section 4.4) demonstrate the superior performance of this approach compared to other VC formation mechanisms.

2. **Mobility- and Deadline-aware Task Scheduling Mechanism**

This contribution concerns the proposal of a task scheduling mechanism to maximize the number of tasks scheduled while minimizing the monetary costs of utilizing VC's resources. The proposal runs on VEC controllers to coordinate the task scheduling in multiple VCs. A vehicle without enough computational resources to run a specific task can forward this task to be processed somewhere in the vehicle ecosystem. In this sense, the solution selects a set of tasks to be scheduled in real-time in each available VC based on the Pareto optimality and Bin Covering Problem (BCP). Pareto optimality allows the joint minimization between the deadline and estimated processing time. Besides, the BCP makes it possible to find the best fit for the minimization provided by Pareto, always seeking to maximize the number of scheduled tasks. The proposal also considers a Long Short-Term Memory (LSTM) to predict resource availability in each VC based on vehicular mobility information. Hence, prioritizes scheduling tasks in VCs with more available resources to maximize the fulfillment of demands in as few rounds as possible. Simulation results show that (presented in Section 5.4), compared to state-of-the-art solutions, the solution proposed can schedule more tasks while minimizing monetary cost and system latency.

3. **Fairness and Load Balancing in VEC Systems**:

   This contribution concerns the proposal of a fair task scheduling approach that maximizes both successfully scheduled tasks and load balancing and fairness in the use of computational resources. This solution runs on VEC controllers and uses Pareto optimality to schedule tasks in different VCs. The mechanism splits the set of tasks into different parts to improve the system efficiency with parallel management, obtaining k different Pareto sets and being able to make k decisions at the same time, where k is the number of threads running in each VEC controller. It aims to minimize processing time within VCs, thus reducing resource utilization and, subsequently, monetary costs. Also, it considers contextual aspects in its decision process, such as resource mobility in each VC and task's requirements. We assessed the efficiency of this approach compared to other mechanisms, and the results (presented in Section 6.4) indicate its capability to schedule a larger quantity of tasks, minimize monetary costs, and reduce overall system latency. Lastly, the proposal employs better load balancing in the scheduling process, resulting in greater fairness in the resource usage of VCs.

It is important to highlight that the solutions presented in this thesis are published in relevant conferences and prestigious journals on computer networks and communication. The conferences in which the solutions were published include *IEEE Vehicular Technology Conference (VTC)* [39, 37, 34] and *Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)* [40, 36, 32]. On the other hand, the published journals: *IEEE Transactions on Intelligent Transportation Systems* [35], *Elsevier Ad hoc Networks* [38], and *Elsevier Vehicular Communications* (Under Review). The complete list of publications are presented in Chapter 7 in Section 7.3.

1.4 Thesis Outline

The structure of this thesis is outlined in chapters as follows:

- **Chapter 2** provides the background for this thesis describing a theoretical basis on VANETs and their challenges. Also, it presents the concept of Cloud and Edge Computing and how the union between these two paradigms happens to create the VEC paradigm. Finally, a comprehensive discussion is offered, examining characteristics and challenges associated with this scenario.

- **Chapter 3** describes the related work highlighting the limitations and advantages of literature solutions for VC Formation, Task Scheduling, and Fairness and Load Balancing. Moreover, qualitative comparison and classification are proposed.

- **Chapter 4** proposes and evaluates an efficient VC formation mechanism that uses mobility prediction to select the most stable vehicles in the network to coordinate their respective VCs, called PREDATOR. Such an approach mitigates the constant exchange of information between the vehicles and the RSU that executes the mechanism. The stability of PREDATOR is tested with a real time task scheduling application and it proves to be efficient.

- **Chapter 5** proposes and assesses a mobility- and deadline-aware task scheduling mechanism for VEC, called MARINA. MARINA uses Pareto optimality and BCP to schedule tasks in the available VCs. Furthermore, MARINA employs an Recurrent Neural Network (RNN) architecture to estimate computational resources in each VC. MARINA efficiently schedules more tasks and reduces both system latency and monetary cost.

- **Chapter 6** presents and evaluates a task scheduling mechanism that considers contextual aspects of its decision process and applies a probabilistic selection function on VCs to balance the processing load and increase the fairness in the use of vehicular resources, called FARID. Also, FARID applies a segmentation in the task queue to reduce the scheduling problem and uses *multithreading* to solve each subproblem in a parallel way.

- **Chapter 7** concludes this thesis with a summary, describes directions for future work, discusses the challenges faced, and presents the publications produced from this thesis.

Chapter 2

Background and Key Concepts

This chapter presents the concepts that will be used in this thesis. Initially, a theoretical basis on VANETs and their challenges is presented. Then, a study of Cloud Computing is carried out and how the union between Cloud and Edge Computing happens to create the VEC paradigm and its main entity called VC. In addition, a discussion of the unique challenges and characteristics that VCs have is presented.

2.1 Vehicular Ad Hoc Networks

In VANETs, the nodes are the vehicles, and the orientation of the public road limits the movement of these nodes. VANETs inherit the main characteristics of Mobile Ad hoc Networks and can function in three communication types. The first is Vehicle-to-Infrastructure (V2I) communication, with access points to provide Internet access and routing for transmissions. Such infrastructures can be either RSUs or BSs. Vehicles communicate with this infrastructure to access the desired service, as illustrated in Figure 2.1. However, this type of communication has high implementation and maintenance costs [30].

Another type of communication is Vehicle-to-Vehicle (V2V), where all nodes communicate directly with each other and play the role of routers for forwarding data through multiple hops between source and destination. This is possible through a vehicle-installed networking device called an On-Board Unit (OBU) that connects to the Dedicated Short-Range Communication (DSRC) wireless network. Connectivity between nodes depends on the network density and how vehicles are moving along the road, that is, their mobility pattern. Figure 2.1 illustrates V2V communication between two vehicles.

Finally, there is the V2X communication, which represents the communication between the vehicle and any other device [47, 8]. The node can communicate with another node *ad hoc* and/or communicate with an infrastructure to consume certain content and/or service. Figure 2.1 illustrates V2X communication between vehicle-Unmanned Aerial Vehicle (UAV) and vehicle-Intelligent Traffic Light. This type of communication is widely explored because it combines different services, always seeking to adapt to the user's needs. For example, at a certain point in the trip, passengers decide to watch a video on the Internet, and, at another point in the trip, a message about an accident on the road

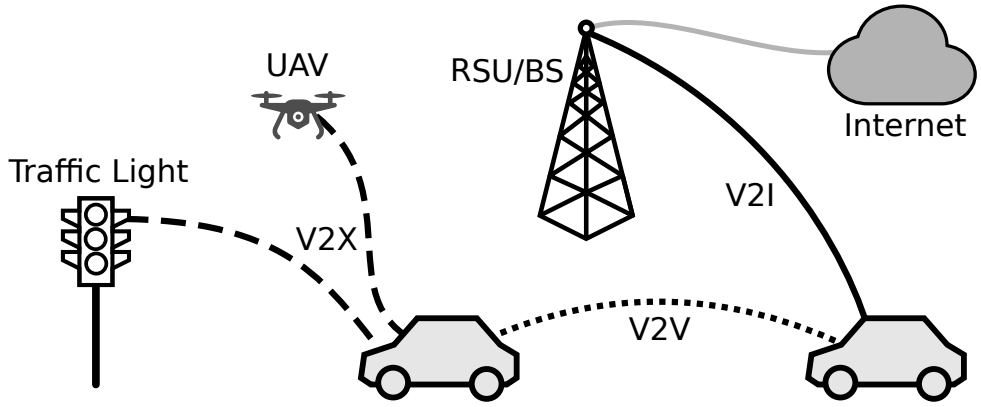


Figure 2.1: Different types of communication in VANETs.

is propagated between the vehicles to avoid accidents.

This type of network is already quite exploited, so much so that it has its own DSRC spectrum. It uses IEEE 802.11p as a wireless network interface, also known as Wireless Access in a Vehicular Environment (WAVE) which has been adjusted for operations with low *overhead* in the DSRC spectrum, with a range of hundreds of meters and transfer rate of up to 27 megabits per second [78]. This communication possibility was initially thought for security applications, which allocate a certain propagation spectrum, however, this was expanded and other categories of applications are supported.

In this scenario, applications on VANETs can be classified into three categories [30]:

- **Safety and Security**: that allows the driver to receive warnings about road conditions, accidents, and warnings about events that may impact their well-being and that of passengers.
- **Entertainment**: where most applications are associated with Internet access and use inside the vehicle, for video, image, and other media consumption.
- **Driver Assistance**: applications focused on automating certain tasks such as, for example, locating gas stations, service areas, and tourist information.

Each application has well-defined and distinct operating requirements, such as security applications requiring high propagation speed and low delivery delay. On the other hand, entertainment and assistance applications require more bandwidth, as the content is often multimedia (image, audio, and video). Moreover, traffic management applications that fit the driver assistance category require data describing traffic mobility patterns to efficiently serve the central management or vehicles comprising a distributed traffic management system. Currently, learning applications for artificial intelligence require a high amount of data exchanged among network components [65].

2.2 Cloud and Edge Computing

Cloud Computing is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources. These resources

can be servers, storage, applications, and services, which can be rapidly provisioned and released with minimal management effort or service provider interaction [101]. Some important and innovative features have been introduced by Cloud compared to traditional local systems.

Among these advances, the following can be highlighted:

- On-demand provisioning of computing resources, storage, and services. Which does not lock the user into planning the supply of physical resources.
- Resources "unlimited" available to users. With numerous *data centers* distributed around the world, Cloud companies are investing more and more so that this "infinite" resource capacity remains valid.
- Possibility for users to rent services and resources according to their needs. Offering flexible alternatives for acquiring extra hardware resources only when a real need is handy for small companies, leaving aside the need to build local infrastructure.

Other characteristics of the Cloud is the way of offering the resources. This selective offering can be divided into three major types, described as follows.

- **Infrastructure-as-a-Service (IaaS)**: A cloud service provider makes elastic resources such as computing resources, network connections, and storage capacity available to customers through virtualization technology [54]. Amazon EC2[1] and Google Compute Engine[2] are examples of this service.
- **Platform-as-a-Service (PaaS)**: PaaS provides the means for users to develop, run, and manage applications, providing a platform where users do not have to worry about the lower-level details of the environment. Microsoft Azure[3] and Google AppEngine[4] are examples of this type of service.
- **Software-as-a-Service (SaaS)**: The provider is responsible for maintaining software, licensing applications, and making software available to customers on-demand. Typically, the suite consists of office suites, messaging system, a data management system, customer relationship management systems, and antivirus. IBM Cloud[5] is an example of a provider offering this type of service.

However, it was identified that some applications were unsatisfied with the Cloud resources. The major problem is that specific requirements of some applications, such as low latency and real-time processing, were not met with the efficiency required by the fact that the cloud *data centers* are geographically distributed worldwide. Also, the massive data exchange between devices and applications in the Cloud can cause congestion in the network core [64, 14]. In this sense, a type of cloud computing emerges

[1] `https://aws.amazon.com/ec2`
[2] `https://cloud.google.com/compute`
[3] `https://azure.microsoft.com/en-us/`
[4] `https://cloud.google.com/appengine/`
[5] `https://www.ibm.com/cloud`

that brings processing, communication, and storage resources closer to the user to meet the requirements of specific applications. This type of cloud computing is called Edge Computing [125].

The Edge Computing paradigm attracts the attention of researchers for its potential to satisfy requirements previously not met by traditional Cloud Computing [55]. Edge Computing helps devices with low resource capacity and extends the computational resources available in the cloud infrastructure to the network edge, providing mobility, scalability, low latency, and robustness to users [119]. Resource sources that were geographically distant from users can now be nearly and thus serve applications with a critical level of response. However, considering the vehicular scenario that proves to be quite costly when depending on infrastructure to support the execution of centralized applications/services, having the possibility of using the vehicles' resources for data processing and storage can reduce latency and cost deployment for critical applications.

2.3 Vehicular Edge Computing and Vehicular Clouds

Vehicles are increasingly intelligent and connected, with considerable energy, processing, storage, and communication resources. Such available resources can be used in cooperation by a group of vehicles and thus create a local cloud to process tasks that were previously performed in the Cloud or even in the Edge [12, 4, 14]. This grouping of vehicles is known as VC. With this, vehicle resources can be made available for applications not necessarily part of the vehicular scenario. In summary, the methods used to make resources available in traditional Cloud Computing can be incorporated into this new vehicular scenario [100, 144].

In this way, researches present VC with the primary objective of gathering and using the resources available on board vehicles. These resources encompass processing, storage, and sensing, which can be dynamically collected in groups of vehicles under drivers' authorization. The cloud, drivers, passengers, and users benefit from efficiently aggregating all the resources around and efficiently delivering them as cloud services to the network [12, 111, 14].

VC services are complex and relevant, which complement traditional Cloud Computing services. As defined by some researchers, VC refers to *"a group of vehicles that are largely autonomous and contain computing, communication, sensing, and storage resources that can be coordinated and dynamically allocated to duly authorized users"* [112, 12]. Some authors differentiate the concepts that encompass the VC depending on how the coordination of the clouds occurs. For example, there is the Vehicular Cloud Computing (VCC) concept which, in essence, uses the resources of the traditional Cloud to provide vehicular services. It behaves like a centralized approach [12, 123]. There is also Vehicular Edge Computing (VEC), which uses vehicle resources and the traditional Cloud to provide services, but has coordination at the network edge or even among the vehicles themselves [14, 123].

VEC is inspired by Multi-Access Edge Computing (MEC), which emerged with a key role in enabling mobile users to use computational resources at the cellular network edge,

thus unburdening traffic intense to the network core [139]. With this, benefits such as low latency, higher bandwidth, and location-based services are possible for users. In the vehicular context, applications such as vehicular security applications can use the MEC for services with strict latency requirements. However, some other applications require large amounts of data exchanged between vehicles or other entities to perform complex tasks, such as real-time learning mechanisms for autonomous vehicles or Artificial Intelligence-oriented applications, such as those expected in 5G and 6G networks [42, 97, 155, 27]. Table 2.1 shows a comparison between VEC and VCC.

Table 2.1: Comparison between Vehicular Edge Computing (VEC) and Vehicular Cloud Computing (VCC) [12, 123]

Feature	VEC	VCC
Location	At user's proximity	Remote
Latency	Low	High
Mobility support	High	Limited
Decision making	Local	Remote
Communication	Real time	Constraints in bandwidth
Storage capacity	Limited	Highly scalable
Processing capacity	Medium	High
Resource flexibility	Dynamic	Highly dynamic
Context awareness	Yes	No
Device heterogeneity	Highly supported	Limited supported
Autonomous composition	Yes	No
Network architecture	Peer-to-Peer or Client-Server	Client-Server
Battery limitation	No	No
Cost of development	Low	High

Regarding computational and storage capacities, it is noticeable that the VCC has more resources than the VEC. Furthermore, mobility is a valuable and inherent part of the physical VEC resources, whereas VCC resources are commonly located in the fixed data center. It is also known that VCC does not have limited energy resources due to the constant supply of energy in the data center. Likewise, the VEC does not have limited energy resources since a vehicle contains a large capacity battery, and the operating mechanism incorporated by automakers can recharge it continuously.

Modern vehicles already have powerful computational resources to use sophisticated applications and services. The Cloud can expand such capabilities by providing unlimited resources, thus increasing access to services. In addition, built-in vehicle features tend to be underutilized for long periods, such as when vehicles are parked or stuck in traffic jams. Such situations raise promising opportunities to exploit underutilized computing resources to assemble dynamic clouds autonomously, thus providing even more computational capacity for various services.

Conceptually, neighboring vehicles can participate in a VC, with some strategy that contributes to addressing complex issues in real-time, which can lead to a significant evolu-

tion for transport systems [111]. The vehicular mobility and the environment dynamicity require sophisticated strategies for managing resources efficiently [103, 13]. VEC is essential for the design of the robust solutions for the VANETs environment, which includes support for a wide range of applications and services [12, 65].

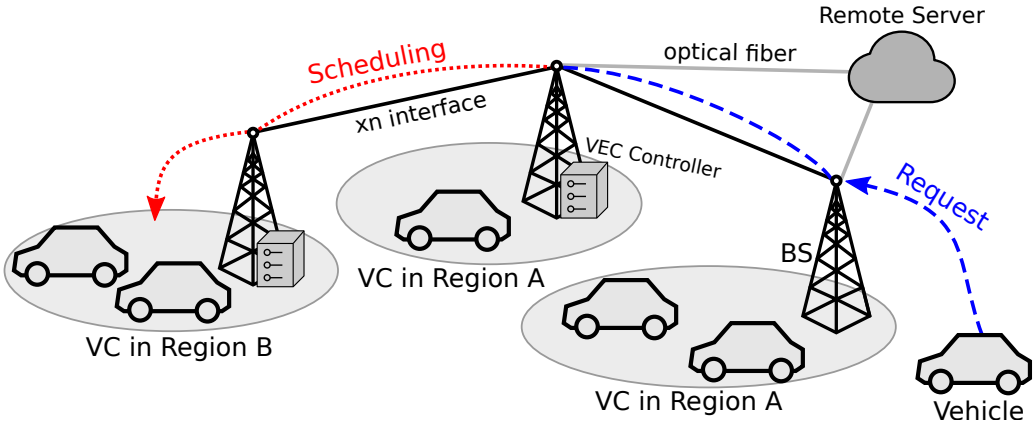


Figure 2.2: A traditional VEC system architecture, presenting its main components, such as vehicles, Base Stations (BSs), Remote Server (RS), Vehicular Clouds (VCs), and Vehicular Edge Computing (VEC) Controllers.

Figure 2.2 presents an example of a VEC architecture assisted by a 5G network. In this case, the VCs are formed with the help of the BS deployed in the city. In addition, network controllers (VEC controller, in this case) are placed in the scenario to help the decision-making process at the network's edge. A Remote Server (RS) in the Internet cloud can be used to store specific authentication information or digital media of users, for example. Thus, after a VC formation process, users can request resources on the network, and the controller decides which VC will process the demands of these users.

In the VEC scenario, the use and management of resources can be divided into two main steps: *(i)* VC Formation, which aims to group vehicles based on some characteristic, and with that, grouping the computational resources available in these vehicles; and *(ii)* Task Scheduling: whose purpose is to decide in which VC a given application will be executed, as well as to manage in real-time the resources available in that cloud after the scheduling process. However, due to the high vehicular mobility and the dynamic nature of VANETs, some challenges must be overcome. The VC formation process is directly compromised by vehicular mobility, since the stability and maintenance of these clouds depend on robust algorithms that mitigate the impact of mobility. Furthermore, if the VCs formed are not stable, the scheduling process becomes imprecise, since it is essential to ensure that the processing times (*deadlines*) of the tasks are respected.

In this sense, as this thesis focuses on using and managing resources present in vehicles to provide greater computational capacity in the network, it is essential to emphasize that the solutions presented here operate in the VEC scenario. Although extensive efforts have been applied to find solutions that deal with the highly dynamic changes of communication topology in VANETs, challenges persist and are actively present in VEC, which depends entirely on vehicular communication.

The VEC architecture can be divided into three main levels [123, 89], namely:

- **Cloud Layer**: the most important advantages of the cloud tier are data aggregation, data mining, analytics optimization, storage, batch processing, and complex data computing, which are beyond the computing power of edge nodes. This layer can handle extensive applications that do not require low latency in their operations.
- **Edge Layer**: this layer connects the cloud layer and intelligent vehicles. This connection is enabled by the communication capabilities inherent in the vehicles. The aim is to provide low latency, location awareness, caching, and content discovery and enhance computational power. The edge layer can improve the quality of real-time services by being close to vehicles.
- **Vehicles Layer**: This layer contains a group of geographically close vehicles that share computing and storage resources over the wireless network. It is responsible for abstracting information from built-in sensors, Global Positioning System (GPS), cameras, radar, Light Detection and Ranging (LiDAR), and other devices that vehicles may contain. This sensory information can be sent to the other layers and thus provide input to various services.

2.4 Chapter Conclusions

VANETs represent an essential paradigm for providing applications and services to drivers and passengers. The advantages of this type of technology are even more evident with the advancement that the automobile industry has been using in vehicles to make them increasingly intelligent, connected, and autonomous. Allied with this, Cloud Computing, together with VANETs, can provide cloud services for vehicular users, such services concern the availability of computational resources for the execution of vehicular applications that require high processing power and low response latency. In this way, the availability of vehicular computational resources to bring the processing power closer to the users originates the paradigm known as VEC. Furthermore, groups of vehicles formed in the VEC are called VCs. This chapter brought concepts from VANETs and Cloud and Edge Computing and how the union of these paradigms can benefit the vehicular environment.

Chapter 3

Related Work

This chapter presents a classification, review, and qualitative literature analysis for each important process in VEC systems, such as VC formation, task scheduling, and fairness and load balancing. Section 3.1 describes the proposed solutions for the VC formation. Section 3.2 presents the state-of-the-art about task scheduling in VEC systems. Section 3.2 describes solutions to increase fairness and load balancing levels in vehicle resource usage. At last, Section 3.4 concludes the chapter.

3.1 Vehicular Cloud Formation

This section presents state-of-the-art research regarding grouping network devices to use their computational resources for different purposes. However, we analyzed these works from a perspective for application in VCs formation scenarios, which are usually formed using traditional clustering concepts in VANETs [26, 6]. In general, VC formation approaches can be classified into two classes. *(i)* Infrastructure-based, where the entity that makes the grouping decision is concentrated in a communication infrastructure, such as RSU or Cellular BS. This approach allows resources to be better managed, and the infrastructure can provide minimal computational resources to service requests even without vehicles in its coverage. With the advancement of 5G networks, these communication infrastructures are viable in this vehicular scenario [49, 152]; and *(ii)* Distributed, where the grouping decision is made individually by each node that is part of the network. This approach, specifically for the VCs scenario, makes the resource management process more challenging and only guarantees minimum computational resources to meet demands in areas with communication infrastructures. In this sense, we divided the related works into three perspectives, considering the support provided by the infrastructures, the grouping strategy considered, and whether the approach considers some mobility information in its decision process.

For instance, Zhao *et al.* [157] 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.* [73] 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.*, 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.* [33] introduced a mechanism that considers the VC 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 VCs. Besides, the choice of the Vehicular Cloud Head (VCloudHead) in this approach is carried out by calculating the centroid in the spatial distribution of the VC and identifying the vehicle closest to this centroid.

Peixoto *et al.* [115] 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.* [17] 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 VCs: 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.* [2] 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 interference plus noise 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.* [156] 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.* [58] presented a map-based approach to VC 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 VCs, allowing for more complex services and covering entire cities. This work shows that an efficient VC formation enables vehicular applications to use aggregated computational resources efficiently. In this approach, the VC formation process is limited to intersections in urban environments and the VCs 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.* [147] 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.* [95] 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 3.1 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 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 VCs 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 [32], 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 VC members and enhancing resource management capacity [113].

Table 3.1: Summary of related work referring to the Vehicular Cloud (VC) formation, highlighting their characteristics and limitations.

Work	Grouping strategy	Infrastructure-based	Mobility info.
Zhao et al. [157]	Degree centrality	✓	
Hagenauer et al. [58]	Centroid	✓	
Da Costa et al. [33]	DBSCAN	✓	
Peixoto et al. [115]	OPTICS & DBSCAN	✓	
Bute et al. [17]	Fuzzy		
Kamakshi et al. [73]	Centrality metrics		Relative mobility
Abbasi et al. [2]	Fuzzy C-Means	✓	Vehicular density
Zhao et al. [156]	Fuzzy C-Means	✓	Moving direction
Wang et al. [147]	KNN + Machine Learning	✓	Trajectory
Magaia et al. [95]	Social + Machine Learning	✓	Social contacts
Da Costa et al. [32]	Dwell time	✓	Prediction

3.2 Task Scheduling

This section presents state-of-the-art research regarding task scheduling in VCs. In general, scheduling approaches can be classified into three perspectives. *(i)* Centralized, where an entity has a scenario holistic view and makes more accurate decisions. This approach has the advantage of an environment global view, but maintaining this knowledge is a challenge for system scalability; *(ii)* Decentralized, which has different agents that make decisions based on local knowledge. This approach does not need to maintain global knowledge, but the accuracy of decisions is compromised in some cases; and *(iii)* Hybrid, which combines the advantages of two previous techniques to increase system efficiency [84]. Observing the advantages of each scheduling approach, we consider only the Centralized and Hybrid approaches due to the essential role the 5G network can play in these specific scenarios.

Pereira et al. [117] proposed FORESAM, a policy for scheduling tasks in VCs based on the fog computing paradigm for urban environments. Specifically, vehicles cooperate with the set of BSs to create a pool of resources for vehicular services. FORESAM decides whether resources are available based on Analytic Hierarchy Process (AHP) mathematical method. However, FORESAM employs greedy decision-making, where in case a task does not fit into the VC, it is discarded, impacting the overall system efficiency.

Some works consider optimization algorithms for decision-making. Da Costa et al. [33] introduced CRATOS, a combinatorial optimization-based mechanism for task scheduling in VCs. CRATOS considers the arrival of tasks in real-time and regardless of each other following a Poisson process. The VEC controller located at a higher level in the network receives the resource requests (which are the tasks) and schedules them in the available VCs. 0/1 Knapsack Problem is used to schedule tasks optimally given the contextual configuration (tasks' input size and VC's available computational resources). However, CRATOS does not consider tasks with deadlines higher than 1s in its decision process. Also, the user defines how much it will cost to process his task. Hence, CRATOS would

not fit into an actual application, as the monetary cost model used by cloud providers is different.

Dai et al. [41] presented a probabilistic algorithm for cooperative task scheduling in VCs. The authors formulate Cooperative Computing Offloading (CCO) problem by modeling the procedure for uploading, migrating, and scheduling tasks based on queuing theory to minimize the delay in the system. The BS makes online scheduling based on the calculated probability provided by a probabilistic offloading algorithm. Three aspects are considered for computing the offloading probability: the time that vehicles will stay in BS's coverage range; the density of vehicles; and resources available in the BS. However, this computation is offline, which means it is executed on a high-performance server in the Internet cloud. The probability that the BS will meet demand is computed before the process of task arrival, which can degrade the system's performance in a highly dynamic environment. Finally, VCs considers only the resources available in the BSs.

Luo et al. [94] presented a detailed analysis of the delay and cost of task offloading for VCs. The authors first establish an offloading framework with communication and computation for VC, considering tasks with different requirements. In this sense, a multi-objective optimization problem is formulated to joint minimization the delay and the monetary cost. A Particle Swarm Optimization (PSO)-based computation offloading (PSOCO) algorithm is proposed to obtain the Pareto-optimal solutions. However, due to its bio-inspired approach, its convergence time can impact the algorithm's performance.

Some works use queuing theory to model the vehicular scenario. Hattab et al. [60] proposed a polynomial time algorithm for task scheduling in VCs with different resources. First, the algorithm classifies tasks according to the completion and wait times. Afterward, it selects a subset of tasks with the lowest proportion and then solves a sequence of Linear Programs. They formulate the bottleneck assignment problem, where the goal is to minimize the completion time of the scheduled tasks in the available VCs. However, this work does not consider the vehicle mobility for VC formation, *i.e.*, VCs are stationary, and the proposed algorithm considers only one VC.

Some approaches use ML techniques for the scheduling decision process. Kazmi et al. [77] proposed a task scheduling mechanism for mobile vehicular networks using a Deep Reinforcement Learning (DRL)-based approach. The approach considers electric vehicles in task processing. The focus is to make energy consumption more efficient. Mobility information considers the relative mobility and kinematic equations to estimate the communication phase between two vehicles. However, an evaluation comparing the solution with state-of-the-art is still necessary. Furthermore, the BS does not cooperate in task processing, decreasing system efficiency.

Gao et al. [53] present a solution for task scheduling that aims to minimize service delay and energy consumption. The authors use a Deep Q-Network (DQN) approach for scheduling decision-making and a Gradient Descent (GD) method for Central Processing Unit (CPU) frequency allocation. However, the work needs to discuss the solution convergence time, which depending on the scenario, can make it unfeasible for more dynamic environments.

Chen and Xu [19] presented a Reinforcement Learning (RL)-based algorithm for task scheduling in VC, called DATE-V, which provides resources for scheduling tasks with

deadline constraints in the VCs. DATE-V is based on a contextual and combinatorial Multi-Armed Bandit (MAB) learning framework. The algorithm uses vehicle contextual data (*i.e.*, speed, location, and computational resources available) to infer the probability of completing a task replication under random vehicles. DATE-V also replicates the task received in the BS and sends such replications to different vehicles to ensure service attendance and increase its success rate. However, these replications potentially lead to a costly solution, given that defining the number of replications is challenging.

Liu et al. [90] proposed a vehicle-assisted task scheduling approach for mobile users, considering delay constraints. The authors formulate an optimization problem to maximize the long-term usefulness of VC's resources. Considering stochastic vehicle traffic, dynamic computation requests, and time-varying communication conditions, the problem is later formulated as a Semi-Markov Decision Process (SMDP). Two reinforcement learning methods are proposed to obtain the optimal computation offloading and VC's resource allocation, namely *(i)* Q-learning based and *(ii)* Deep Reinforcement Learning (DRL) methods. However, given the nature of DRL approaches, it is necessary to train the model in advance. Thus, the solution depends on the offline phase for its execution, which can negatively impact highly dynamic scenarios.

Some works consider mobility prediction to estimate vehicle positions and ensure reliability in the task scheduling process. For example, Misra and Bera [107] proposed a mobility-aware task scheduling scheme named Soft-VAN, which aims to minimize task computation delay in a software-defined vehicular network. Soft-VAN consists of two phases, fog node selection and task scheduling. An Integer Linear Program (ILP) is solved in the first phase to get the optimal number of fog nodes required for a given network. In the second phase, the authors formulate an optimization problem to minimize the delay in task computation and consider the vehicle's mobility and the parameters associated with the scheduling decisions. However, the algorithm depends on an offline phase to identify fog nodes to assist the delivery process of the tasks performed. Also, they do not consider the vehicles' dynamics to make the processing decisions for the task, only to send the final result.

Wu et al. [151] investigate the task scheduling and resource allocation optimization problem by considering the vehicular mobility effect in the VEC environment. Specifically, the authors formulate the joint optimization problem from a Min-Max perspective to reduce the overall task latency. Then they decompose the non-convex problem into two sub-problems, one-to-one matching and bandwidth resource allocation. Also, considering a vehicle's relatively stable moving patterns in a short period, the authors further introduce mobility prediction to design a mobility prediction-based scheme to obtain a better solution. However, the mobility model is unrealistic since vehicles need constant acceleration during the task scheduling.

Table 3.2 summarizes the main characteristics of reviewed studies considering system design, the entities forming the VCs, the approach used in the task scheduling process, and vehicular mobility information. Based on the state-of-the-art analysis, we conclude that centralized approaches allow the constructed knowledge to be global and more accurate. However, the network's overload levels can be high and overload the centralized entities, raising the monetary costs of using computational resources [60, 33, 107, 151]. In

Table 3.2: Summary of related work referring task scheduling, highlighting their characteristics and limitations.

Work	Characteristics			
	System design	VC's entities	Scheduling strategy	Mobility info.
Hattab et al. [60]	Centralized	Vehicles	Queueing theory	
Da Costa et al. [33]	Centralized	Vehicles	Optimization	
Pereira et al. [117]	Hybrid	Vehicles/BS	AHP	
Dai et al. [41]	Hybrid	BS	Optimization	
Chen and Xu [19]	Hybrid	Vehicles	RL	
Liu et al. [90]	Hybrid	Vehicles/BS	DRL	
Luo et al. [94]	Hybrid	Vehicles/BS	PSO	
Gao et al. [53]	Hybrid	Vehicles/BS	DQN	
Misra and Bera [107]	Centralized	Vehicles	ILP	✓
Wu et al. [151]	Centralized	Vehicles/BS	Optimization	✓
Kazmi et al. [77]	Hybrid	Vehicles/BS	DRL	✓

another direction, some works consider hybrid architectures, enabling to build on regional knowledge without burdening a central entity and the network's core with high message exchange [117, 41, 19, 90, 94]. A VEC architecture allows the cooperation of computational resources among vehicles and BSs for VCs creation, enabling resources to be always available. For instance, considering only vehicles, the approaches are limited to regions with high vehicle flows to maintain the high resource availability. On the other hand, considering BS as an entity that composes the VCs enables more effective management both in forming VCs and using resources, making the processes more stable and reliable [117, 90, 94, 151].

Regarding the scheduling approach, it is essential to consider decision-making techniques with low complexity to deal with the real-time requirements of different classes of applications explicitly defined by their deadlines. In this sense, in computational complexity order, some studies consider simple computing algorithms [60, 117], others consider optimization techniques [33, 41, 94, 107, 151], and some studies consider machine learning algorithms [19, 90, 77, 53]. In this sense, the techniques used should be less computationally complex, and thus their decision time does not negatively impact the system's overall efficiency. Furthermore, creating knowledge about the dynamics of computational resources through vehicular mobility is essential to estimate future resource availability. In this sense, approaches can make better decisions, relating task requirements, such as deadlines, and computational resources available in the future, reducing the task scheduling interruptions and the monetary cost employed by this process. However, only a few studies consider vehicular mobility information in their decision-making processes [107, 151, 77].

3.3 Fairness and Load Balancing

Numerous studies have created diverse strategies to optimize task scheduling in VEC systems. Also, some of these strategies try to balance the scheduling load in the VCs. For instance, Hattab et al. [60] introduced an optimized algorithm for scheduling tasks in VCs with varied resources, which mainly uses the First-Come, First-Served (FCFS) standard from the literature. The algorithm classifies tasks by completion-to-waiting time ratio, identifies those with the lowest ratio, and uses Linear Programs to solve them. It aims to minimize task completion time within VC's resources, assuming a static VC and focusing solely on a single VC. In this case, load balancing is trivial, as there is only one VC for decision-making.

Some works consider Multi-Criteria Decision Making (MCDM) methods for selecting the VC to perform the scheduling process. These MCDM methods tend to balance the selected options to insert a load balance during the decision process. Aligning with this approach, Mishra et al. [106] introduced two AHP-based resource allocation policies, named SECA and AHP-EV. Resource allocation can also be defined as task scheduling, depending on the scenario. The proposed schemes consider the network load and compute load during decision-making, aiming to minimize each task's delay. These policies differ in assigning weights to each criterion, but both have similar results. AHP-EV employs pre-set weights for computing power and network, whereas the SECA dynamically determines criteria weights.

Similarly, Pereira et al. [117] devised a mechanism known as FORESAM, which utilizes the AHP to select the optimal set of tasks for scheduling within VCs. FORESAM aims to maximize vehicular resource use, considering all task requirements during decision-making. Despite this, FORESAM employs only one VEC controller and fails to specify its placement within the VANET. Depending on the controller's position, requests may still need to traverse the network core, potentially leading to congestion and subsequent service delays.

Other works utilize multi-objective optimization methods for the task selection process that will be scheduled. Da Costa et al. [37] presented a task scheduling mechanism for VCs based on Pareto optimality, named EFESTO. EFESTO employs vehicle mobility information to estimate the resources available in each VC, thereby enabling more precise decision-making. Moreover, it uses Pareto optimality to select the optimal subset of tasks for joint minimization of deadline and processing time, which minimizes the monetary cost associated with resource usage. However, EFESTO prioritizes the VCs with more available resources without considering possible overloads that this continuous selection can cause.

Similarly, Luo et al. [94] evaluated delay and cost implications associated with task scheduling in VCs. The study established a scheduling framework that accommodates communication and computation within VCs, considering tasks with varied requirements. Consequently, a multi-objective problem was designed to minimize both delay and cost. A Particle Swarm Optimization-based scheduling algorithm was suggested to derive Pareto-optimal solutions. However, due to the bio-inspired nature of the approach, the convergence time could impact the solution's overall performance. Additionally, the authors did

not factor in vehicular mobility as a prerequisite in the scheduling process.

Ribeiro et al. [126] introduced a metaheuristic approach that models VCs as a coalition. In game theory, a coalition refers to a group of players who agree to cooperate by combining their strategies to enhance their joint payoff. In this context, the authors propose a coalition game to maximize resource utilization while dynamically balancing the load among the VCs. Firstly, the mechanism establishes a strategy based on the Shapley value to determine the sequence in which tasks are scheduled. After that, the mechanism uses a queue to schedule tasks within VCs based on values calculated. Nonetheless, the authors overlook the contextual factors in the scheduling decision process. They do not consider the implications of vehicular mobility and task deadline constraints.

Some works consider combinatorial optimization problems for task scheduling and, given the nature of the problems, load balancing is inherently applied. Wang et al. [145] proposed a task offloading algorithm for VCs, which builds on the Multi-dimensional Multiple Knapsack Problem (MMKP) concepts. It considers that each user will pay for computing tasks according to task size to maximize the total profits from computing offload from an infrastructure perspective. A modified Branch-and-Bound algorithm is proposed to obtain the ideal solution with a Greedy heuristic method to get approximate performance with lower computational overhead. However, these algorithms are computationally costly because they compute the optimal solution for each knapsack in one step and only then use this information to allocate tasks in different clouds.

Nabi et al. [108] presented a task scheduling scheme for VCs that uses Knapsack Problem concepts. This scheme solves the scheduling for a single task in polynomial time and provides a greedy solution for the same purpose. Besides, they extend the algorithm to solve the allocation problem for n tasks with lower bounds and Fractional Knapsack Problem. However, this scheme considers an offline environment in which the problem instance is entirely available and known before the simulation begins. In this sense, it requires prior knowledge of the tasks to be allocated, not operating satisfactorily in a real-time decision-making environment.

3.4 Chapter Conclusions

This chapter described literature solutions for the VC formation, task scheduling, and fairness and load balancing in VEC systems, highlighting the advantages and limitations of each solution. Also, a classification and a qualitative comparison were proposed. Based on the state-of-the-art analysis, some conclusions are taken, namely: *(i)* a VC formation solution needs to consider predicted mobility information in its 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; *(ii)* Scheduling tasks with high computational power requirements, such as traffic image processing or real-time learning, is a challenging issue in a highly dynamic vehicular environment [135]. In other words, the task scheduling mechanisms must make accurate decisions considering all the tasks' computational requirements, regardless of the

variability of these requirements in different classes of applications. Besides, scheduling mechanisms must always minimize costs for end-users; and *(iii)* the approaches prioritize scheduling results, even if fairness and load balancing do not present high levels during the process of using computational resources. In this way, it is essential to consider load balancing during the task scheduling decision process to increase fairness in using resources and not overload specific regions in the city, increasing its maintenance costs.

In this way, the next chapters will describe the solutions proposed to address these limitations. Chapter 4 presents an VC formation mechanism that uses mobility prediction to increase the lifetime of VC without overloading the network with management messages. Chapter 5 presents a task scheduling mechanism that uses mobility prediction to estimate the available resources in each VC using an RNN architecture. At last, Chapter 6 presents a strategy to increase fairness and load balancing in the use of vehicle resources without negatively impacting the system's overall efficiency.

Chapter 4

Vehicular Cloud Formation using Mobility Prediction

VEC considers a set of VCs to allow vehicular users to request resources to meet application demands [114, 22]. Specifically, VC 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.*, RSUs, 5G BSs, and RS in the Internet cloud) nodes to provide cloud services [58]. In other words, some vehicles increase their computing capacity by using VEC resources, while others lend their available resources to VC. 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 VC formation mechanisms for grouping these vehicles in VCs, where VC formation occurs by creating groups with a set of vehicles in a geographic region sharing the same preferences, such as direction, trajectory, path similarity, etc [11, 102].

The VC formation mechanisms rely on the communication infrastructures to assist the VCs formation process [17]. In this sense, vehicles under infrastructure coverage are considered members of this VC. This is because, in a macroscopic view, these VCs can be inter-communicable through infrastructures deployed in cities, allowing cooperation between them [14]. 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 VCs as vehicles traveling on multiple road segments can meet at one place [113]. Unlike distributed approaches for VC formation, we consider that vehicles do not have autonomy in defining the roles of 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 [85], and considerably reduce the number of maintenance messages that fully-distributed approaches need in their decision processes.

However, as we discussed in Chapter 2, high vehicular mobility is the main challenge for proposing efficient VC formation mechanisms since vehicle mobility causes several variations in network topology and intermittent connections [31]. For example, it may happen that the connection between the client vehicle and the vehicles processing in the VC 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 VC, causing the loss of task processing results and impacting the number of tasks attended/scheduled [124]. In this context, existing works have considered mobility information in their decision-making processes to mitigate the impact of vehicular mobility [93, 95, 156, 147]. Although mobility is considered in such works, they do not involve mobility in their optimization problems, and mobility is a trigger condition for other processes [151]. Hence, providing a mobility-aware VC formation mechanism to select the most stable vehicles to lead each VC and make managing more stable and reliable is still an open issue [88].

In this context, this chapter describes a mobility-aware mechanism called PREDATOR (mobility **PRED**iction-based **A**pproach to enhance vehicular cloud forma**T**i**O**n p**R**ocess) that enhances the VC 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 VC, thereby increasing the VC's lifetime. Vehicle stability is measured by the dwell time, representing the duration a vehicle spends within the RSU coverage area [113]. 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 VC formation process. The number of VCs 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 VCs and manage their aggregated resources. Simulation results demonstrate the superior performance of PREDATOR compared to other VC formation mechanisms. The results highlight the benefits of PREDATOR, including a 42.15 % longer VC lifetime, 45.81 % lower number of VC 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.

4.1 PREDATOR

As discussed above, vehicular mobility directly impacts the approaches to forming VCs due to the dynamic nature of moving vehicles. VC 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 VC formation process more complex.

A solution aware of vehicular mobility can overcome the challenges of high mobility for VC 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 VCs are formed, adaptive algorithms and load-balancing strategies can be implemented to optimize the task scheduling among vehicles grouped in these VCs, 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 VC. Thus, vehicular mobility prediction information can be leveraged as an advantage for forming more robust and resilient VCs.

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 VCs assisted by RSUs. PREDATOR considers a mobility prediction model for selecting the most stable vehicles to coordinate each VC. 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.

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 VC 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 GPS.
- A VC starts when an RSU selects a new VCloudHead, and this VCloudHead receives the warning message from an RSU.

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 VCs as vehicles traveling on multiple road segments can meet at one place [113]. 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 VC 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 VC directly, so VCs' 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 VC by an RSU. The second role is VCloudMember, the vehicles that will share their computing resources on the network through

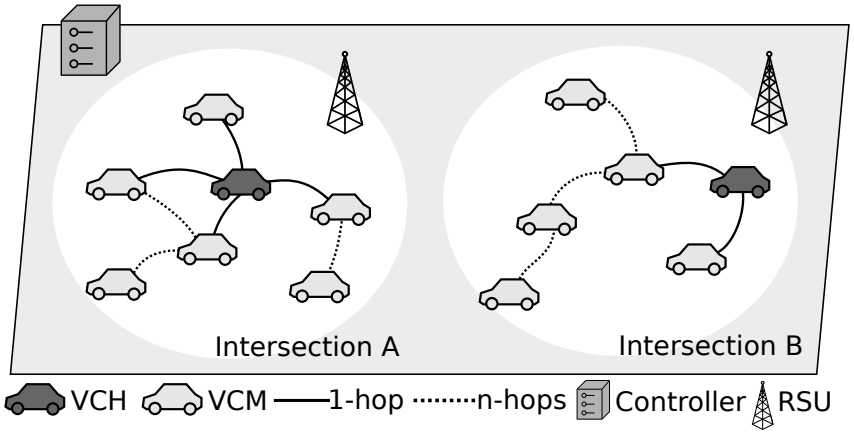


Figure 4.1: System architecture (VCH: Vehicular Cloud Head (VCloudHead); VCM: Vehicular Cloud Member (VCloudMember).

a VC. Therefore, when a vehicle is associated with an RSU, it automatically becomes part of the VC assisted by that RSU. The primary function of an RSU is to select the vehicle leading this VC. Considering the communication range of RSUs is greater than that of vehicles, the lead vehicle can be n hops away from the other VC members. After the RSU selects and informs the VCloudHead of this VC, 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 VCs. In this case, the controller does not actuate directly in the VC formation process. It is responsible only for information aggregation sent by RSUs after ending the VC formation process. So, VANET applications that need computational resources can request these resources on the network, and the controller decides which VC runs the tasks of these applications [33].

Figure 4.1 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 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 VC (VCloudHead). Besides, it is considered a scenario where the RSU has a higher communication range than vehicles. This way, V2V communication within the VC can occur over more than one hop.

In summary, PREDATOR considers two phases in the VC 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 received 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 VC identification, the VCloudHead identification, and the num-

ber of resources available in that VC. 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.

4.2 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 VC. 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 [28, 140]. Also, the forecast granularity can be defined based on the VC formation intervals.

The vehicular mobility pattern makes it possible to model a mathematical system to predict the future geographic positions of nodes [7]. 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 VC for location-based services. However, an RSU maintains a macroscopic view of the VC 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 VC, and this vehicle is called VCloudHead. However, this VCloudHead definition is one of the critical points in the VC 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.

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 received. Otherwise, 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 VC, applying the Equation (4.1) 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 VC.

$$S_{u_i} = 1 - \left(\alpha \times \bar{A} + \beta \times \bar{B}\right) \tag{4.1}$$

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 (4.2). 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.

$$\bar{A} = \left(\frac{dist}{R_{max}}\right) \tag{4.2}$$

On the other hand, portion $\bar{B}$ of the Equation (4.1) 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 (4.3). $DwellTime$ is the dwell time calculated by checking the predicted positions.

$$\bar{B} = \left(\frac{1}{DwellTime}\right) \tag{4.3}$$

To illustrate the calculation of Equation (4.1), 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 VC 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 4.2 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 VC. 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 VC 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 VC formation stage, the controller has no role other than to keep the information of each RSU for eventual resource orchestrations.

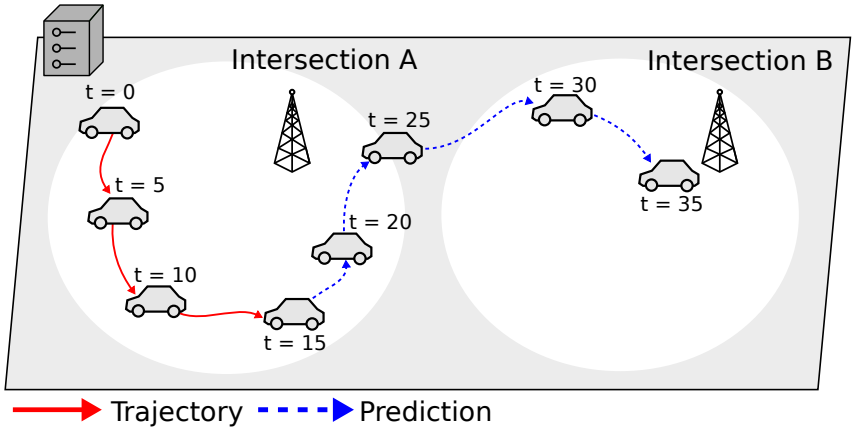


Figure 4.2: PREDATOR's microscopic vision.

In this way, Algorithm 1 shows a pseudocode of PREDATOR. The set of vehicles $N_k \subseteq U$ 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 VC 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 VC (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 VC is incremented in a 1-time unit (Lines 11 and 13). With the distance *dist* and dwell time *DwellTime* calculated, the Equation (4.1) 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 VC (Lines 17 and 18).

Algorithm 1: PREDATOR mechanism overview

```
   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 (4.1)
           ▷ 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)
```

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 VC's member vehicles; and (iii) VCloudHead, which represents the leader of the

VC. 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 VC (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 VC (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).

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())
```

Additionally, to illustrate the entire message exchange process on the network, some actions must be taken when the VCloudHead informs its neighbors about its role in the VC. 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).

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)
```

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 (4.4). 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.

$$\begin{aligned} l_t = \theta_0 + \Phi_1 l_{t-1} + \Phi_2 l_{t-2} + \cdots + \Phi_p l_{t-p} \\ -\theta_1 \mu_{t-1} - \theta_2 \mu_{t-2} - \cdots - \theta_p \mu_{t-p} \end{aligned} \tag{4.4}$$

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 [136]. Therefore, the model is trained for each vehicle and its respective geographic coordinates [28].

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 $\mathcal{O}(m)$ [57], 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 is $\mathcal{O}(n \times m)$. In summary, PREDATOR is a mechanism that operates in polynomial time and is efficient enough to operate in practical scenarios.

4.3 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.

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 [134], 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 [3] with $1\,\text{km}^2$ area, as shown in Figure 4.3(a). In this case, the traffic behavior is configured to use the Krauss car-following model for its accuracy and simplicity [80]. Second, we consider the realistic mobility trace of the Luxembourg city (LuST) [25, 113], as shown in Figure 4.3(b). We selected a $2\,\text{km}^2$ 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 [46]. 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 [31]. 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 VC formation process starts. We setup 5 RSUs at the main intersections in all scenarios, as indicated by red points in Figures 4.3(a) and 4.3(b).

(a) Grid scenario

(b) LuST scenario [25]

Figure 4.3: Simulation scenarios considered.

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 [109]. At each VC 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 [43, 91]. 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 4.1.

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 [157, 73], CENTROID [58], and SPATIAL [33, 115]. Additionally, we compare the performance of PREDATOR with its previous version, called NEMESIS [32]. 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 ($24 \times 2.67\,\text{GHz}$) with Linux architecture x86-64.

2. VC formation assessment:

   - *Lifetime* is the accounting of how many time units the VCs lasted. A high VC 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 VC. The high number of leader changes implies that vehicles selected as VCloudHead were not the most stable.

3. Scalability assessment:

   - *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 [31].
   - *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 VCs' stability in using the aggregated computing resources.

4.4 Results

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

Mobility prediction Assessment Perspective

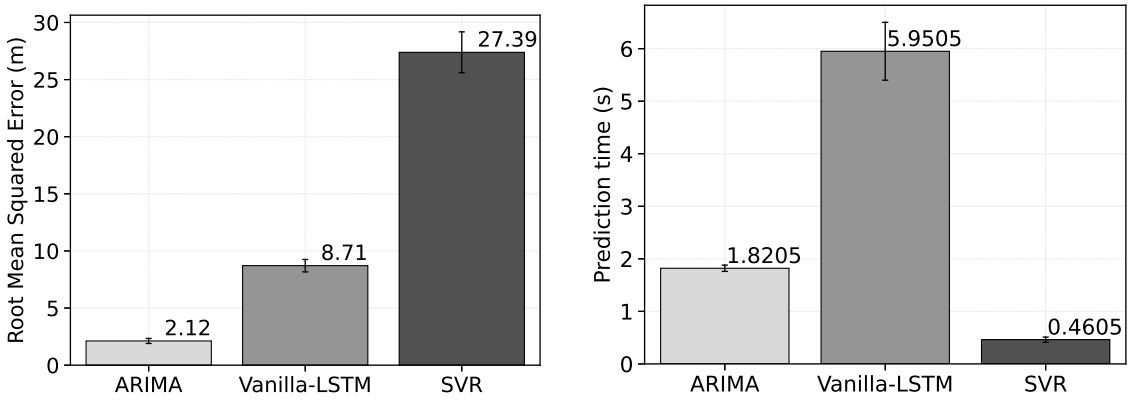
Figure 4.4(a) presents the RMSE results obtained with the considered models, namely ARIMA, Unmodified Long Short Term Memory (Vanilla-LSTM), and Support Vector Regression (SVR), in the prediction data. This assessment calculates the average RMSE among all vehicles in the scenario. However, please note that Figure 4.4(a) exemplifies the RMSE obtained for only one vehicle. The figure shows that the predictions provided

Table 4.1: Simulation parameters for Vehicular Cloud (VC) formation assessments.

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
VC formation interval	5 s
Number of RSUs	5
Number of vehicles	100
α, β	0.3, 0.7
ARIMA (p, d, q)	(2,2,1)
Vehicle speed limits (Grid)	15 m/s
Simulation area (Manhattan Grid)	1 km^2
Simulation time (Manhattan Grid)	200 s
Vehicle speed limits (Luxembourg)	From the LuST [25]
Simulation area (Luxembourg)	2 km^2
Simulation time (Luxembourg)	500 s

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 4.4(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 4.4(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 VC 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 4.4: Prediction results on a single vehicle's data.

VC formation Assessment Perspective

Figure 4.6 shows the results obtained for VC metrics, such as Lifetime and Leader changes, considering the two different mobility traces. First, Figures 4.6(a) and 4.6(c) show the VCs' lifetime results for both scenarios. It can be noted that VCs 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 VC 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 VC lifetime metric over NEMESIS, SPATIAL, CENTROID, and DEGREE, respectively.

Complementarily to the experiment shown in Figure 4.6(a) and to reinforce the impact of vehicular mobility in the context of VC 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 4.5 displays the VCs' lifetime using all VC 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.

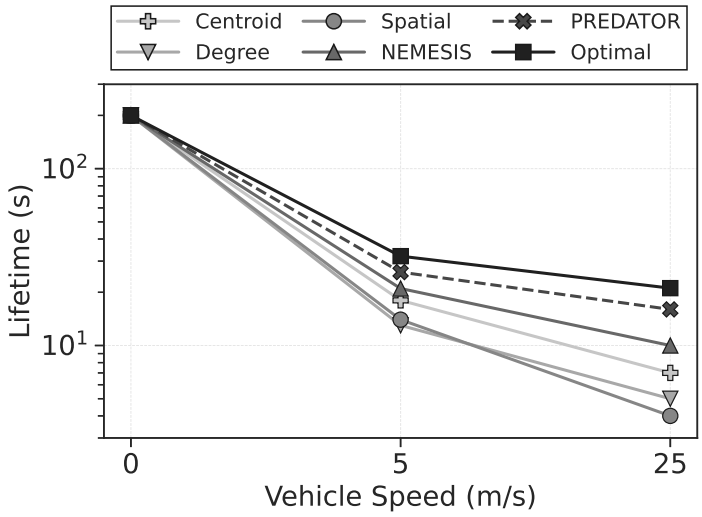
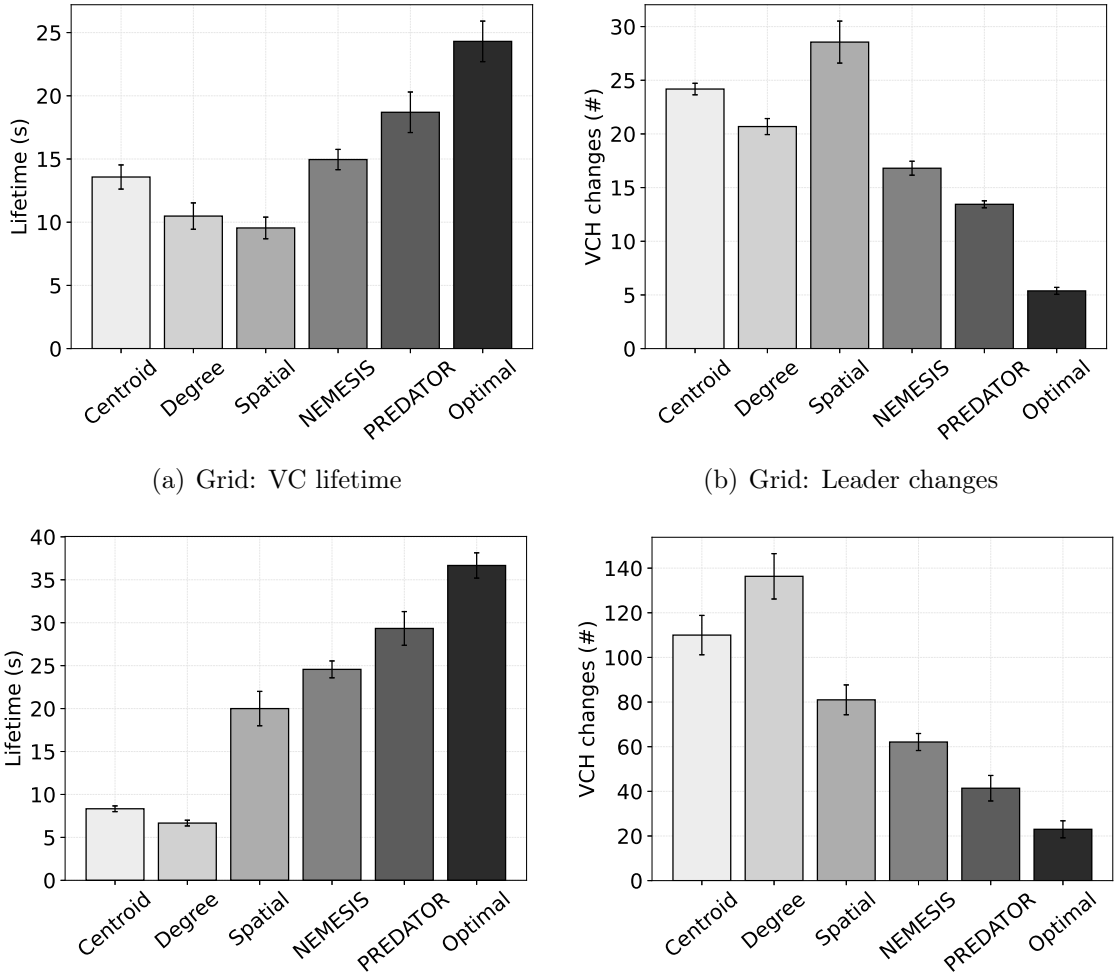


Figure 4.5: An example of how vehicular mobility impacts all VC formation solutions

In addition, Figures 4.6(b) and 4.6(d) show the general stability of the created VCs. It can be observed that PREDATOR spends more time in the RSU coverage, which implies greater stability in VC 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.



(a) Grid: VC lifetime

(b) Grid: Leader changes

(c) LuST: VC lifetime

(d) LuST: Leader changes

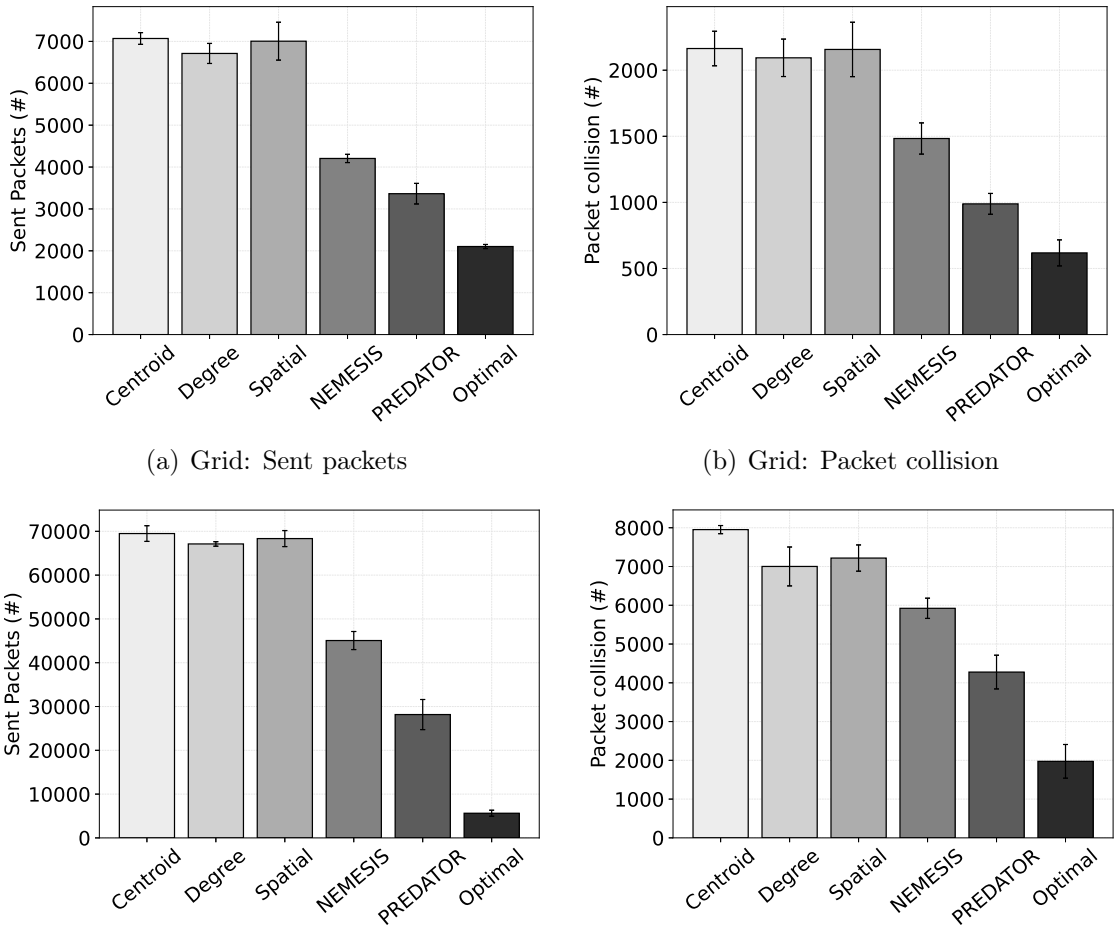
Figure 4.6: VC formation results considering different mobility traces.

Scalability Assessment Perspective

Regarding scalability metrics, Figures 4.7(a) and 4.7(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 VC 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 4.6(b) and 4.6(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 4.7(b) and 4.7(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 [31]. 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.



(a) Grid: Sent packets

(b) Grid: Packet collision

(c) LuST: Sent packets

(d) LuST: Packet collision

Figure 4.7: Networks metrics considering different mobility traces.

Scheduling Assessment Perspective

Considering the created VCs' 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 VC 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 VCs 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.

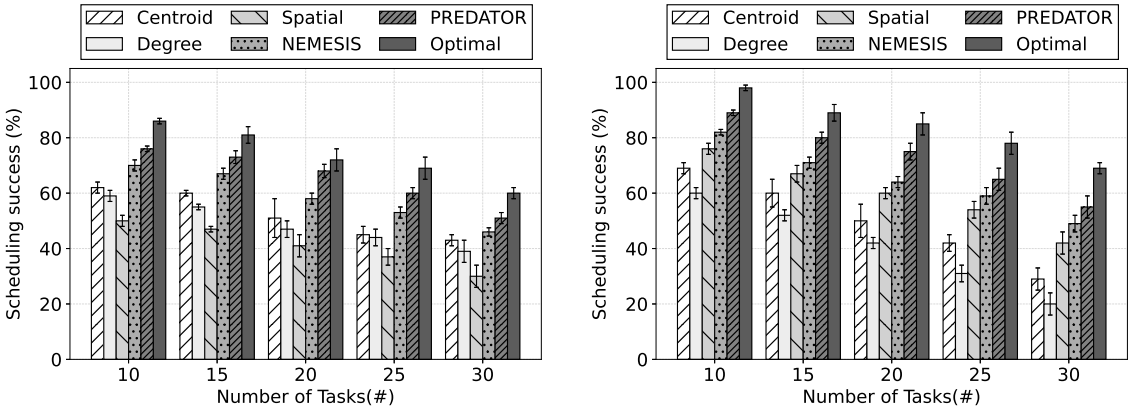
Algorithm 4 presents a pseudocode of what happens in the VCloudHead vehicle when it receives the schedule message from the controller. VCloudHead starts a timer to control when the task completes its execution (Line 1). Therefore, VCloudHead adds a time unit to the timer (Line 3). As the VC formation process is independent of the scheduling process, verifying if the current vehicle is still VCloudHead of this VC 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 VC (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 VC 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 VC formation presented in the previous section. However, to assess the efficiency of VCs, we consider applications with processing times ranging from 5 s to 10 s. As the objective is to evaluate the efficiency of VCs about stability, other metrics besides the time of tasks and VCs were not considered. The number of tasks was varied by 10, 15, 20, 25, and 30.



(a) Task scheduling in the Grid scenario

(b) Task scheduling in the LuST scenario

Figure 4.8: Task scheduling results considering computational tasks with different requirements.

Figure 4.8 presents the results for task scheduling assessment. We can see that the results obtained corroborate the results obtained in evaluating the VCs formation about the average lifetime of the VCs. That is, the longer the lifetime of the VC, the greater the chance of it serving a greater number of tasks in the system. Not only that, but VC 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 VCs 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. PREDATOR is 89.13 % and 86.87 % closer to the optimal approach in the Grid and LuST scenarios, respectively.

4.5 Chapter Conclusions

The VEC paradigm emerged and enabled vehicles to actively act in the consumption and supply of computational power to the 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 VC formation. In this context, we introduced a mobility-aware VC 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 VCs with the assurance that their processes will be successfully completed within the required time frame. The results demonstrate that PREDATOR can increase the average lifetime of VCs, 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.

Chapter 5

Mobility- and Deadline-aware Task Scheduling Mechanism

In the VEC scenario, each network controller, called VEC controller, is responsible for coordinating a predefined city region, such as a neighborhood, for forming the VCs based on the regional knowledge about the vehicle and infrastructure nodes through the V2I communication. Hence, the VEC controller can be responsible for two main processes, namely, VC formation and task scheduling [76, 111, 33]. The VC formation process concerns an efficient grouping and management of computational resources available by the network entities to form a VC [14, 102]. On the other hand, the task scheduling process concerns the efficient use VC resources, *i.e.*, the tasks will be scheduled to be processed in a given VC. Therefore, these two processes are crucial to group computational resources and provide cloud services closer to vehicular users. Advance the state-of-the-art in the task scheduling process is essential to the success of connected autonomous vehicle ecosystems. New approaches to efficiently select and decide where and when tasks will be scheduled in dynamic and mobile vehicular environments are required. At least three key issues need to be addressed for developing an efficient task scheduling mechanism to meet users' demands in VEC scenarios, namely *(i) Vehicular Mobility*; *(ii) Deadline Constraints*; and *(iii) Monetary Costs*.

The *Vehicular Mobility* is one of the main challenging factors for the efficient performance of vehicular scenarios with support to VEC applications since vehicle mobility causes several changes in network topology and (including intermittent connections) [31]. For instance, vehicular mobility causes an unexpected disconnection between VC members and the VEC controller, leading to a fluctuation in terms of the number of available resources in the VC and/or causing the task processing results loss, impacting the system efficiency. Hence, the mobility of vehicles has a high impact in the number of tasks attended/scheduled [124]. In this sense, it is essential to consider vehicular mobility information to estimate vehicles' location, which can be used to estimate the future resource availability in each VC. This aims in selecting stable VCs to run the tasks, where VC's stability refers to its low rate of resource variability over time. However, how to predict these vehicular dynamics accurately and explore their spatiotemporal correlation is still an open issue [45]. In this sense, RNN can use in vehicular mobility estimates due to their high accuracy rates in general form predictions [44, 67, 15].

The *Deadline Constraints* represent the time that tasks can wait to be served. In other words, it is important that the obtained result must be returned to the requesting entity before a hard deadline. Otherwise, this result becomes useless [148, 19]. Therefore, the tasks' deadline constraints can be better observed and considered from a resource availability estimation in the VCs, making it possible to infer the processing time and the service probability to increase the scheduled and completed tasks rate. Also, the system latency is minimized by prioritizing tasks with less processing time. The system latency represents the waiting time and the time that a task will remain running in a given processing configuration.

Finally, *Monetary Cost* refers to the price of using computational resources, such as the pay-as-you-go basis commonly used in cloud computing IaaS architectures [143]. For instance, the monetary cost can be minimized by knowing the task processing time, making using computational resources less costly for the end users. In addition to task deadline constraints, the tasks often have high processing requirements. Scheduling tasks with high computational power requirements, such as traffic image processing or real-time learning, is a challenging issue in a highly dynamic vehicular environment [135]. In other words, the task scheduling mechanisms must make accurate decisions considering all the tasks' computational requirements, regardless of the variability of these requirements in different classes of applications. Besides, scheduling mechanisms must always minimize costs for end-users.

To the best of our knowledge, a unique solution that considers mobility-awareness, deadline constraints, and monetary costs for decision-making in a task scheduling process is still an open issue and remains a challenge. Thus, this chapter proposes MARINA (**M**obility and de**A**dline-awa**R**e task schedul**I**ng mecha**N**ism for vehicul**A**r edge computing) to maximize the number of tasks scheduled while minimizing the monetary costs of utilizing VC's resources. The MARINA runs on VEC controllers to coordinate the task scheduling in multiple VCs. In MARINA's operation, a vehicle without enough computational resources to run a specific task can forward this task to be processed somewhere in the vehicle ecosystem. In this sense, MARINA selects a set of tasks to be scheduled in real-time in each available VC based on the Pareto Optimality and BCP. Pareto optimality allows the joint minimization between the deadline and estimated processing time. Besides, the BCP makes it possible to find the best fit for the minimization provided by Pareto, always seeking to maximize the number of scheduled tasks. MARINA also considers an LSTM to predict resource availability in each VC based on vehicular mobility information. Hence, MARINA prioritizes scheduling tasks in VCs with more available resources to maximize the fulfillment of demands in as few rounds as possible. Simulation results show that, compared to state-of-the-art solutions, MARINA can schedule more tasks while minimizing monetary cost and system latency.

The main contributions of this chapter are the following:

- The use of vehicular mobility information and an LSTM architecture to predict the available resources in each VC with high accuracy.
- An efficient task scheduling mechanism that considers the task deadline, mobility-awareness, and monetary costs in its decision-making process. Thus, it schedules a

Table 5.1: Summary of key notations for the task scheduling study.

Symbol	Description	Symbol	Description
U	set of vehicles	u_i	a vehicle $\in U$
B	set of BSs	b_y	a BS $\in B$
V	set of VCs	v_j	a VC $\in V$
T	set of tasks	t_l	a task $\in T$
C	set of controllers	ω_{u_i}	vehicle's CPU-cycle
n	# of tasks	ϕ_{u_i}	vehicle's storage
x	# of vehicles	ω_{b_y}	BS's CPU-cycle
m	# of BSs/VCs	ϕ_{b_y}	BS's storage
e	# of controllers	Ω_j	VC's total CPU-cycle
l	index of task	Ψ_j	VC's shared CPU-cycle
i	index of vehicle	Φ_j	VC's total storage
j	index of VC	D_l^t	task deadline
o	index of controller	$d_l^t j$	task proc. time in VC j
y	index of BS	$\mathcal{C}_l$	task monetary cost
$\mathcal{Q}$	waiting queue	w_l^t	task CPU-cycle required
$\mathcal{R}$	run queue	$Price(t_l)$	resource price used by t_l
R	number of regions	$\mathcal{P}$	pareto set

high number of tasks without increasing the monetary cost of using VC's computational resources.

- A combined approach that leverages low-computational complexity techniques to minimize the monetary cost and maximize the number of scheduled tasks.
- A detailed performance evaluation with a realistic mobility trace, showing the benefits of vehicular mobility information for the task scheduling process compared to other state-of-the-art approaches.

5.1 MARINA

This section introduces the MARINA task scheduling in VEC scenarios, which considers available resources, tasks' deadline constraints, and vehicular mobility for decision-making. We first overview the designed system and then illustrate the VCs formation process, mobility prediction approach, and monetary cost model. Finally, we formulate the problem. Table 5.1 summarizes the key notations used in this work.

Overview

Figure 5.1 shows the system architecture composed of vehicles, BSs, VEC controllers, and an RS in the Internet cloud. We consider a scenario composed of x moving vehicles, denoted as $u_i \in U = \{u_1, u_2, \ldots, u_x\}$. Also, we consider a set of BSs deployed in the city, denoted as $b_y \in B = \{b_1, b_2, \ldots, b_m\}$, where m is the total number of BSs. Each BS

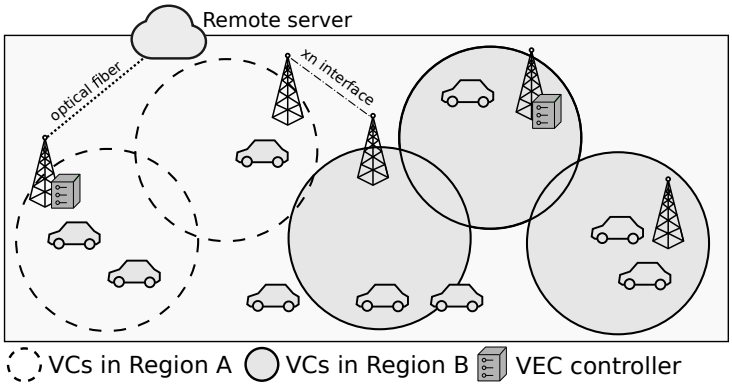


Figure 5.1: The architecture employed by MARINA, presenting its main components; in particular, Vehicular Clouds (VCs) and Vehicular Edge Computing (VEC) controllers.

has wired communication (*e.g.*, optical fiber) with the RS and could provide processing power and storage capacity on the network edge, decreasing response time for some applications [90]. Also, each BS on the 5G network has an *Xn* interface, which allows the information exchange between neighboring BSs and assists in the handover process [152]. Each vehicle has an OBU that allows communication with the neighbor devices through V2X communication. For instance, vehicles can associate with a BS and communicate with an RS to access the Internet, request resources for data processing, and share their resources.

We used a BS-UE (Base Station-User Equipment) method based on the maximum SINR for this association process. In this case, a vehicle will associate with a BS which provides the Max-SINR [142]. The Max-SINR ensures that the vehicle is associated with only one BS [86]. After the association between vehicle and BS, the BS sends the vehicle' information to RS.

In this scenario, we consider the city divided into R regions (neighborhoods), and each region has at least one BS. We also consider that each VEC controller covers a given region R, and it is randomly deployed in a given BS of such region. It is important to mention that VEC controllers are randomly deployed in the scenario since this is not our research focus. We denote the set of VEC controllers as $c_o \in C = \{c_1, c_2, \ldots, c_e\}$, where $e = |R|$ is the total number of controllers, and directly related to the number of regions R. Each controller is responsible for managing the BSs in its city region, where this management includes the VC formation and task scheduling processes.

In the VC formation process, the VEC controller requests to the RS the information about BSs and vehicles to build their regional knowledge. In this case, the Publish/Subscribe paradigm is considered to obtain the relevant information without inserting unwanted traffic on the network. The VEC controller subscribes to BSs' updates in its region. Thus, the RS plays the role of Publisher, and the regional VEC controllers play the role of Subscribers. Based on the information about the BSs, the VEC controllers could starts the VCs formation process. In this sense, VCs can be classified in different regions according to BSs' positions in the city. Considering that the number of VCs is the same number of BSs in the scenario, we can denote a set of VCs by $v_j \in V = \{v_1, v_2, \ldots, v_m\}$, where m is the total number of VCs. A VC consists of a set of nodes (*i.e.*, vehicles and

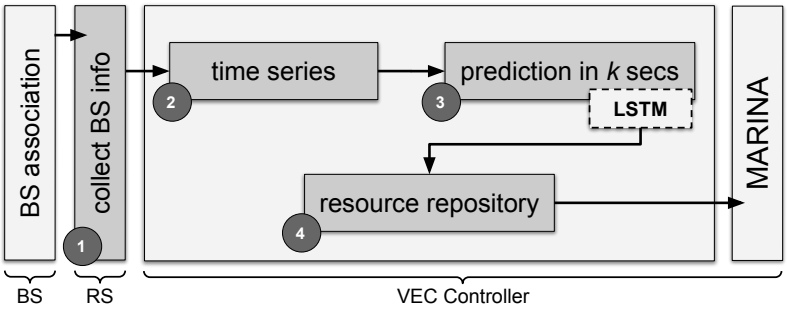


Figure 5.2: Vehicular Cloud (VC) formation process – from Base Station (BS) association to scheduling – based on a prediction approach using Long Short-Term Memory (LSTM).

BS) that can share two types of computational resources ω and ϕ, namely CPU-cycle frequency (processing power) and storage capacity, respectively.

In this context, ω_{u_i} denotes the vehicle's CPU-cycle frequency, ϕ_{u_i} denotes the vehicle's storage capacity, ω_{b_y} denotes the BS's CPU-cycle frequency, and ϕ_{b_y} denotes the BS's storage capacity. Therefore, each VC is represented by a tuple $\{id, resources_{vehicles}, resources_{bs}, resources_{total}\}$, where id is the VC's unique identification, $resources_{vehicles}$ is the total resources of vehicles (ω_{u_i} and ϕ_{u_i}), $resources_{bs}$ is the total resources of BS (ω_{b_y} and ϕ_{b_y}), and $resources_{total}$ is the sum of resources in the VC (Ω_j and Φ_j), calculated as

$$\Omega_j = \sum_{i=1}^{x} \omega_{u_i} + \sum_{y=1}^{m} \omega_{b_y}, \quad \text{if } u_i, b_y \in v_j, \tag{5.1}$$

$$\Phi_j = \sum_{i=1}^{x} \phi_{u_i} + \sum_{y=1}^{m} \phi_{b_y}, \quad \text{if } u_i, b_y \in v_j. \tag{5.2}$$

In short, the total amount of processing power resources Ω_j and Φ_j of each VC v_j is the sum of the shared resources from each vehicle u_i and BS b_y belonging to a given VC v_j.

Figure 5.2 depicts the VCs formation process based on a spatial layer to add temporal information about vehicle mobility and their computational resources. We assume that BS is aware of vehicle mobility to add the temporal layer in the VC formation, which is possible to take advantage of beacons already exchanged by vehicles to obtain the vehicles' information, avoiding extra overhead. Specifically, each vehicle transmits periodic beacons containing its id, computational resources, speed, positioning, and route. In this sense, the BS receives such information and forwards it to the RS. Based on the information requested by the VEC controller to the RS, the VC formation process begins with mapping which BS holds which vehicles in its coverage (Label 1 in Figure 5.2). Given the history of associations in the BSs, it is possible to decompose this information into time series representing the time variation of vehicles in each VC (Label 2 in Figure 5.2). Spatiotemporal information on the dynamics of resources in each VC is created and stored in the regional VEC controller. Considering the importance of estimating the resources available in a VC to make task scheduling decisions with high precision, an LSTM-based prediction model is used in this step. With this, we can define a time window and estimate the resources available in each VC (Label 3 in Figure 5.2). Finally, a resource

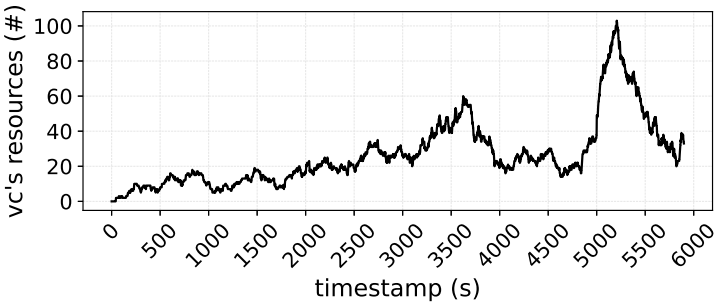


Figure 5.3: Time series created by the system for a Vehicular Cloud (VC).

repository stores the predicted information, and the MARINA consults such spatiotemporal information for decision-making (Label 4 in Figure 5.2). Therefore, it is possible to estimate resources available in each VC in k time slots by considering the spatiotemporal information. In this sense, we can represent the resources available at VC in each $k \in K$ by Ω_{jk} and Φ_{jk}.

Mobility prediction model

We consider a mobility prediction process to add a temporal layer to the spatial information about vehicle mobility and their computational resources, which is essential in task scheduling scenarios for VCs environments. In this sense, we need to estimate the vehicles' dwell time in each VC, enabling to obtain information on the VCs' computational capacity in a predefined time window.

Several works consider Markov models, Extended Kalman Filter (EKF), SVR, and Autoregressive Integrated Moving Average (ARIMA) models to predict vehicular mobility [137, 122]. However, neural networks have gained increasing attention from academia and industrial groups for the accurate predictions offered by their various models. In this scenario, an RNN is a deep learning approach that extends the traditional feed-forward networks with internal cycles [83]. These internal cycles allow tracking information sequences to create spatiotemporal knowledge through current and past inputs. LSTM is an advanced version of the RNN architecture developed to model chronological sequences and their long-term dependencies with greater precision [56].

In this sense, to employ an RNN in MARINA, we need to build a dataset containing resources available information in each VC. The model learns from past dynamics and accurately estimates the available resources in a given time window. To this end, the information about user association in a given BS is stored in the RS. In this way, when the VEC controller starts the VC formation process, it is easier to aggregate and create each VC's resources time series. Let $Z = \{z_0, z_1, \ldots, z_k\}$ be a vector that represents the dataset, in which each element consist of a tuple $z_k = \{timestamp, resources\}$ representing a simulation step and resources available in this step, for each VC, as shown in Figure 5.3.

In summary, the predictor model is defined as $f = \Upsilon \circ \Theta$, where $\circ$ indicates applying function Υ on function Θ's output. The feature learning machine $\Theta(\cdot)$, which converts inputs into features, is utilized to process the input data first. After that, the next step involves the representation function $\Upsilon(\cdot)$, which maps features into a prediction [44]. In

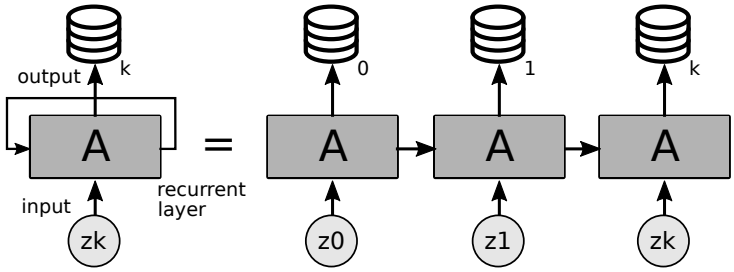


Figure 5.4: Recurrent Neural Network (RNN) employed by MARINA.

this sense, the prediction process of the next available resources for each VC is given according to

$$z_{\Omega_j}^{k+1} = \Upsilon_k \circ \Theta_k(z_{\Omega_j}^k). \tag{5.3}$$

where $z_{\Omega_j}^{k+1}$ represents the next available resources of the VC j considering its previous observations $z_{\Omega_j}^k$.

Figure 5.4 shows how the RNN employed by MARINA works. In summary, the RNN receives an input z_k representing a set of tuples, and for each VC, it employs an LSTM as a recurrent layer. The dynamic employed by LSTM is to store past information in long-term memory to explore the internal relationships between each prediction. The information persists across the network and is used in comparisons to improve prediction estimates [63]. Finally, the output represents the future resources available in each VC given a time window k.

5.2 Problem Definition

Each task $t_l \in T = \{t_1, t_2, \ldots, t_n\}$ is denoted by a tuple $\{id_l^t, s_l^t, w_l^t, D_l^t\}$ where id_l^t means the unique task identification, s_l^t denotes the size of task input data, w_l^t is the numbers of CPU cycles required to complete the task, and D_l^t is the deadline constraint. In this way, the scheduling mechanism aims to optimize scheduling a set of tasks $T' \subset T$ to be processed in the available VCs without increasing monetary costs and scheduling as many tasks as possible.

The computational resources of the same VC are shared among different tasks scheduled in that cloud. Thus, Ω_j for calculating the computation delay for a given task must be updated according to the degree of sharing of that resource within the VC, represented by Ψ_j. The value of Ω_j is divided by the number of tasks $|T_j'|$ that have been scheduled for this VC, according to

$$\Psi_j = \frac{\Omega_j}{|T_j'|}. \tag{5.4}$$

According to the literature, all processes that add delay must be considered to compute the computing delay for a task scheduling in VC, [148, 138, 19]. For example, the transmission delay of the resource request, the scheduling delay between BS and VC, the task computation delay in the VC, and the entire reverse path until the requesting obtains the result of its task. However, this article only considers the task computation delay in the VC. This is because this metric simplifies the understanding of the efficiency

of task scheduling solutions, given that the entire network infrastructure is the same for all approaches. The computation delay d_{lj}^t can be obtained based on the required CPU cycle w_l^t divided by the CPU-cycle Ψ_j of the server (VC), according to

$$d_{lj}^t = \frac{w_l^t}{\Psi_j}, \quad \forall\ t_l \in v_j. \tag{5.5}$$

As noted, each task has deadline constraints in its configuration, represented by D_l^t. Therefore, scheduling solutions must consider this metric to ensure that these restrictions are respected and tasks are successfully processed in the VCs. In this scenario, if $d_{lj}^t \leq D_l^t$, the task was successfully scheduled and executed in the v_j VC. Also, when a task is scheduled and starts to be processed, there is a cost associated with this execution. The monetary cost is modeled as

$$\mathcal{C}_l = d_{lj}^t \times \left(w_l^t \times Price(t_l)\right), \tag{5.6}$$

where d_{lj}^t is the t_l processing time in $v_j \in V$ and w_l^t is its CPU cycles required. $Price(t_l)$ is the resource price, calculated as

$$Price(t_l) = \begin{cases} 11.444 & \text{if } t_l \text{ uses } b_y\text{'s proc. resources,} \\ 5.016 & \text{if } t_l \text{ uses } u_i\text{'s proc. resources.} \end{cases} \tag{5.7}$$

These costs are based on instances with GPU capacity available on Amazon EC2[1], such as *g4ad* and *g3* for BS and vehicle, respectively.

When a task arrives in the system, the VEC controller must select the best VC to process this task. This selection must consider the monetary cost of using VC's resources. In this case, we must first apply a Pareto optimization for a joint minimization between processing time and task deadline. These metrics are directly related to monetary costs. Thus, the Pareto set allows us to find, among the queued tasks in the system, a set $\mathcal{P}$ that jointly minimizes the processing time and the deadline. With the Pareto set $\mathcal{P}$ defined, which has the tasks that imply a lower monetary cost, we can select from this set the tasks that will be processed in the VCs, represented by T'.

To coordinate this selection, we can reduce this problem as a Bin Covering Problem (BCP). BCP solves the items' packaging problem with different weights in a finite set of bins. Given a set of items, the BCP decides how many items can be stored in the same bin. The algorithm aims to maximize the number of items stored. However, given that BCP is a combinatorial NP-hard problem, we use an approximation heuristic First-Fit Decreasing (FFD) to find a solution to our problem instances in polynomial time [130]. With the FFD heuristic, they are sorted in non-increasing order of sizes before placing the items in bins. Each item attempts to be placed in the first bin that can accommodate this item. If no bin is found, a new bin is observed, and the item is put in this new bin. FFD can be implemented to have a running time of at most $\mathcal{O}(n \log n)$, where n is the number of items (tasks).

In this context, we formulate a task scheduling problem by maximizing the number of

[1] `https://aws.amazon.com/ec2/dedicated-hosts/pricing/`

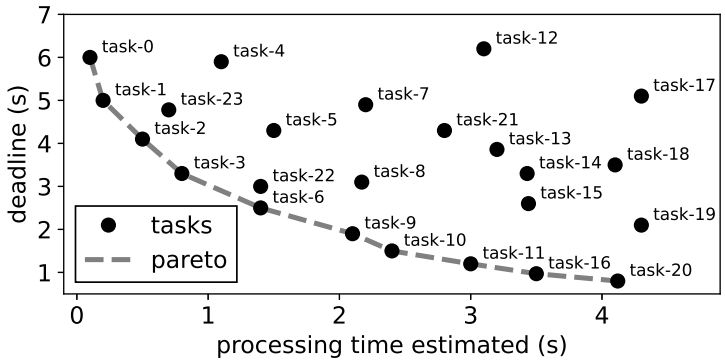


Figure 5.5: Pareto set example with 24 tasks.

tasks served while minimizing the cost monetary of VC's resources used in this process. It should be noted that minimizing the monetary cost is provided by the step that finds the Pareto set $\mathcal{P}$. Furthermore, maximization is performed by selecting tasks in the given Pareto set. In this way, we can define the problem as follows:

$$\text{P1} : \text{maximize} \sum_{l=1}^{|\mathcal{P}|} t_l, \qquad l \in \mathcal{P}, \tag{5.8}$$

$$\text{subject to } d_{lj}^t \leq D_l^t, \qquad j \in V,\ l \in \mathcal{P}, \tag{5.9}$$

$$\sum_{l=1}^{|\mathcal{P}|} s_l^t \leq \Phi_{jk}, \qquad k \in K,\ j \in V,\ l \in \mathcal{P}, \tag{5.10}$$

$$\sum_{l=1}^{|\mathcal{P}|} w_l^t \leq \Omega_{jk}, \qquad k \in K,\ j \in V,\ l \in \mathcal{P}, \tag{5.11}$$

The constraint (5.9) guarantees that the task deadline is respected, avoiding rescheduling. The constraints (5.10) and (5.11) ensures that VCs' storage and processing limits are respected during the k required processing time intervals.

MARINA's operations

In short, MARINA first searches for the Pareto set $\mathcal{P}$ using the joint minimization of task processing times and task deadlines as a criterion, creating a vector for each criterion (processing time and deadline). Therefore, the vectors are arranged in a 2-dimensional plane, and the Pareto set is found. Figure 5.5 presents an example of searching for the Pareto set in a queue with 24 tasks. We can obtain a Pareto solution set in 2-dimensional in polynomial time $\mathcal{O}(n \log n)$ [10]. This step ensures that the tasks selected to be verified with BCP already have joint minimization of estimated processing times and deadline constraints. In this case, this directly impacts the overall monetary cost.

As discussed earlier, the VEC environment is highly dynamic due to vehicular mobility. Therefore, many works model VEC environments as an M/M/1 queue to bring the system's dynamics closer to the real world [148, 131]. However, in realistic scenarios,

several entities can process existing requests. Also, it is essential to consider that the service time of the queued tasks is defined by the task requirements, such as size, required number of CPU cycles, and deadline constraints [148]. Thus, we consider that our VEC environment works as an M/G/z queue. The tasks arrive in the system following a Poisson process (M). The service time of the tasks is defined by their characteristics (that is, it follows a general distribution G). Finally, the system has more than one server (z) to attend to requests, which are the various VCs formed in the scenario.

When tasks arrive in the system, they are queued. Traditional approaches schedule tasks based on the organization in that queue. However, as the processing time and deadline of the tasks are crucial factors, MARINA prioritizes the minimization of these aspects in its decision-making process. That is, if a scheduling choice returns a lower processing time than another, the use of resources will also be minimized. In the same way, the monetary cost associated with the lower processing time will also be minimized. However, to perform this selection, MARINA needs to know the processing time of all queued tasks. In this step, Equation (5.5) estimates the processing time since it is unknown how many tasks will be scheduled in this VC.

Algorithm 5 shows how our VEC system works over time. Initially, the data structures that store the tasks in the system, the tasks in execution, and the VCs of the scenario are created (Lines 1 and 2). In a dynamic environment, tasks are generated independently, following an arrival rate that can be defined after observing the system's behavior. Thus, each time slot $k \in K$ and following a Poisson process, a set of tasks arrives in the system to be executed (Line 3). The task set is queued and goes on standby to be scheduled and executed (Line 4). For the task scheduling process to occur, it is necessary to know the VCs available in the scenario. Thus, if k is equal to the VC formation interval, the formation process occurs, and the VC information is maintained until the next interval (Lines 5 and 6). If the $\mathcal{Q}$ queue is not empty (Line 7), the queued tasks and the set of VCs are passed to the task scheduling mechanism (Lines 8 and 9) presented in Algorithm 6. After the scheduling process, verification is performed at each time slot k if the tasks in the $\mathcal{R}$ queue (run queue) reached their processing time in each VC (Lines 10 and 11). If the task has completed its execution, the monetary cost is calculated, and the task is removed from $\mathcal{Q}$ and $\mathcal{R}$ queues (Lines 12 to 15). Otherwise, the task execution estimate is updated, as other tasks may have left the system, and more resources may be available at the corresponding VC (Lines 16 and 17). We emphasize that Algorithm 5 is executed in the VEC controllers.

After a VC formation process, task scheduling is triggered, and information such as available VCs and the set of tasks is provided to the MARINA. In summary, Algorithm 6 describes the MARINA's operations in a VEC controller. In this sense, the controller gets the VC set V and task set T, which gives the T' task scheduling set as an output. The set V is sorted non-increasing, so VCs with more available resources are prioritized (Line 1). Two vectors are created based on T, the first PT for the estimated processing time, and the second D for deadlines (Lines 3 and 4). MARINA calls the procedure ParetoSet with configuration for joint minimization of vectors PT and D (Line 5). The procedure returns a set $\mathcal{P}$ containing the ID of the tasks that are part of the minimal Pareto set. With the Pareto set defined, the BCP uses this task subset to schedule tasks, considering

Algorithm 5: Vehicular Edge Computing (VEC) environment

```
1  Q, R ← ∅                                          ▷ Waiting and Run queues
2  V, T ← ∅                                          ▷ VC and Task sets
3  foreach time slot k ∈ K do
4      Q.enqueue(T_k)
5      if k is update interval then
6          V ← update VCs with model in Section 5.1
7      if Q ≠ ∅ then
8          T ← Q
9          R ← MARINA(T, V)                          ▷ Algorithm 6
10     if R ≠ ∅ then
11         foreach r ∈ R do
12             if r has completed its execution then
13                 compute cost with Equation (5.6)
14                 Q.dequeue(r)
15                 R.dequeue(r)
16             else
17                 Update r computation estimation
```

other computational requirements of each task (Line 6). In this step, BCP returns a subset of candidate tasks S'. Also, the total number of resources needed for this returned set is calculated (Line 7). After that, for each task in the set, it is verified if the VC will have available resources until its deadline D_l^t (Lines 8 and 9). If not, that task is removed from the set S' and its required resources are removed from the total resource estimate (Lines 10 and 11). If so, its actual processing time is calculated (Line 13). If the processing time is longer than its deadline, the task is removed from S' and will be rescheduled in the next round. Otherwise, set S' is added to the scheduled tasks list T'.

Computational complexity

The MARINA's time complexity is analyzed as follows. MARINA has three main stages. The first stage runs in time $\mathcal{O}(n \log n)$ in the worst case, which is the time complexity to find Pareto set [10]. In the same direction, the BCP algorithm needs $\mathcal{O}(n \log n)$ to sort non-increasing order of items by sizes (CPU cycles) and to cycle through all the items to be checked. Also, the number of tasks decreases based on the select VC with maximum resources available, where the number of VCs is represented by m. In the worst case, the second and third stages have linear $\mathcal{O}(n)$ complexity to check the deadline and computation delay constraints for each temporarily scheduled task S'. Finally, a constant $\mathcal{F}$ is added that represents the computational complexity of the prediction step, which may vary according to the technique used. In summary, the algorithm's time complexity can be described as $\mathcal{O}(\max\{m\} + n \log n + \mathcal{F})$ in the worst case, with m being the number of VCs and n the number of tasks. In this context, MARINA is a polynomial-time algorithm.

Algorithm 6: MARINA mechanism overview

Input: task set T and VC set V
Output: scheduled tasks T'

$V \leftarrow$ decreasing order of available resources
foreach $v_j \in V$ **do**
  $PT \leftarrow$ processing time (t_l, v_j) for each $t_l \in T$
  $D \leftarrow$ deadline of each $t_l \in T$
  $\mathcal{P} \leftarrow$ ParetoSet($\{PT, D\}$, sense=$[min, min]$)
  $S' \leftarrow$ BinCoveringProblem($\mathcal{P}, v_j$)
  $totalResources \leftarrow \sum_{l=1}^{|S'|} w_l^t,\ \forall t_l \in S'$
  foreach $t' \in S'$ **do**
    ▷ `use predicted vehicular information`
    if $totalResources < v_j$ *until* D_l^t **then**
      $S' \leftarrow S' \setminus \{t'\}$
      $totalResources \leftarrow totalResources - t'$
    else
      $d_{lj}^t \leftarrow$ Equation (5.5)
      if $d_{lj}^t > D_l^t$ **then**
        $S' \leftarrow S' \setminus \{t'\}$
      else
        $T' \leftarrow S'$
return T'

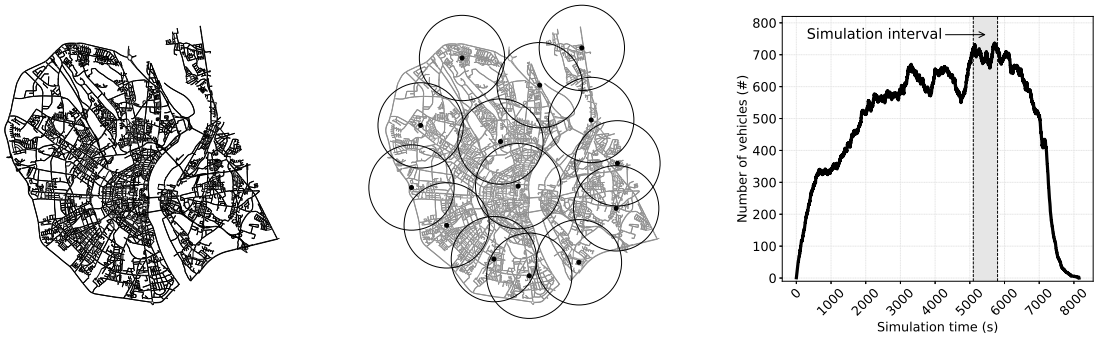
5.3 Performance Evaluation

This section describes the methodology and metrics used to evaluate MARINA performance in a VEC environment. First, we show the simulation environment, including implementation, parameters, and evaluation metrics. Second, to better understand the resource prediction model used, we present LSTM results compared to other models in the literature. Finally, we present and discuss the results of task scheduling using the prediction model employed and the main insights.

Simulation environment

The experiments were carried out with the SUMO, in version 1.11.0. The algorithms were implemented in Python 3.8 and connected to SUMO through the TraCI interface. We considered a deterministic realistic mobility trace from TAPAS Cologne[2] project, which reproduces vehicle traffic in the city of Cologne, Germany, as shown in Figure 5.6(a). The trace contains vehicular mobility from 6 to 8 AM on a typical working day and covers a region of 400 km^2. However, only a city submap with 114 km^2 was picked for our simulation experiments because it contains a greater variability of vehicles over time and

[2] `http://kolntrace.project.citi-lab.fr/`



(a) TAPAS Cologne submap (b) Base Stations (BSs) in the scenario (c) simulation interval considered

Figure 5.6: Simulation scenario.

up to 700 vehicles at peak times. The simulation time was 700 s, with 100 initial seconds of warm-up, as shown in Figure 5.6(c). The simulations were run 33 times to obtain a 95 % confidence interval.

The Bag-of-Tasks (BoT) applications were considered since they have no dependence on each other and can be executed out of the submission order. The tasks' deadline varies between 3, 5, and 7 seconds. This is important to generalize the representation of possible application classes. The VC formation interval was defined in 5 seconds. The scenario considers different task arrival rates (*i.e.*, 1, 3, and 5 tasks/second) following a Poisson distribution [41, 148, 138]. Also, the size s_l^t of each task is $[1, 10]$ MB. The number of CPU cycles w_l^t required to complete the task is fixed in $[1, 30]$ Million of Instructions (MI). We consider that a vehicle u_i makes available resources (CPU-cycle capacity and storage capacity) in 1 unit/vehicle, the number of CPU cores ω_{u_i} (CPU-cycle capacity in Million of Instructions Per Second (MIPS)), which without loss of generality is proportional to the storage capacity ϕ_{u_i} (1 MB). In this way, we can get an idea of the impact that sharing lower resource units employ on the system. The communication ranges of vehicles and BSs were 250 m and 2000 m, respectively.

In addition, we consider 14 BSs, and each one is capable of sharing processing and storage resources, where such values were configured at 15 MIPS and 15 MB, respectively. We deployed 4 VEC controllers, and each can manage up to 4 neighboring BSs. We deployed the BSs in the city following positioning information provided by the TAPAS Cologne project, as shown in Figure 5.6(b). Moreover, we used the TensorFlow framework version 2.8.2 to implement the RNN [1]. We also consider a Graphics Processing Unit (GPU) NVIDIA(R) Tesla V100 with 5120 CUDA cores and 32 GB of VRAM to train the ML models. Table 5.2 summarizes the main simulation parameters.

We considered three scheduling mechanisms to compare their performance with MARINA, namely: (*i*) *UNC* [60] task scheduling scheme is a classic queuing theory algorithm that is widely used as a policy for scheduling tasks in computational systems. It is similar to *First-Come-First-Serve* (FCFS) scheme, but the *UNC* scheme is free to select any task in the task queue for scheduling during the current time. (*ii*) *FORESAM* [117] considers a multi-criteria analytical method to select the most appropriate VC to receive the task. *FORESAM* considers all task requirements. (*iii*) *CRATOS* [33] considers a combinatorial

Table 5.2: Simulation parameters for task scheduling assessments.

Parameter description	Value
BS's wireless communication range	2000 m
Vehicle's wireless communication range	250 m
Mean size of the tasks	[1,10] MB
Required CPU of the tasks	[1,30] MI
Task delay constraint	3, 5, and 7 s
Tasks distribution	Poisson
Tasks arrival rate λ	1, 3, and 5 tasks/second
Computational capability of each BS	15 MIPS
Computational capability of each vehicle	1 MIPS
Number of vehicles	$550 \sim 700$
Simulation time	700 s
Scenario area	114 km^2
Number of VEC controllers	4
RS's communication latency	[1,5] ms
Recurrent layer	Bidirectional LSTM
Loss function	Mean Square Error (MSE)
Batch size	64
Sequence length	5
# of LSTM cells	128
Number of epochs	100

optimization approach to schedule tasks in the available VCs. The algorithm was adapted for our scenario. The value of each task was defined as 1, given that, by default, the tasks have no value associated with them.

We consider the following evaluation metrics:

- *Root Mean Square Error (RMSE)*: quantifies the difference between ground-truth and predicted data regarding resource availability in each VC. This metric is widely used to measure the performance of predictors. In our case, it was used to evaluate and decide the best model for predicting vehicular resources for the proposed task scheduling mechanism.

- *Scheduled tasks (%)*: percentage of successfully scheduled tasks. We consider successful scheduling when a task is scheduled in a VC and can be executed respecting its deadline constraint.

- *Monetary Cost ($/time)*: represents the monetary cost of using VC's computational resources for k units of time. As defined in Section 5.2, a VC comprises vehicles and BS, each of which has a different monetary cost.

- *System latency (s)*: refers to the processing time of the task in a given computational configuration plus the queue waiting time. That is, this metric shows how efficient the decisions made by the mechanisms are.

- *CPU time (s)*: represents the sum of the execution time of all the processes involved in the mechanism. This time may vary depending on the machine's configuration used in the evaluation. For this evaluation, we used an Intel(R) Xeon(R) CPU X5650 (24×2.66GHz) with Linux architecture x86_64.

5.4 Results

Mobility prediction

We used 85 % of the samples of the data for training and 15 % for testing to analyze the RNN performance in our scenario. The prediction performance of the RNN was measured in terms of RMSE compared to Dense Neural Network (DNN) and Support Vector Regression (SVR), which are two approaches widely used in state-of-the-art to predict time series. Figure 5.7 shows an example of a time series built for VC with ID number 4. As can be seen, most of the time series (85 %) is used for model training, with the remainder (15 %) being used to test the predictions. Given the characteristics of the mobility trace, records from 6000 s were disregarded as they only contain the dynamics of removing vehicles from the simulation.

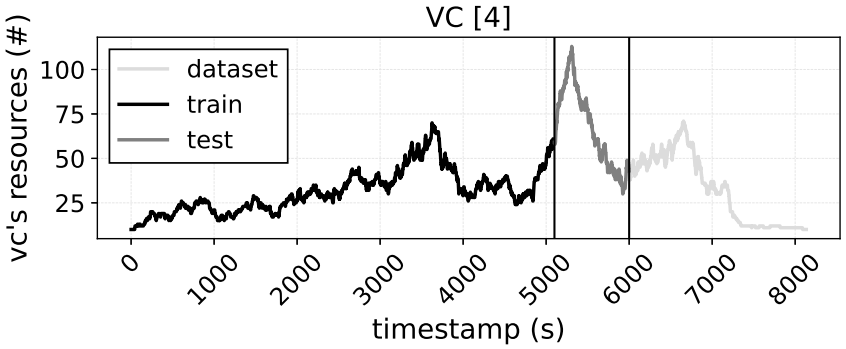
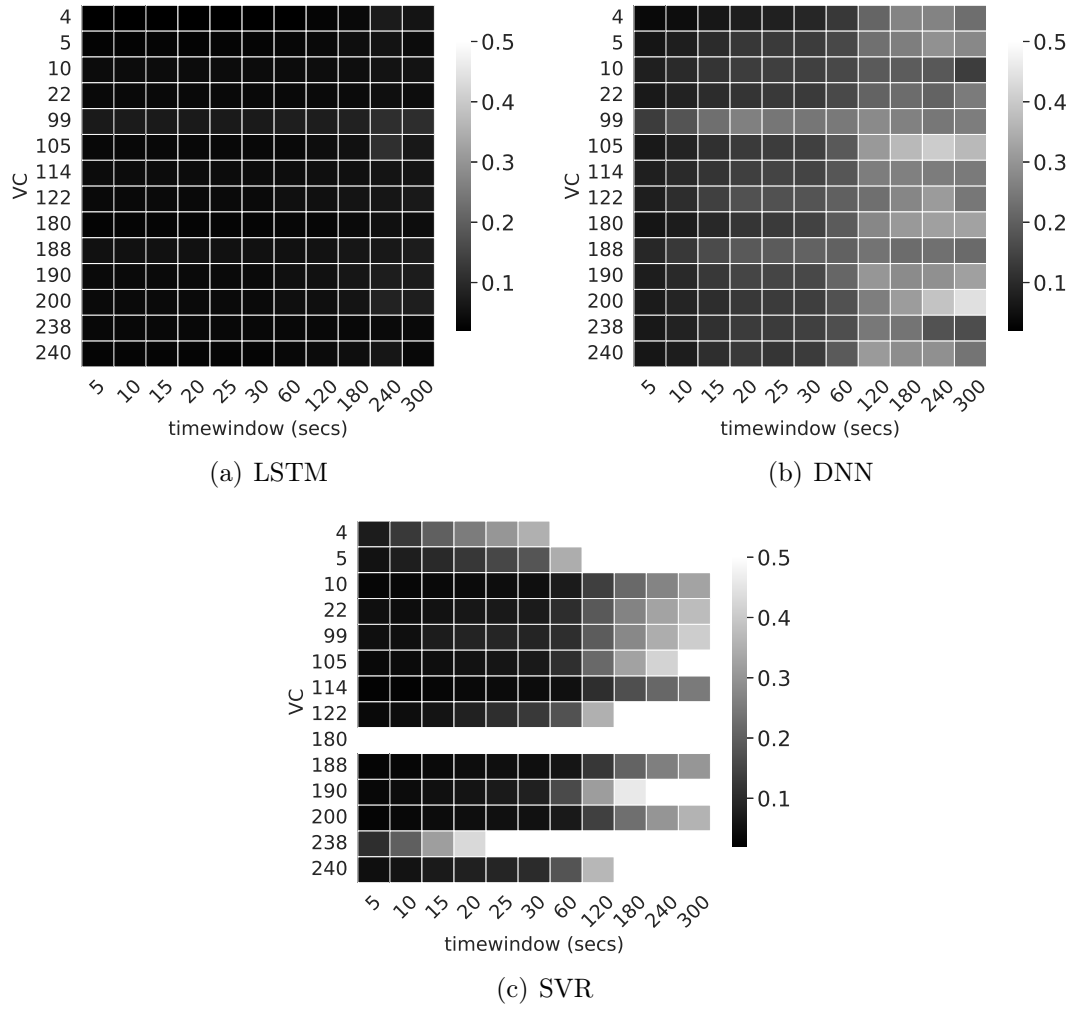


Figure 5.7: Example of the data used for the training and prediction processes.

Figure 5.8 compares the RMSE obtained in the predictions with the considered models. This assessment considers the average RMSE among the 14 VCs in the scenario in different time horizons for prediction. At this stage, we performed an exploratory assessment of the size of the prediction window. This is important to show the degradation of the prediction as the prediction window increases, which is expected in this type of evaluation. Thus, for each VC, we consider the prediction windows 5, 10, 15, 20, 25, 30, 60, 120, 180, 240, and 300 seconds. In summary, the hyperparameters for training the LSTM model are: number of epochs = 100, batch size = 64, and number of LSTM cells = 128.

In this sense, we can see that the predictions provided by LSTM achieve an average RMSE of about 0.1 in most prediction windows, while the predictions provided by DNN and SVR suffer more significant degradations as the prediction window increases. LSTM is generally more efficient than other models at considering past events to provide predictions closer to dataset ground truth. Thus, this type of RNN proved ideal for helping our task scheduling mechanism. Based on the results, the LSTM proved to be more robust in

resource prediction phase. Thus, we chose to use it to help our task scheduling process with the time window setting equal to 5 s.



(a) LSTM

(b) DNN

(c) SVR

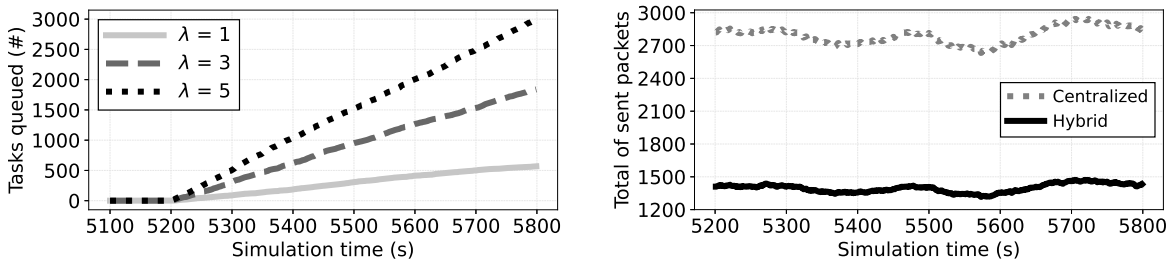
Figure 5.8: Results of resource predictions by Long Short-Term Memory (LSTM), Dense Neural Network (DNN), and Support Vector Regression (SVR).

Task scheduling

Figure 5.9(a) shows the behavior of the waiting queue Q over the simulation time considered if no scheduling mechanism is used. Different arrival rates significantly increase the difficulty of orchestrating the task schedule process. For example, at the highest arrival rate ($\lambda = 5$), 4500 tasks are queued at the end of the simulation run. In other words, it is a challenging scenario, considering that task sizes and deadline constraints also vary according to their arrival in the system.

Figure 5.9(b) shows the difference in packets sent on the network, considering the centralized and hybrid architectures. In this sense, in a centralized architecture, where one entity makes all the decisions and builds its global knowledge, all vehicles must maintain communication with that entity. Thus, the number of packets sent per time unit is directly related to the number of vehicles communicating with the central entity.

However, in a hybrid architecture, intermediary entities aggregate the messages of vehicles in their coverage and send this information to the remote server in a single data stream. The number of data messages transiting the network core is significantly reduced. In this evaluation, we can observe an average reduction of 50 % in the number of sent packets on the network when the hybrid architecture is considered.



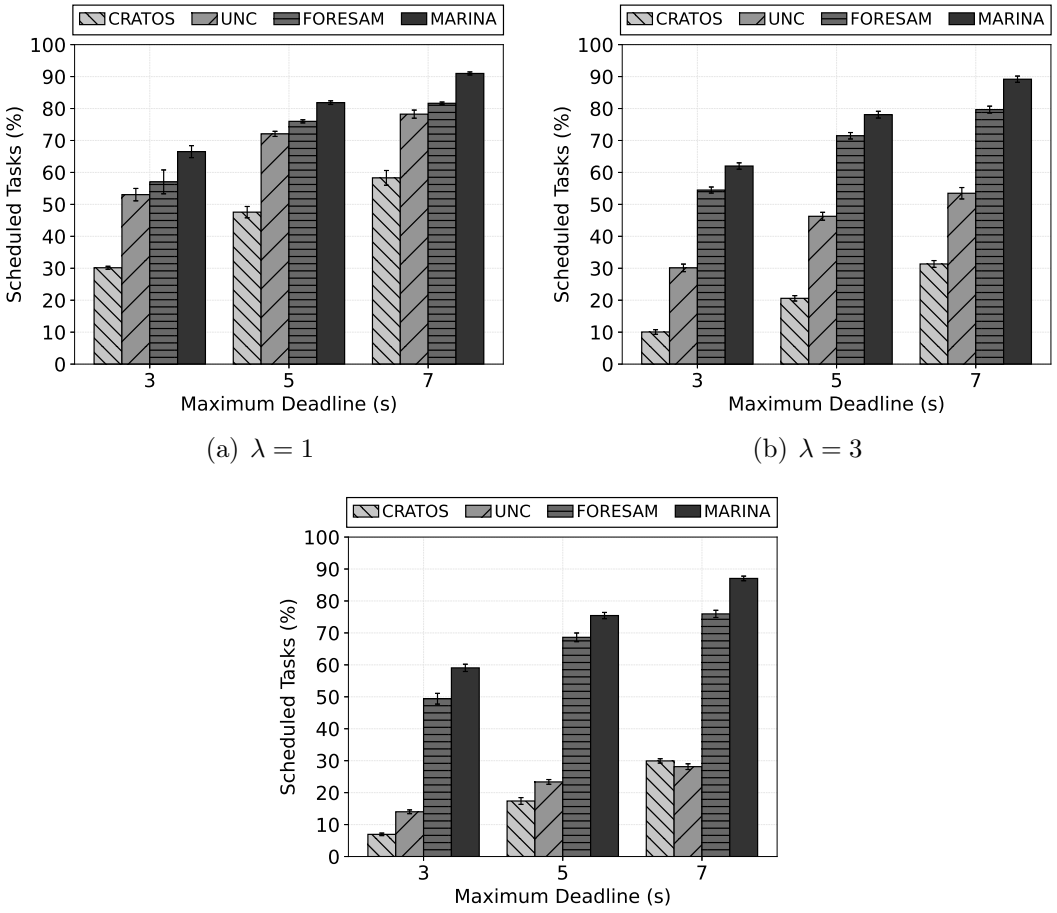
(a) Example of task queue Q over the simulation time considered.

(b) Total sent packets on the network considering different system architectures.

Figure 5.9: Task arrival rate example and total sent packets on the network.

Figure 5.10 displays the percentage of successfully scheduled and executed tasks with different maximum delay constraints 3, 5, and 7 seconds, respectively. We can check the behavior of the mechanisms when the system receives a high load of requests. All mechanisms improve their performance as the deadline increases. However, we can see that they all have difficulty scheduling as the task arrival rate increases. In all configurations MARINA performs better than other mechanisms. Also, when the arrival rate is equal to 1, MARINA can schedule over 91 % of the tasks in the configurations with the largest maximum deadline constraint, as shown in Figure 5.10(a). Both FORESAM and UNC operate similarly in this configuration. CRATOS has the worst performance when the deadline increases. This is due to its selection strategy, which is only concerned with the task size, not considering fundamental aspects such as deadline and computation delay. The tasks scheduled with CRATOS have processing restrictions that are not considered in its decision-making process. In the configuration with a task arrival rate equal to 3, MARINA already starts to have difficulty scheduling when the deadline grows, as shown in Figure 5.10(b). However, the MARINA still manages to schedule up to 90 % of tasks in scenarios with the highest deadline constraints. In this configuration, we see a more significant difference between FORESAM and UNC due to UNC applying the simple greedy policy. In this scenario, MARINA is superior in all configurations considered. Finally, when the task arrival rate is 5, MARINA maintains its results closer to 88 %. FORESAM has results close to MARINA in this scenario. In FORESAM's setup, the main factor taken into account in its decision process was the task deadline, so when the deadline increase, its scheduling choices cannot relate to all the problem restrictions well. Even applying a simple scheduling method, the UNC can handle many requests compared to CRATOS.

Figure 5.11 shows the monetary cost of using the computational resources of the VCs. As mentioned in Section 5.2, the cost of the vehicle's resources is less than that of the BS. Therefore, to minimize this cost, the approaches choose to select the resources of the



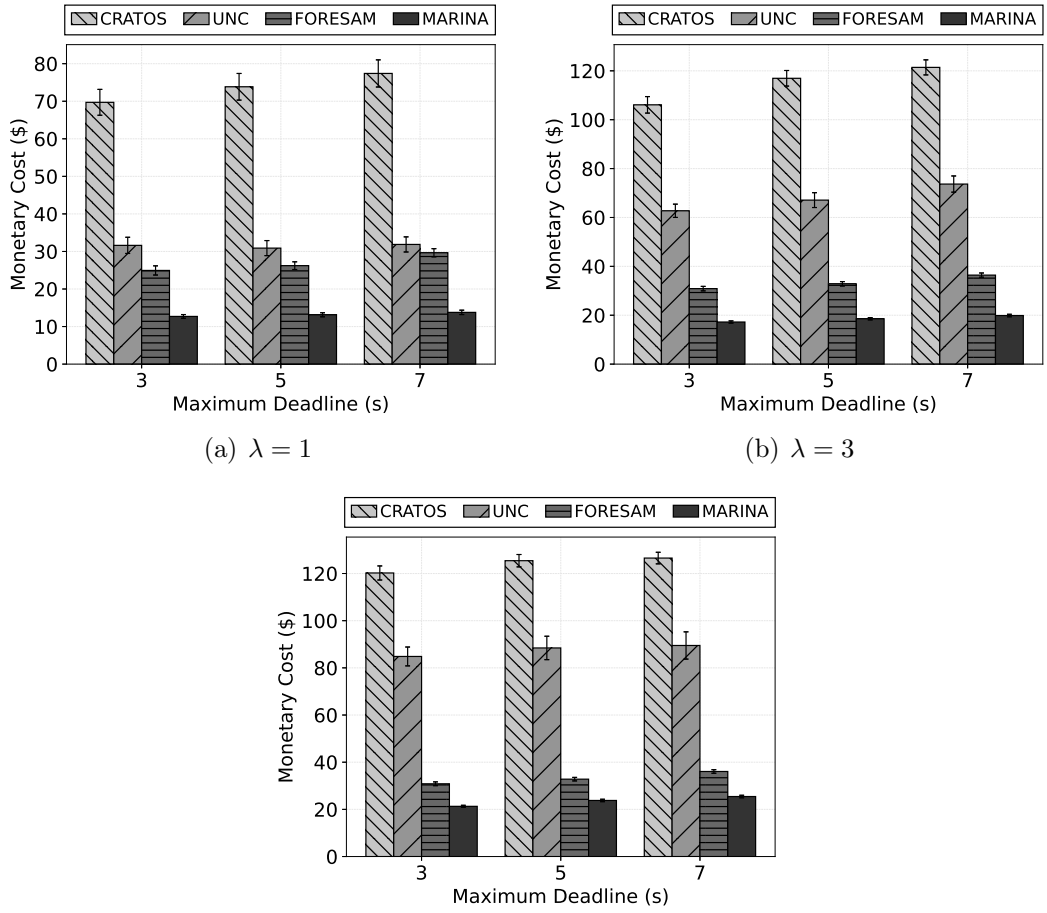
(a) $\lambda = 1$

(b) $\lambda = 3$

(c) $\lambda = 5$

Figure 5.10: Results of the scheduled tasks with different maximum deadline constraints and task arrival rates.

vehicles for the scheduling first. We can see that MARINA minimizes the monetary cost in all evaluations. In this case, it is natural for the performance of all mechanisms to fall due to the percentage of scheduled tasks that also decreases in more challenging scenarios, as shown in Figure 5.10. The best performance of MARINA occurs due to the selection of the tasks considering the VC's predicted available resources and the joint minimization provided by the Pareto set approach. Applying BCP allows a more significant number of tasks to be scheduled out in the same VC, prioritizing that such resources come from vehicles and not from BS. In addition, the more challenging the scenario (increasing the task arrival rate) becomes, the more costly the scheduling process becomes. More computational resources need to be used to run the existing tasks. MARINA reduces the monetary cost by up to 80 % in all scenarios. CRATOS proves to be more costly because its decision process involves only the amount of computational resources available. In this way, many tasks are rescheduled during the mechanism's operation, thus increasing its final monetary cost. The UNC maintains its expected behavior of increasing the monetary cost as deadline constraints increases. The application of Pareto set and BCP by MARINA allows the best task set to be scheduled in the same VC with minimum processing time and deadline constraints.



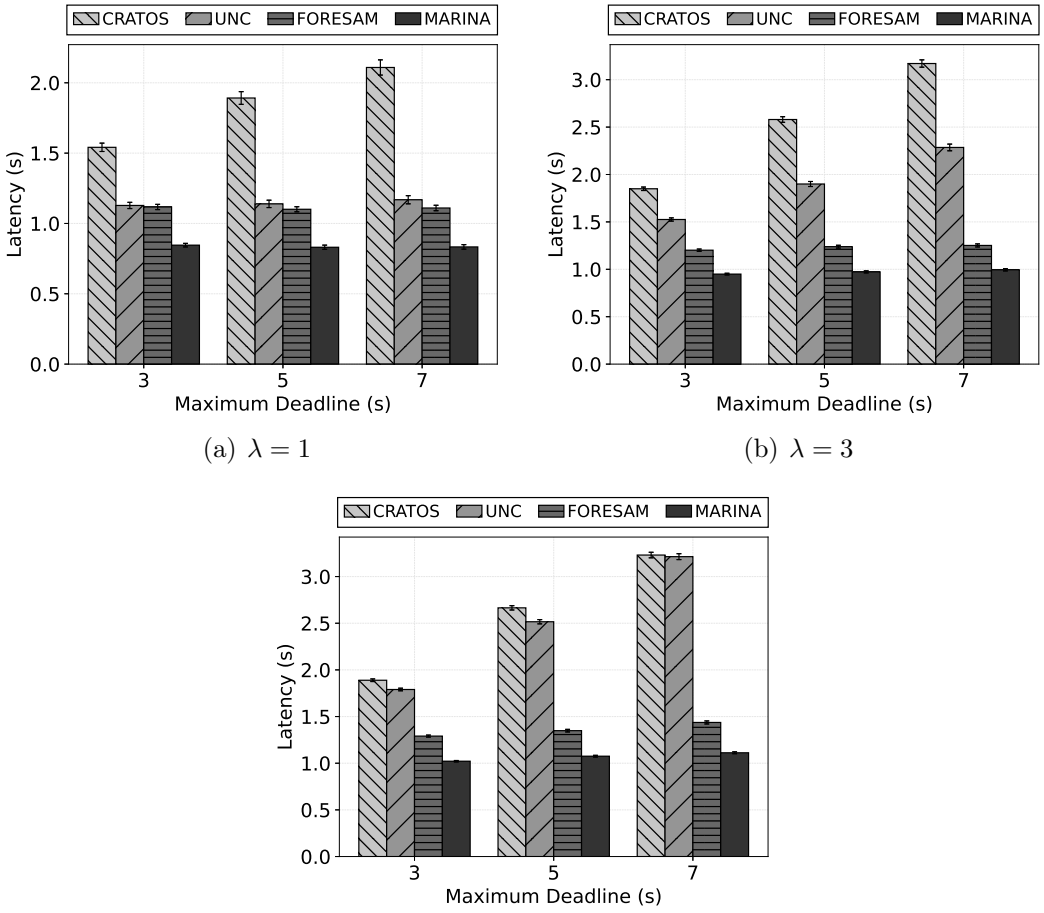
(a) $\lambda = 1$

(b) $\lambda = 3$

(c) $\lambda = 5$

Figure 5.11: Results of the monetary cost with different maximum deadline constraints and task arrival rates.

Figure 5.12 depicts the system latency results, including the computational delay and queue time. This metric is important because it shows the impact of scheduling tasks in a given processing configuration and is directly associated with how the scheduling approach selects VCs. We can see, in all evaluations, that MARINA reduces the computation delay of the tasks in the VCs. Hence, it is mainly due to selecting the VCs employed by MARINA, filtering based on the VC's processing rate considering the task deadline constraint. In addition to selecting the VC with greater computational capacity, selecting tasks based on the joint minimization of processing time and deadline helps reduce system latency. In addition, CRATOS and UNC employ a higher computation delay in all assessments due to their decision strategy that considers only the order in which tasks arrive in the system. Specifically, in the configuration with a tasks arrival rate equal to 3 and 5, MARINA and FORESAM stabilize their performance due to the absence of considerable variation in the number of tasks scheduled, as shown in Figures 5.12(b) and 5.12(c). In summary, MARINA has better managed VC's resources when considering aspects of both tasks and VCs in its decision-making process. In other words, an efficient task scheduling approach should maximize the number of scheduled tasks while minimizing the time that such tasks await their schedule.



(a) $\lambda = 1$

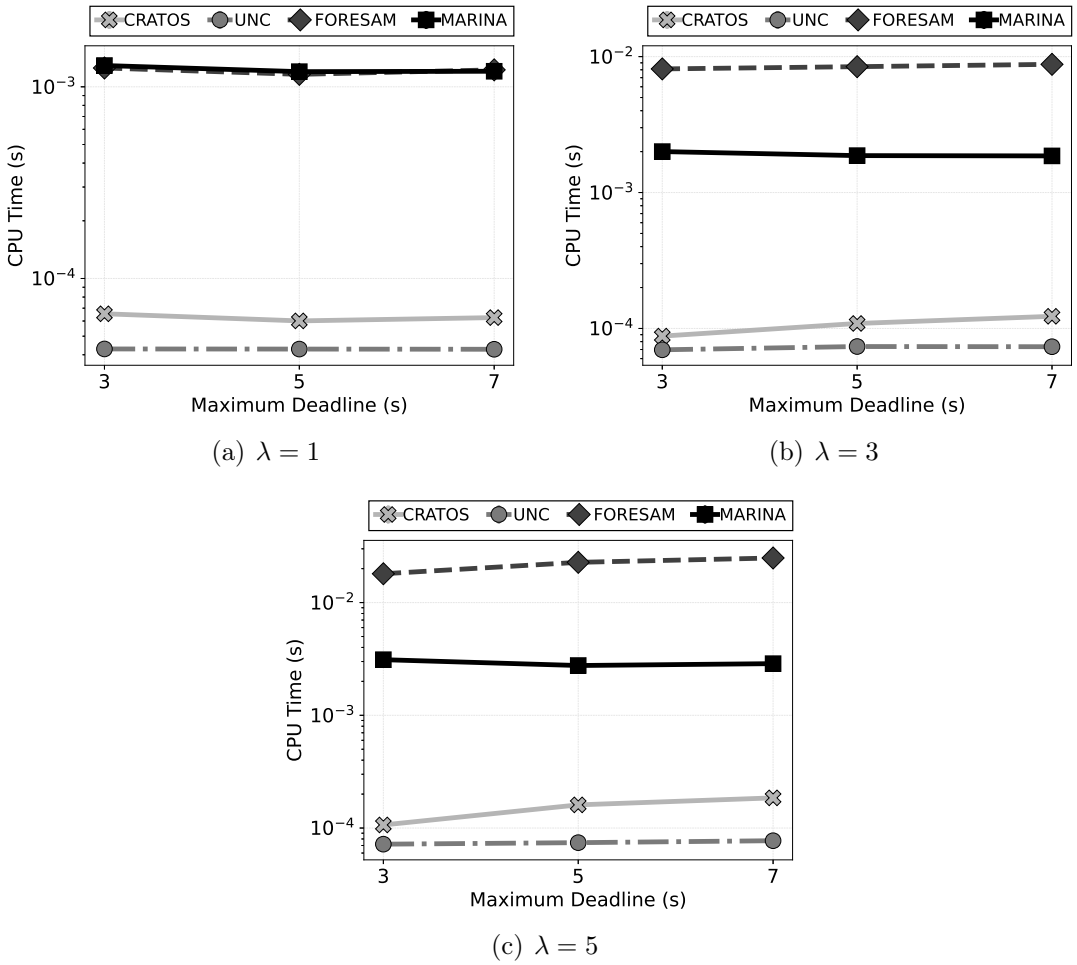
(b) $\lambda = 3$

(c) $\lambda = 5$

Figure 5.12: Results of the system latency with different maximum deadline constraints and task arrival rates.

Finally, Figure 5.13 presents the CPU time by the VEC controllers that run the scheduling approaches. This metric is directly related to the approach's computational complexity. Also, an approach that manages to schedule more tasks has more overall CPU time, even with less computational complexity. In all evaluations, UNC has lower CPU time because its selection method is simple, selecting the task to be scheduled based on its order of arrival in the system, operating with time complexity $\mathcal{O}(n^2)$, where n is the total number of tasks. CRATOS has the second-lowest CPU time, but this can be justified by the number of tasks scheduled and successfully executed, even having pseudo-polynomial complexity ($\mathcal{O}(\max\{m\}+n\times W)$, where m is the set VC, n is the number of tasks, and W is the size of the VC considered in each round). FORESAM has a high CPU time because it iteratively selects one task at a time given a VC, making a more significant number of checks for each VC considered in the round. FORESAM uses the AHP technique in its decision process, and the time complexity of AHP is $\mathcal{O}(\min\{mn^2, m^2n\})$, where m is alternatives, and n is criteria. Finally, MARINA has CPU time statistically close to FORESAM. This is also true of its iterative decision-making process. In certain rounds, the returned Pareto set may be small, requiring further rounds to fill the VC. A 2-dimensional set is constructed at each decision round, and the algorithm searches for the

Pareto optimal set, taking $\mathcal{O}(n \log n)$ time.



(a) $\lambda = 1$

(b) $\lambda = 3$

(c) $\lambda = 5$

Figure 5.13: Results of the Central Processing Unit (CPU) time with different maximum deadline constraints and task arrival rates.

5.5 Chapter Conclusions

This chapter introduced MARINA, an efficient task scheduling for VEC environments. MARINA divides its scheduling process into three stages. The first step is finding the Pareto set based on a joint minimization of processing time and deadline constraints. After that, it applies BCP with the FFD heuristic to select a set of tasks for a given VC, aiming to maximize the number of tasks scheduled and minimize the monetary cost of this processing. The second step selects the tasks based on the correlation between the tasks' deadline constraint and predicted information about the computational resources available in each VC. Finally, the third step verifies the processing time of these tasks in the selected VC to reduce the computation delay and, consequently, the monetary cost of using the computational resources. MARINA employs an RNN architecture to predict vehicular resources in VCs and assist in its decision-making process.

Chapter 6

Fairness and Load Balancing in Vehicular Clouds

As already discussed and exemplified in previous chapters, one of the main challenges of task scheduling in VCs is the dynamic nature of the vehicular network, as vehicles move and change their position, making it challenging to predict resource availability in a specific location. To tackle this challenge, advanced strategies are used to determine the best task placement based on factors such as communication costs and mobility prediction [69].

However, much less attention has been paid to questions of fairness in resource usage in the scheduling process [18]. We treat this as an issue of load balancing among the nodes processing tasks (vehicles and BSs), that is, distributing the workload evenly across available computational resources [61]. Load balancing is performed either when a task arrives or once it has already been queued [99]. It ensures that tasks are allocated efficiently to prevent resource over or underutilization [75]. By balancing the load, the system can achieve better resource utilization and response times. Additionally, load balancing can help mitigate congestion and handle varying demand patterns, resulting in a more efficient and effective task scheduling process [154]. Therefore, another important challenge of VCs is to meet user demands by maintaining fair load balancing among available computational resource usage [18, 61] while still considering node mobility in the scheduling process.

Towards this end, this chapter introduces FARID (**FA**ir **R**esource usage in veh**I**cular clou**D**s). FARID uses the same decision strategy employed by MARINA but applies a probabilistic selection function to choose VCs who will participate in the decision process. FARID runs on VEC controllers and uses Pareto optimality to schedule tasks in different VCs. The mechanism splits the set of tasks into different parts to improve the system efficiency with parallel management, obtaining k different Pareto sets and being able to make k decisions at the same time, where k is the number of threads running in each VEC controller. FARID aims to minimize processing time within VCs, thus reducing resource utilization and, subsequently, monetary costs. Also, it considers contextual aspects in its decision process, such as resource mobility in each VC and task's requirements. We assessed the efficiency of FARID compared to other mechanisms, and the results indicate its capability to schedule a larger quantity of tasks, minimize monetary costs, and reduce overall system latency. Lastly, FARID employs better load balancing in the scheduling

process, resulting in greater fairness in the resource usage of VCs.

6.1 FARID

This section describes FARID, which considers Pareto optimality and a probabilistic selection function to maximize scheduled tasks, load balancing, and fairness in resource usage.

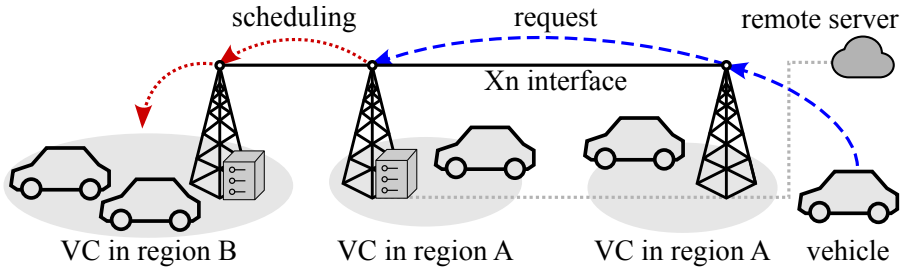


Figure 6.1: A system architecture employed by FARID, presenting its main components, such as vehicles, Base Stations (BSs), Remote Server (RS), Vehicular Clouds (VCs), and Vehicular Edge Computing (VEC) Controllers.

Figure 5.1 presents the system architecture composed of vehicles, BSs, VEC controllers, and a RS in the Internet. The scenario have a set of x vehicles, denoted as $u_i \in U = \{u_1, u_2, \ldots, u_x\}$. Also, there is a set of p BSs deployed in the city, denoted as $b_y \in B = \{b_1, b_2, \ldots, b_p\}$. Each BS can communicate with the RS via optical fiber link.

To improve the management of BSs, the city is divided into R regions, and each region has at least one BS. Furthermore, we consider a set of $|R|$ VEC controllers, since each region is managed by exactly one controller. Therefore, after the association between the vehicle and BS, the BS sends this information to the RS. The association process considers the Max-SINR approach. BS information only is updated as the number of vehicles in its coverage changes.

In the resource aggregation process (VC formation), the VEC controller needs to request the RS about BSs and vehicles information to build its regional knowledge [35]. In this way, the system employs a *Publish/Subscribe* scheme to obtain the relevant information without introducing unwanted traffic into the network. The set of VCs can be denoted by $v_j \in V = \{v_1, v_2, \ldots, v_m\}$, where m is the total number of VCs. We consider that the number of VCs is the same number of BSs, as the positioning of the BS defines where the VC will act. In summary, a VC consists of a set of vehicles and BS capable of sharing processing power ω in MIPS and storage capacity ϕ in Megabytes (MB). The total amount of processing power Ω_j and storage capacity Φ_j of each VC v_j is the sum of the shared resources of vehicles and BS that make up these VCs.

Due to VC's resource variability over time, we utilize accurate vehicular mobility data to determine each vehicle's stay within BSs' coverage. Even though mobility prediction is not the primary focus of this study, we employ an optimal mobility prediction method. Future vehicle mobility data is gathered from the vehicular dataset within a time window Z. To simulate prediction errors, we introduce a white Gaussian noise to each collected

data [37]. Finally, as we now estimate the available resources in the VCs at $z \in Z$ time units, these resources can be denoted by Ω_{jz} and Φ_{jz}.

6.2 Problem Definition

Each task $t_l \in T = \{t_1, t_2, \ldots, t_n\}$ is denoted by a tuple $\{id_l^t,\ s_l^t,\ w_l^t,\ D_l^t\}$ where id_l^t represents the unique identification number, s_l^t denotes the input data size (in MB), w_l^t is the number of CPU cycles required to process the task, and D_l^t is its deadline constraint. The processing time d_{lj}^t (*i.e.*, execution time of a task in a specific computational configuration) can be obtained based on the required CPU cycle w_l^t divided by the server's CPU cycle frequency Ω_j, as $d_{lj}^t = \frac{w_l^t}{\Omega_j}, \forall t_l \in v_j$.

As VC's computational resources are shared among different tasks, the Ω_j considered for getting processing time for a given task must be updated according to the degree of sharing of this resource within the VC, represented by Ψ_j. Thus, Ω_j is divided by the number of tasks $|T_j'|$ that were scheduled in this VC to yield $\Psi_j = \frac{\Omega_j}{|T_j'|}, j \in V$.

Additionally, each task has a deadline constraint D_l^t. This deadline represents a time limit that the task can wait to be processed. If $d_{lj}^t \leq D_l^t$, the task can be scheduled and executed in the VC v_j. Also, when a task is scheduled and starts to be processed, there is a cost associated with this execution. The monetary cost is modeled as

$$\mathcal{C}_l = d_{lj}^t \times (w_l^t \times ResourcePrice(t_l)). \tag{6.1}$$

where d_{lj}^t is the t_l processing time in $v_j \in V$ and w_l^t is its CPU cycles required. $ResourcePrice(t_l)$ indicates the resource price used and is set to \$14.309 (if t_l uses BS's resources) or \$6.27 (if t_l uses vehicles' resources). The prices are based on instances available on Amazon EC2 [1] (Region Europe, Frankfurt), such as *g4ad* (BS) and *g3* (vehicle).

In summary, when a task arrives in the system, it is queued and waits until it is scheduled. The VEC controller must select the VC to process this task. This selection decision should consider the VC's processing power over time and the task requirements. So, to consider these different objectives, a task scheduling problem was formulated that primarily seeks to maximize the number of tasks scheduled considering constraints that directly impact the monetary costs, as follows:

$$\text{maximize} \sum_{l=1}^{n} t_l,\ l \in T, \tag{6.2}$$

$$\text{subject to } d_{lj}^t \leq D_l^t,\ l \in S',\ j \in V, \tag{6.3}$$

$$\sum_{l=1}^{n} s_l^t \leq \Phi_{jz},\ l \in S',\ j \in V,\ z \in Z, \tag{6.4}$$

$$\sum_{l=1}^{n} w_l^t \leq \Psi_{jz},\ l \in S',\ j \in V,\ z \in Z. \tag{6.5}$$

[1] `https://aws.amazon.com/ec2/dedicated-hosts/pricing/`

The constraint (6.3) guarantees that the task deadline is respected and helps reduce the monetary cost, avoiding rescheduling. Also, constraints (6.4) and (6.5) ensure that VCs' storage and processing limits during the z required processing time intervals are respected.

FARID's operation

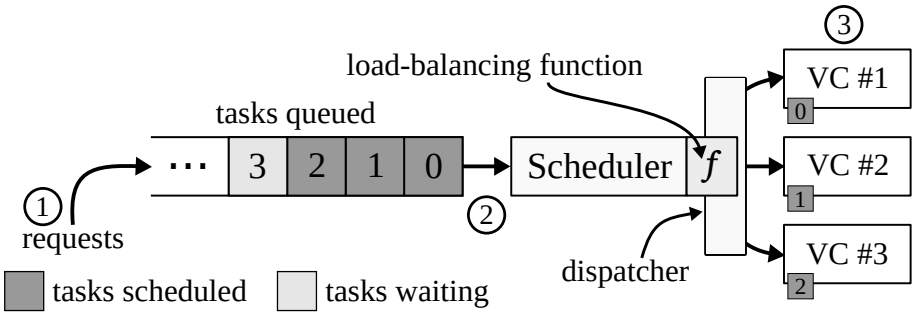


Figure 6.2: The task scheduling pipeline with a load-balancing function integrated into the scheduler.

Figure 6.2 presents a simplified task scheduling process pipeline in VCs. In summary, tasks are queued on the VEC controller as soon as they arrive in the system (Label ①). In this phase, the task can assume two states, waiting and scheduled. A task can assume a scheduled state and return to a waiting state if the VC fails to complete its processing and the task's deadline is $D_l^t > 0$. From that moment on, the task deadline must be observed constantly since the task is waiting for the scheduler's decision. Based on its criteria, the scheduler decides which VC the task will be processed. This scheduler can employ a load-balancing strategy to increase fairness levels in utilizing the VEC's computational resources. A Dispatcher has the role of distributing the tasks to the corresponding VCs (Label ②). The task distribution ratio is stored by VEC controller for load-balancing purposes in future decisions. Finally, the VCs receives the tasks and starts processing (Label ③).

In this context, FARID seeks the Pareto set using a dual-criteria approach: it simultaneously minimizes tasks' processing times and deadlines. A distinct vector is created for each criterion (processing time and deadline), arranged in a 2-dimensional plane to find the Pareto set. We can obtain a Pareto set in 2-dimensional in polynomial time $\mathcal{O}(n \log n)$ [37]. Besides, as the objective is to maximize the number of tasks scheduled, we simplify our problem to an instance of the BCP, which solves this issue [35].

Furthermore, FARID divides the task queue and set of VCs into k parts to make the scheduling process more efficient. Now, each k part is responsible for calculating a Pareto set and the universe of available VCs is smaller for the BCP application. FARID makes k decisions simultaneously. This division is performed in the Dispatcher. The `Dispatcher` Function in Algorithm 7 exemplifies this splitting process.

With the scheduling decision taken, carrying out a load balance among the available VCs to increase fairness in resource usage is essential. With this in mind, we consider a selection probabilistic function when the scheduler needs to select the VC in the scheduling

process. After a VC is selected, its selection probability decreases. Thus, when FARID selects the VCs in the next round, an ordering of the probability vector is performed, and the VC with the highest current probability is selected. With that, after m rounds, all VCs are selected at least once during the execution of FARID, where m is the total number of VCs.

In summary, every VC has an original probability of being selected during the scheduling process, according to

$$P_j^0, \quad j = 1, 2, \ldots, m \tag{6.6}$$

The selection probability is halved (0.5) if the VC is selected in the current round. Otherwise, the previous probability is maintained, according to Equation (6.7).

$$P_j^z = \begin{cases} 0.5 \cdot P_j^{z-1}, & \text{if item } j \text{ was selected} \\ P_j^{z-1}, & otherwise \end{cases} \tag{6.7}$$

To avoid the scenario that after a high number of rounds, the probability of a VC is reduced to zero and it is never selected again, a minimum nonzero probability is considered as

$$P_j^z = \begin{cases} 0.0001, & \text{if } P_j^z \leq 0 \\ P_j^z, & otherwise \end{cases} \tag{6.8}$$

Algorithm 7 describes the operations of FARID in a VEC controller. The controller gets the VC set V, the task set T, and the number of threads k, which gives the S scheduled task set as output. The `Dispatcher` Function splits the T and V into k parts and starts the execution of the `Decision` Function in parallel for each defined k (Lines 2 and 4). In the `Decision` Function, the VCs' probability vector is sorted in descending order to select VC with the highest selection probability (Line 6). For each v checked and selected, the probability must be updated (Line 8). Two vectors are created based on T, the first R for the estimated processing time, and the second D for deadlines (Lines 9 and 10). FARID calls the procedure ParetoSet with configuration for joint minimization of vectors R and D (Line 11). The procedure returns a set $\mathcal{P}$ containing the id of the tasks in the Pareto set. After that, FARID runs BCP to select the best subset $S' \in \mathcal{P}$ that best fits in V (Line 12). Also, the total number of resources needed for this returned set is calculated (Line 13). For each task in the set S', it is verified if the VC will have available resources until its deadline D_l^t (Line 15). If not, that task is removed from the set S' (Line 16). If so, its actual processing time is calculated (Line 19). If the processing time is longer than its deadline, the task is removed from S' and will be rescheduled in the next round. Otherwise, set S' is added to the scheduled tasks list S (Line 23).

6.3 Performance Evaluation

This section describes the methodology and metrics used to evaluate the efficiency of FARID. The experiments were carried out with the Simulation of Urban MObility (SUMO) 1.16.0. The algorithms were implemented in Python 3.8.10 and connected to SUMO through the TraCI interface. We used a central sub-map of 114 km^2 from TAPASCologne

Algorithm 7: FARID mechanism overview

Input: task set T, VC set V, and number of threads k
Output: scheduled tasks set $\mathcal{S}$

Function `Dispatcher`(T, V, k)**:**
    Split task set T into k parts
    Split VC set V into k parts
    Running `Decision` (T_k, V_k) for each thread k
Function `Decision`(T, V)**:**
    $V \leftarrow$ descending order of selection probability
    foreach $v \in V$ **do**
        Update v's probability using Equation (6.7)
        $R \leftarrow$ tasks' processing time for v
        $D \leftarrow$ tasks' deadline
        $\mathcal{P} \leftarrow$ ParetoSet($\{R, D\}$, obj=$[min, min]$)
        $S' \leftarrow$ BinCoveringProblem($\mathcal{P}, v$)
        $totalResources \leftarrow$ Sum all resources in S'
        foreach $t' \in S'$ **do**
            if $totalResources < v_j$ *until* D_l^t **then**
                S'.remove(t')
                $totalResources \leftarrow totalResources - t'$
            else
                $d_{lj}^t \leftarrow$ As shown in Section 6.2
                if $d_{lj}^t > D_l^t$ **then**
                    S'.remove(t')
                else
                    $S \leftarrow S'$
    return S

Table 6.1: Simulation parameters for fairness and load balancing assessments.

Parameter description	Value
BS's wireless communication range	2000 m
Vehicle's wireless communication range	250 m
Mean size of the tasks	[1,10] MB
Required CPU of the tasks	[1,30] MI
Task delay constraint	$[0.5, 0.8), [0.8, 1.0]$, and $(1.0, 3.0]$ s
Tasks distribution	Poisson
Tasks arrival rate λ	5 tasks/second
Computational capability of each BS	15 MIPS
Computational capability of each vehicle	1 MIPS
Number of vehicles	$550 \sim 700$
Simulation time	800 s
Scenario area	114 km^2
Number of VEC controllers	4
RS's communication latency	[1,5] ms

trace[2], which reproduces vehicle traffic in the city of Cologne, Germany. We consider 2 hours of vehicular mobility and up to 700 vehicles. The simulation time was 800 seconds, with 100 initial seconds of warm-up. We ran the simulations 33 times to obtain a 95% confidence interval.

The BoT applications were considered since they can be executed outside the arrival order. The tasks' deadline varies between $[0.5, 0.8), [0.8, 1.0]$, and $(1.0, 3.0]$ seconds. This is important for generalizing different application classes. VC formation intervals were 5 seconds. The arrival rate of tasks λ is 5 tasks/second, following a Poisson distribution. The communication ranges of vehicles and BSs were 250 and 2000 meters, respectively.

Furthermore, the size assigned to the tasks was $s_l^t = [1, 10]$ (MB), and the CPU cycles required varying in $w_l^t = [1, 30]$ Million of Instructions (MI). The number of CPU per vehicle is 1, which without loss of generality represents 1 MIPS. Each vehicle's storage capacity has been simplified to 1 MB. 14 BSs were used, and each can share processing power equal 15 MIPS and storage capacity equal 15 MB. 4 VEC controllers were considered, and each can manage up to 4 neighboring BSs. The BSs deployment positions followed the information provided by the TAPASCologne project.

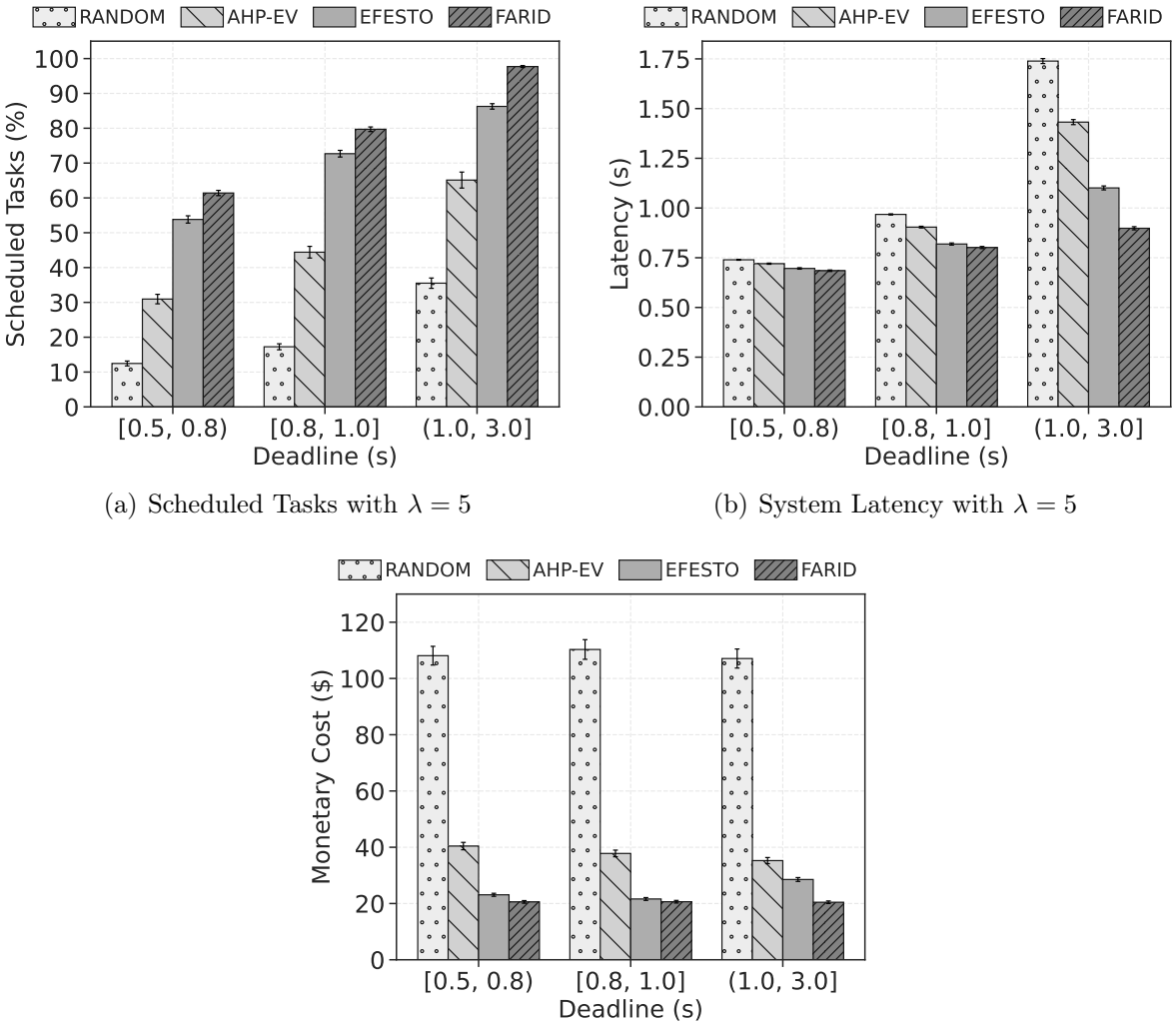
We compared the performance of FARID with three approaches, namely: *RANDOM*, which combines a policy based on First-Come, First-Served (FCFS) [60] with a randomized policy to select VCs [9]; *EFESTO* [37], which uses Pareto optimality in the task scheduling process and selects VCs with more resources every round; and *AHP-EV* [106], which uses the AHP multi-criteria approach in its decision-making process. Table 6.1 summarizes the main simulation parameters.

The metrics used for evaluation were: *i*) *Scheduled Tasks* represent the percentage of tasks successfully completed; *ii*) *Monetary Cost* refers to the resources usage price; *iii*) *System Latency* refers to the processing time plus the queue waiting time; and *iv*)

[2] `https://sumo.dlr.de/docs/Data/Scenarios.html`

Fairness represents Jain's fairness index. This metric is widely used to measure how fair the use of resources is in a computational system.

6.4 Results



(a) Scheduled Tasks with $\lambda = 5$

(b) System Latency with $\lambda = 5$

(c) Monetary Cost with $\lambda = 5$

Figure 6.3: Simulation results considering different deadline constraints.

Figure 6.3(a) shows the percentage of successfully scheduled and executed tasks. The performance of all approaches enhances as the maximum deadline extends. This means that the mechanisms have more time for decision-making and can make more unsuccessful attempts until the deadline is reached. FARID can schedule more tasks in all observed scenarios. Also, our mechanism can schedule more than 98% of tasks in configurations with less complex time constraints (maximum deadline $(1.0, 3.0]$). The use of mobility information helps in more accurate decisions, ensuring that the task will complete its processing in the selected VC. EFESTO is the second mechanism that can schedule more tasks. However, the fact that it only considers one Pareto set can make the decision

process difficult, as possible better subsets are disregarded. The AHP-EV reaches lower levels of scheduling, compared to the two previous mechanisms, due to its decision strategy that selects only one pair (task, VC) at each round. This approach aims to integrate the criteria, which are the task requirements, and select the best task for the current VC. RANDOM has the worst performance in all scenarios. This is due to its decision strategy, which is only concerned with the task's size employed by FCFS, disregarding fundamental aspects such as deadline and processing time. Besides, the selection of VCs is carried out at random to increase fairness in the use of resources. In the most challenging configuration, with a maximum deadline equal to $[0.5, 0.8)$, FARID is still superior, but managing to schedule only 61% of tasks.

Figure 6.3(b) shows the results regarding the system latency, which includes queue waiting time and task processing time. In this metric, the lower latency means the scheduling rounds are more efficient. In all the evaluation scenarios, it can be observed that FARID reduces the system latency. The best result is mainly achieved due to the selection of VCs employed by FARID, which minimizes the task processing time and the deadline constraint jointly. Also, the use of k decision-makings at the same time allows a smaller universe of tasks to be explored and helps to reduce system latency. In this way, FARID can handle a more significant number of tasks in a shorter time than other approaches. Mobility information helps estimate future resources, guaranteeing lower error rates in its scheduling process. RANDOM has higher latency in all evaluations due to its decision-making prioritizes the task size. EFESTO has an advantage over AHP-EV in the least challenging scenario (maximum deadline $(1.0, 3.0]$). In summary, FARID improved resources management by incorporating contextual aspects of tasks and VCs into its decision-making process.

Figure 6.3(c) shows the monetary cost of using the VCs' computational resources. As the resource prices differ, as discussed in Section 6.2, the approaches prioritize vehicle resources to minimize the final monetary cost. It can be noted that FARID minimizes the monetary cost in all evaluations performed. The best performance of FARID is due to the selection of tasks considering the future resources available in the VC. Pareto optimality allows the best task set to be scheduled in the same VC with minimum processing time and deadline constraints. Furthermore, separating the queued tasks into k parts makes the checks more efficient, and FARID makes fewer errors during the scheduling decision. RANDOM performs worse on this metric because its selection of VCs is entirely random, which does not guarantee that the task will be completed on the VC selected.

Finally, Figure 6.4 presents the fairness index obtained by the approaches when selecting VCs for scheduling. FARID obtains the best fairness index in all considered evaluation scenarios. The probability-based VC selection function allows all VCs to be selected during the scheduling process, thus increasing the network's ability to meet user demands (as seen in Figure 6.3(a)). The RANDOM approach obtained the second-best performance in this metric because it randomly selects a VC each round. This way, the chance of all VCs being selected increases significantly. However, it is crucial to view these metrics holistically. It is not sufficient for a scheduling approach to have a high fairness index if it can not satisfy even 15% of user demands in more complex scenarios. EFESTO has the worst fairness index since it prioritizes VCs with more resources in each scheduling

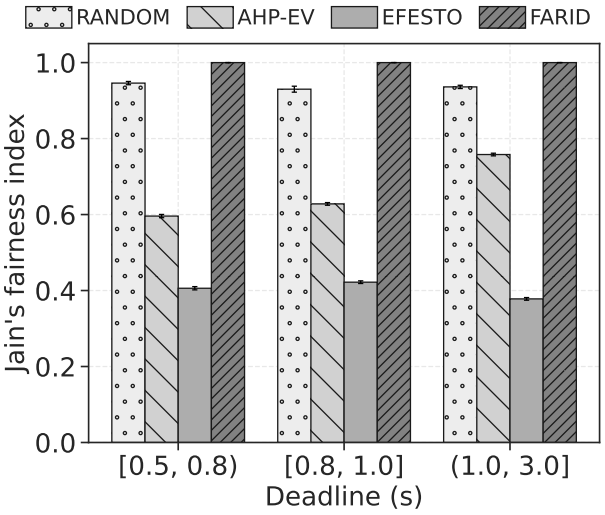


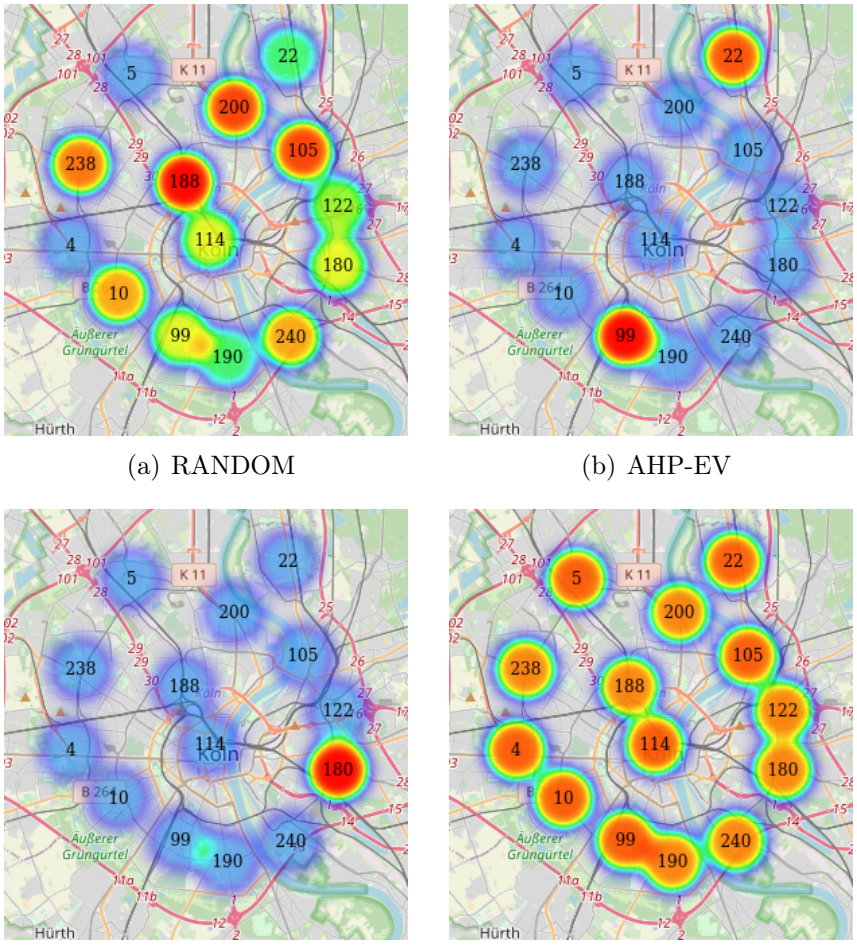
Figure 6.4: Jain's fairness index considering different deadline constraints.

round.

Figure 6.5 presents a visualization of the spatial distribution of tasks across VCs. This visualization complements the data shown in Figure 6.4. In this context, the EFESTO approach concentrates the majority of tasks in a single VC, which is, by definition, the VC with more available resources. The AHP-EV approach schedules tasks across VCs based on the multicriteria selection provided by AHP. AHP is a multicriteria decision-making technique that evaluates alternatives based on weighted criteria. When applied to task scheduling in distributed systems, AHP might favor tasks with higher priority or importance according to the defined criteria. This can result in uneven resource utilization and imbalanced load across different VCs, undermining system efficiency and fairness. The RANDOM approach demonstrates an acceptable spatial distribution of tasks due to its random nature in VC selection. However, fairness and load balancing must be analyzed along with the scheduling metrics, as both metrics need to be maximized. FARID is superior, as it jointly maximizes the scheduling metrics, fairness, and load balancing indices. The homogeneous distribution of tasks across VCs becomes apparent when FARID is considered.

6.5 Chapter Conclusions

In this chapter, we presented a task scheduling mechanism for VEC environments, using the Pareto optimality principle to select the best task set to be scheduled in VCs. We combine Pareto optimality with BCP to find the most suitable fit between tasks and VC resources. The task selection is based on contextual aspects, such as the processing time and task deadline. Besides, we consider vehicular mobility information to estimate the resources in each VC. The proposed approach also guarantees high levels of fairness in using vehicular resources, applying a probability function for load balancing on the selected VCs. Compared to state-of-the-art solutions, FARID has a higher level of fairness and can schedule more tasks while minimizing monetary costs and system latency.



(a) RANDOM (b) AHP-EV

(c) EFESTO (d) FARID

Figure 6.5: Spatial distribution of tasks processed with deadline equal to $[0.5, 0.8)$.

Chapter 7

Final Remarks

This chapter summarizes this thesis and discusses directions for future research. The objective is to highlight our main contributions and point out some possible directions to proceed with the research to address the drawbacks of the proposed solutions. In this context, we first present the thesis conclusions in Section 7.1. Then, in Section 7.2, we present future directions of this work. Finally, in Section 7.3, we present the publications related to this thesis.

7.1 Conclusions

The constant technological advancement has made cars increasingly intelligent and connected. This evolution significantly impacts the automotive sector as the industry strives to make vehicles increasingly intelligent and autonomous. In this context, specific applications and services, such as artificial intelligence applications and distributed learning, are emerging that require increasing computational power. Considering this, strategies must be developed to circumvent these resource limitations and provide services with computational resources closer to the network edge. With this in mind, MEC emerges, bringing resources to the edge of the cellular network that can be used by the end users, thus avoiding congestion in the network core.

VANETs can take advantage of the benefits brought by MEC, giving rise to what is called VEC. The VEC aims to aggregate the computational resources of vehicles and make them available on the network. This aggregation occurs through the clustering of vehicles, or as it is addressed in the VEC context, through the VCs formation. However, due to the inherent characteristics of VANETs, such as high mobility and intermittent communication, this VC formation is a significant challenge. And not just that, for services to utilize the resources present in these clouds, efficient task scheduling mechanisms must be elaborated for this challenging scenario.

Considering all of this, this thesis presented a set of solutions that mitigate the impacts of mobility in this VEC scenario. Each proposed solution is related to a research question, and this relationship can be summarized as follows:

1. **Research Question 1**: *How to form more stable VCs and ensure longer lifetimes?*

A mechanism based on vehicular mobility prediction for VC formation, referred to as PREDATOR, was proposed to address this question. PREDATOR is executed in network controllers present in RSUs, utilizing the information exchanged between vehicles and RSU through beacon messages to create a dataset and apply the mobility prediction method. With this information, PREDATOR can identify the most stable nodes (vehicles) to lead the VC. PREDATOR was compared to other approaches found in literature, and the results indicate that it outperforms these in both synthetic mobility scenarios and realistic ones. PREDATOR was introduced in detail in Chapter 4.

2. **Research Question 2**: *How to ensure that the topological dynamics of the VANETs does not negatively influence task scheduling in VCs?*

   A mobility- and deadline-aware mechanism for task scheduling in VEC scenarios, named MARINA, was proposed. MARINA uses an RNN architecture to estimate the computational resources of each VC. Besides, MARINA uses Pareto optimality as a decision strategy and models the scheduling problem as an instance of BCP. In this way, MARINA uses information from future resources with the scheduling mentioned above strategies to ensure that tasks are processed in VCs with a better chance of completing the processing. Simulation results in a realistic vehicular mobility scenario show that MARINA schedules more tasks while reducing system latency and the monetary cost of using resources in the VEC. MARINA was introduced in detail in Chapter 5.

3. **Research Question 3**: *How to use the computational resources of VCs in a fair and balanced way without degrading the system's overall efficiency in task scheduling?*

   A task scheduling mechanism concerned with fairness and load balancing in the utilization of vehicular resources has been proposed to address this question. The mechanism was called FARID. FARID uses the same decision strategy employed by MARINA but applies a probabilistic selection function to choose VCs who will participate in the decision process. Additionally, FARID divides the decision problem by partitioning the task waiting queue into k segments and applies multithreading to solve these subproblems in a parallel manner. This division results in lower system latency. Moreover, the probabilistic selection function enables all VCs to participate in the scheduling process at least once, leading to a more balanced processing load in the VEC scenario. Simulation results in a realistic mobility trace show that FARID schedules more tasks and has more fairness rate in resource utilization than state-of-the-art solutions. In addition, FARID reduces overall system latency and lowers the monetary cost of using VEC resources. FARID was introduced in detail in Chapter 6.

7.2 Future Work

With the development of this thesis, some points have been observed and can be explored in the future. These points include:

- **Learning Applications**: Evaluation of VCs formed for distributed learning applications, such as Federated Learning and Split Learning [150, 21]. Given that RSU/BS aids the VCs formed with computational capacity and the presence of VEC controllers, learning applications can be used to validate the concept of VEC and process these applications even closer to users who do not have sufficient computational power for heavier learning models.

- **Security**: Security in VEC includes several essential components. Data security and communication security are essential to safeguard enormous amounts of data from illegal access or network attacks. Because sensitive personal data is used, privacy protection is crucial. System and data integrity assurance, secure authentication, and access control techniques are required to thwart hostile alterations. VEC systems must be resilient to cyber threats and secure edge nodes from physical and virtual intruders. Last but not least, efficient incident response techniques are essential for quick action during security breaches, improving VEC systems' overall robustness and dependability [158, 132].

- **Blockchain**: Integrating blockchain into VEC presents advantages. Blockchain's decentralization enhances data integrity and security, reducing the risk of single-point failures and unauthorized data manipulation. Its immutable ledger feature offers a transparent, traceable record of transactions, facilitating data accountability and trust. Blockchain's smart contracts can automate processes, improving efficiency and reliability. It can also aid in creating a trusted and secure environment for V2V and V2I communication. Moreover, it can enable secure, peer-to-peer data trading between vehicles, rewarding those contributing useful data and encouraging active participation in the vehicular network [81, 68].

- **Metaverse**: Incorporating the metaverse concept into VEC could offer unique advantages. The metaverse, a collective virtual shared space, could enhance real-time data sharing and processing in VEC, improving V2V and V2I communication. This could foster a richer, multi-dimensional traffic information system, enhancing navigation and safety. Furthermore, metaverse could facilitate virtual simulations of traffic patterns or vehicle behavior, aiding in better traffic management and optimization of vehicle performance. It might also enable a more immersive, interactive in-vehicle experience for passengers, blending the physical journey with virtual elements for entertainment or information, thereby transforming the overall vehicular experience [62, 74, 5].

- **Satellites**: Incorporating Low Earth Orbit (LEO) satellites into VEC provides numerous advantages. LEO satellites, due to their proximity to Earth, offer lower latency communication, enhancing real-time data exchange between vehicles and the edge infrastructure. This can lead to improved vehicular safety, better traffic management, and more efficient routing. Additionally, LEO satellites can ensure wide and consistent network coverage, including remote or underserved areas, increasing the reach and effectiveness of VEC applications. They can also assist in

load balancing during peak data traffic periods and provide reliable backup communication channels in the event of ground network failures, ensuring consistent VEC performance [51, 52].

7.3 Scientific Production

In the following sections, we present a list of publications produced at the moment of this thesis writing. The list is divided into journals and conference papers (Sections 7.3.1 and 7.3.2). Also, we list the work developed in collaboration (Section 7.3.3) during the doctoral period, with topics that relate directly to those discussed in this thesis.

7.3.1 Work Published in Journals

1. [35] **Joahannes BD da Costa**, Allan M de Souza, Eduardo Cerqueira, Denis Rosário, Christoph Sommer, Leandro Villas. Mobility and Deadline-aware Task Scheduling Mechanism for Vehicular Edge Computing. *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*. 2023. DOI: 10.1109/TITS.2023.3276823.

2. [38] **Joahannes BD da Costa**, Wellington Lobato, Allan M de Souza, Eduardo Cerqueira, Denis Rosário, Christoph Sommer, Leandro Villas. Mobility-aware Vehicular Cloud Formation Mechanism for Vehicular Edge Computing Environments. *Elsevier Ad Hoc Networks*. 2023. DOI: 10.1016/j.adhoc.2023.103300.

3. (Under Review) **Joahannes BD da Costa**, Allan M de Souza, Denis Rosário, Christoph Sommer, Leandro Villas. Fair Process, Fair Outcome: Enhancing Fairness and Load Balancing in Vehicular Edge Computing. *Elsevier Vehicular Communications*. 2023.

7.3.2 Work Published in Conferences

1. [34] **Joahannes BD da Costa**, Allan M de Souza, Wellington Lobato, Denis Rosário, Christoph Sommer, Leandro A Villas. Improving Fairness and Performance in Resource Usage for Vehicular Edge Computing. *IEEE 98th Vehicular Technology Conference (VTC-Fall)*. 2023. To appear.

2. [37] **Joahannes BD da Costa**, Allan M de Souza, Denis Rosário, Christoph Sommer, Leandro A Villas. Efficient pareto optimality-based task scheduling for vehicular edge computing. *IEEE 96th Vehicular Technology Conference (VTC-Fall)*. 2022. DOI: 10.1109/VTC2022-Fall57202.2022.10013029.

3. [32] **Joahannes BD da Costa**, Wellington Lobato, Allan M de Souza, Eduardo Cerqueira, Denis Rosário, Leandro A Villas. NEMESIS: Mecanismo para formação de nuvens veiculares baseado em previsão de mobilidade. *XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. 2022. DOI: 10.5753/sbrc.2022.222309.

4. [36] **Joahannes BD da Costa**, Allan M de Souza, Rodolfo I Meneguette, Eduardo Cerqueira, Denis Rosário, Leandro A Villas. Escalonamento de tarefas ciente de contexto para computação de borda veicular. *XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. 2022. DOI: 10.5753/sbrc.2022.221910.

5. [39] **Joahannes BD da Costa**, Rodolfo I Meneguette, Denis Rosário, Leandro A Villas. Combinatorial optimization-based task allocation mechanism for vehicular clouds. *IEEE 91st Vehicular Technology Conference (VTC-Spring)*. 2020. DOI: 10.1109/VTC2020-Spring48590.2020.9128834.

6. [40] **Joahannes BD da Costa**, Maycon LM Peixoto, Rodolfo I Meneguette, Denis Rosário, Leandro A Villas. MORFEU: Mecanismo Baseado em Otimização Combinatória para Alocação de Tarefas em Nuvens Veiculares. *XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. 2020. DOI: 10.5753/sbrc.2020.12305.

7.3.3 Research Collaborations

The following articles were published during the PhD and, although they did not directly integrate into the Thesis, they indirectly were part of the research trajectory.

1. [31] **Joahannes BD da Costa**, Allan M de Souza, Denis Rosário, Eduardo Cerqueira, and Leandro A Villas. Efficient Data Dissemination Protocol based on Complex Networks' Metrics for Urban Vehicular Networks. *Springer Journal of Internet Services and Applications (JISA)*. 2019. DOI: 10.1186/s13174-019-0114-y.

2. (Submitted) Wellington Lobato, **Joahannes BD da Costa**, Allan M de Souza, Denis Rosário, Christoph Sommer, and Leandro A Villas. Dynamic Semi-Synchronous Federated Learning for Connected Autonomous Vehicles. *16th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*. 2023.

3. [127] Aguimar R Júnior, **Joahannes BD da Costa**, Geraldo P Rocha Filho, Leandro A Villas, Daniel L Guidoni, Sandra Sampaio and Rodolfo I Meneguette. HARMONIC: Shapley Values in Market Games for Resource Allocation in Vehicular Clouds. *Elsevier Ad Hoc Networks*. 2023. DOI: 10.1016/j.adhoc.2023.103224.

4. [92] Wellington Lobato, **Joahannes BD da Costa**, Allan M de Souza, Denis Rosário, Christoph Sommer, and Leandro A Villas. FLEXE: Investigating Federated Learning in Connected Autonomous Vehicle Simulations. *IEEE 96th Vehicular Technology Conference (VTC-Fall)*. 2022. DOI: 10.1109/VTC2022-Fall57202.2022.10012905.

5. [70] Aguimar R Júnior, **Joahannes BD da Costa**, Geraldo P Rocha Filho, Leandro A Villas, Daniel L Guidoni, and Rodolfo I Meneguette. Alocação de Tarefas em Nuvens Veiculares Utilizando Jogos de Mercado. *XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*. 2022. DOI: 10.5753/sbrc.2022.222247.

6. [121] Matheus S Quessada, Douglas D Lieira, **Joahannes BD da Costa**, Geraldo P Rocha Filho, E Robson, and Rodolfo I Meneguette. ARCANE: Algoritmo Meta-heurístico para Alocação de Tarefas em Nuvens Veiculares. *VI Workshop de Computação Urbana (CoUrb)*. 2022. DOI: 10.5753/courb.2022.223498.

7. [87] Douglas D Lieira, Matheus S Quessada, **Joahannes BD da Costa**, Eduardo Cerqueira, Denis Rosário, and Rodolfo I Meneguette. TOVEC: Task Optimization Mechanism for Vehicular Clouds using Meta-heuristic Technique. *International Wireless Communications and Mobile Computing (IWCMC)*. 2021. DOI: 10.1109/IWCMC51323.2021.9498784.

8. [118] Rickson S Pereira, Douglas D Lieira, Marco AC da Silva, Adinovam HM Pimenta, **Joahannes BD da Costa**, Denis Rosário, Leandro Villas, and Rodolfo I Meneguette. RELIABLE: Resource Allocation Mechanism for 5g Network Using Mobile Edge Computing. *MDPI Sensors*. 2020. DOI: 10.3390/s20195449.

9. [129] Lucas B Rondon, **Joahannes BD da Costa**, Geraldo P Rocha Filho, Denis Rosário, and Leandro A Villas. Degree Centrality-based Caching Discovery Protocol for Vehicular Named-data Networks. *IEEE 91st Vehicular Technology Conference (VTC-Spring)*. 2020. DOI: 10.1109/VTC2020-Spring48590.2020.9128557.

10. [116] Rickson S Pereira, Douglas D Lieira, Marco AC da Silva, Adonivam Pimenta, **Joahannes BD da Costa**, Denis Rosario, Rodolfo I Meneguette, and Leandro A Villas. A Novel Fog-based Resource Allocation Policy for Vehicular Clouds in the Highway Environment. *IEEE 11th Latin-American Conference on Communications (LATINCOM)*. 2019. DOI: 10.1109/LATINCOM48065.2019.8937912.

11. [128] Lucas B Rondon, **Joahannes BD da Costa**, Geraldo Pereira, and Leandro A Villas. A Distance and Position-based Caching Discovery Protocol for Vehicular Named-data Networks. *IEEE 11th Latin-American Conference on Communications (LATINCOM)*. 2019. DOI: 10.1109/LATINCOM48065.2019.8938022.

12. [104] Rodolfo I Meneguette, Diego O Rodrigues, **Joahannes BD da Costa**, Denis Rosario, and Leandro A Villas. A Virtual Machine Migration Policy based on Multiple Attribute Decision in Vehicular Cloud Scenario. *IEEE International Conference on Communications (ICC)*. 2019. DOI: 10.1109/ICC.2019.8761248.

13. [72] Wellington Lobato, **Joahannes BD da Costa**, Denis Rosário, Eduardo Cerqueira, and Leandro A Villas. A Comparative Analysis of DSRC and VLC for Video Dissemination in Platoon of Vehicles. *IEEE 10th Latin-American Conference on Communications (LATINCOM)*. 2018. DOI: 10.1109/LATINCOM.2018.8613247.
Best Paper Award.

14. [71] Wellington Lobato, **Joahannes BD da Costa**, Eduardo Cerqueira. Avaliação Exploratória da Disseminação de Vídeos para Aplicações See-through em Redes Veiculares. *XXIII Workshop de Gerência e Operação de Redes e Serviços (WGRS)*. 2018. DOI: 10.1109/LATINCOM.2018.8613247.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *proceedings of the 12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[2] Faezeh Abbasi, Mani Zarei, and Amir Masoud Rahmani. FWDP: A fuzzy logic-based vehicle weighting model for data prioritization in vehicular ad hoc networks. *Vehicular Communications*, 33:100413, January 2022.

[3] Ohoud Alzamzami and Imad Mahgoub. Link utility aware geographic routing for urban vanets using two-hop neighbor information. *Ad Hoc Networks*, 106:102213, 2020.

[4] Ashwin Ashok, Peter Steenkiste, and Fan Bai. Vehicular cloud computing through dynamic computation offloading. *Computer Communications*, 120:125–137, 2018.

[5] Anjum Mohd Aslam, Rajat Chaudhary, Aditya Bhardwaj, Ishan Budhiraja, Neeraj Kumar, and Sherali Zeadally. Metaverse for 6g and beyond: the next revolution and deployment challenges. *IEEE Internet of Things Magazine*, 6(1):32–39, 2023.

[6] Muddasar Ayyub, Alma Oracevic, Rasheed Hussain, Ammara Anjum Khan, and Zhongshan Zhang. A comprehensive survey on clustering in vehicular networks: Current solutions and future challenges. *Ad Hoc Networks*, 124:102729, January 2022.

[7] Leandro N. Balico, Antonio A. F. Loureiro, Eduardo F. Nakamura, Raimundo S. Barreto, Richard W. Pazzi, and Horacio A. B. F. Oliveira. Localization Prediction in Vehicular Ad Hoc Networks. *IEEE Communications Surveys & Tutorials*, 20(4):2784–2803, 2018.

[8] Dmitry Bankov, Evgeny Khorov, Artem Krasilov, and Artem Otmakhov. Analytical model of 5g v2x mode 2 for sporadic traffic. *IEEE Wireless Communications Letters*, 2023.

[9] Roberto Beraldi, Claudia Canali, Riccardo Lancellotti, and Gabriele Proietti Mattia. Randomized Load Balancing under Loosely Correlated State Information in Fog Computing. In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 123–127. ACM, November 2020.

[10] Stephan Borzsony, Donald Kossmann, and Konrad Stocker. The Skyline operator. In *17th International Conference on Data Engineering*, pages 421–430. IEEE, 2001.

[11] Azzedine Boukerche and Noura Aljeri. Design Guidelines for Topology Management in Software-Defined Vehicular Networks. *IEEE Network*, 35(2):120–126, March 2021.

[12] Azzedine Boukerche and Robson Grande. Vehicular cloud computing: Architectures, applications, and mobility. *Computer Networks*, 2018.

[13] Azzedine Boukerche and Rodolfo I Meneguette. Vehicular cloud network: A new challenge for resource management based systems. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*, pages 159–164. IEEE, 2017.

[14] Azzedine Boukerche and Victor Soto. Computation Offloading and Retrieval for Vehicular Edge Computing: Algorithms, Models, and Classification. *ACM Computing Surveys (CSUR)*, 53(4), July 2021.

[15] Bouziane Brik and Adlen Ksentini. Toward Optimal MEC Resource Dimensioning for a Vehicle Collision Avoidance System: A Deep Learning Approach. *IEEE Network*, 35(3):74–80, May 2021.

[16] Bouziane Brik, Nasreddine Lagraa, Nouredine Tamani, Abderrahmane Lakas, and Yacine Ghamri-Doudane. Renting out cloud services in mobile vehicular cloud. *IEEE Transactions on Vehicular Technology*, 2018.

[17] Muhammad Saleh Bute, Pingzhi Fan, Gang Liu, Fakhar Abbas, and Zhiguo Ding. A cluster-based cooperative computation offloading scheme for C-V2X networks. *Ad Hoc Networks*, 132:102862, July 2022.

[18] Chen Chen, Haofei Li, Huan Li, Rufei Fu, Yangyang Liu, and Shaohua Wan. Efficiency and Fairness Oriented Dynamic Task Offloading in Internet of Vehicles. *IEEE Transactions on Green Communications and Networking*, 6(3):1481–1493, September 2022.

[19] Lixing Chen and Jie Xu. Task Replication for Vehicular Cloud: Contextual Combinatorial Bandit with Delayed Feedback. In *IEEE Conference on Computer Communications (INFOCOM 2019)*, pages 748–756. IEEE, April 2019.

[20] Mung Chiang and Tao Zhang. Fog and iot: An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6):854–864, 2016.

[21] Geon Choi, Jeonghun Park, Nir Shlezinger, Yonina C Eldar, and Namyoon Lee. Split-kalmannet: A robust model-based deep learning approach for state estimation. *IEEE Transactions on Vehicular Technology*, 2023.

[22] Hyunseok Choi, Youngju Nam, Yongje Shin, and Euisin Lee. The partial cloud member replacement for reconstructing vehicular clouds in VANETs: Reactive and proactive schemes. *Ad Hoc Networks*, 136:102959, November 2022.

[23] Cisco. White paper: Fog computing and the internet of things: Extend the cloud to where the things are, 2015.

[24] CISCO. Driving Profits from Connected Vehicles. Technical report, CISCO, 2019.

[25] Lara Codeca, Raphaël Frank, Sébastien Faye, and Thomas Engel. Luxembourg sumo traffic (lust) scenario: Traffic demand evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(2):52–63, 2017.

[26] Craig Cooper, Daniel Franklin, Montserrat Ros, Farzad Safaei, and Mehran Abolhasan. A Comparative Survey of VANET Clustering Techniques. *IEEE Communications Surveys & Tutorials*, 19(1):657–681, 2017.

[27] Marius-Iulian Corici, Fabian Eichhorn, Roland Bless, Michael Gundall, Daniel Lindenschmitt, Bastian Bloessl, Marina Petrova, Lara Wimmer, Ronny Kreuch, Thomas Magedanz, and Hans D. Schotten. Organic 6g networks: Vision, requirements, and research approaches. *IEEE Access*, pages 1–1, 2023.

[28] Allan Costa, Lucas Pacheco, Denis Rosário, Leandro Villas, Antonio A. F. Loureiro, Susana Sargento, and Eduardo Cerqueira. Skipping-based Handover Algorithm for Video Distribution Over Ultra-Dense VANET. *Computer Networks*, 176:107252, July 2020.

[29] Rodolfo WL Coutinho and Azzedine Boukerche. Guidelines for the design of vehicular cloud infrastructures for connected autonomous vehicles. *IEEE Wireless Communications*, 26(4):6–11, 2019.

[30] Felipe Cunha, Leandro Villas, Azzedine Boukerche, Guilherme Maia, Aline Viana, Raquel AF Mini, and Antonio AF Loureiro. Data communication in vanets: Protocols, applications and challenges. *Ad Hoc Networks*, 44:90–103, 2016.

[31] Joahannes B. D. da Costa, Allan M. de Souza, Denis Rosário, Eduardo Cerqueira, and Leandro A. Villas. Efficient data dissemination protocol based on complex networks' metrics for urban vehicular networks. *Journal of Internet Services and Applications*, 10(1), August 2019.

[32] Joahannes B. D. da Costa, Wellington V. Lobato J., Allan M. de Souza, Eduardo Cerqueira, Denis Rosário, and Leandro 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)*, pages 280–293. Sociedade Brasileira de Computação, May 2022.

[33] Joahannes B. D. da Costa, Rodolfo I. Meneguette, Denis Rosário, and Leandro A. Villas. Combinatorial Optimization-based Task Allocation Mechanism for Vehicular

Clouds. In *IEEE 91st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, May 2020.

[34] Joahannes BD Da Costa, Allan M de Souza, Wellington Lobato, Denis Rosário, Christoph Sommer, and Leandro A Villas. Improving fairness and performance in resource usage for vehicular edge computing. In *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pages 1–6. IEEE, 2023.

[35] Joahannes BD da Costa, Allan M de Souza, Rodolfo I Meneguette, Eduardo Cerqueira, Denis Rosário, Christoph Sommer, and Leandro Villas. Mobility and deadline-aware task scheduling mechanism for vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, 24(10):11345–11359, 2023.

[36] Joahannes BD da Costa, Allan M de Souza, Rodolfo I Meneguette, Eduardo Cerqueira, Denis Rosário, and Leandro A Villas. Escalonamento de tarefas ciente de contexto para computaçao de borda veicular. In *Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 15–28. SBC, 2022.

[37] Joahannes BD da Costa, Allan M de Souza, Denis Rosário, Christoph Sommer, and Leandro A Villas. Efficient pareto optimality-based task scheduling for vehicular edge computing. In *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, pages 1–6. IEEE, 2022.

[38] Joahannes BD da Costa, Wellington Viana Lobato Junior, Allan Mariano de Souza, E. Cerqueira, Denis Rosário, Christoph Sommer, and Leandro Aparecido Villas. Mobility-aware Vehicular Cloud formation mechanism for Vehicular Edge Computing environments. *Ad Hoc Networks*, 151, December 2023.

[39] Joahannes BD da Costa, Rodolfo I Meneguette, Denis Rosário, and Leandro A Villas. Combinatorial optimization-based task allocation mechanism for vehicular clouds. In *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pages 1–5. IEEE, 2020.

[40] Joahannes BD da Costa, Maycon LM Peixoto, Rodolfo I Meneguette, Denis Lima Rosário, and Leandro Aparecido Villas. Morfeu: Mecanismo baseado em otimizaçao combinatória para alocaçao de tarefas em nuvens veiculares. In *Anais do XXXVIII Simposio Brasileiro de Redes de Computadores e Sistemas Distribuidos*, pages 505–518. SBC, 2020.

[41] Penglin Dai, Kaiwen Hu, Xiao Wu, Huanlai Xing, Fei Teng, and Zhaofei Yu. A Probabilistic Approach for Cooperative Computation Offloading in MEC-Assisted Vehicular Networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):899–911, February 2022.

[42] Mike Daily, Swarup Medasani, Reinhold Behringer, and Mohan Trivedi. Self-driving cars. *Computer*, 50(12):18–23, 2017.

[43] Nishanthi Dasanayaka and Yanming Feng. Analysis of vehicle location prediction errors for safety applications in cooperative-intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):15512–15521, 2022.

[44] Allan M. de Souza, Torsten Braun, Leonardo C. Botega, Leandro A. Villas, and Antonio A. F. Loureiro. Safe and Sound: Driver Safety-Aware Vehicle Re-Routing Based on Spatiotemporal Information. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3973–3989, September 2020.

[45] Allan M. de Souza, Horacio F. Oliveira, Zhongliang Zhao, Torsten Braun, Leandro Villas, and Antonio A. F. Loureiro. Enhancing Sensing and Decision-Making of Automated Driving Systems With Multi-Access Edge Computing and Machine Learning. *IEEE Intelligent Transportation Systems Magazine*, 14(1):44–56, January 2022.

[46] Yonas Abate Debalki, Jin Hou, Hamid Ullah, and Baye Yematawa Adane. Multi-hop data dissemination using a multi-metric contention-based broadcast suppression strategy in vanets. *Ad Hoc Networks*, 140:103070, 2023.

[47] Basem M ElHalawany, Ahmad A Aziz El-Banna, and Kaishun Wu. Physical-layer security and privacy for vehicle-to-everything. *IEEE Communications Magazine*, 57(10):84–90, 2019.

[48] Shuo Feng, Xintao Yan, Haowei Sun, Yiheng Feng, and Henry X. Liu. Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment. *Nature Communications*, 12(1), February 2021.

[49] Weiyang Feng, Siyu Lin, Ning Zhang, Gongpu Wang, Bo Ai, and Lin Cai. Joint c-v2x based offloading and resource allocation in multi-tier vehicular edge computing system. *IEEE Journal on Selected Areas in Communications*, 41(2):432–445, 2022.

[50] Ryan Florin, Puya Ghazizadeh, Aida Ghazi Zadeh, Ravi Mukkamala, and Stephan Olariu. A tight estimate of job completion time in vehicular clouds. *IEEE Transactions on Cloud Computing*, 2018.

[51] Mario Franke, Florian Klingler, and Christoph Sommer. Addressing the Unbounded Latency of Best-Effort Device-to-Device Communication with Low Earth Orbit Satellite Support. In *IEEE Consumer Communications and Networking Conference (CCNC 2023), Track Communication and Applications for Connected and Autonomous Vehicles on Land, Water, and Sky*, pages 823–828, Las Vegas, NV, January 2023. IEEE.

[52] Mario Franke, Roland Stroop, Florian Klingler, and Christoph Sommer. Low Earth Orbit Satellite Supported Multi-Hop Dissemination of Messages in V2X Networks. In *97th IEEE Vehicular Technology Conference (VTC 2023-Spring)*, Florence, Italy, June 2023. IEEE. to appear.

[53] Jian Gao, Zhufang Kuang, Jie Gao, and Lian Zhao. Joint offloading scheduling and resource allocation in vehicular edge computing: A two layer solution. *IEEE Transactions on Vehicular Technology*, pages 1–12, 2022.

[54] Rahul Ghosh, Francesco Longo, Vijay K Naik, and Kishor S Trivedi. Modeling and performance analysis of large scale iaas clouds. *Future Generation Computer Systems*, 29(5):1216–1234, 2013.

[55] Judy C Guevara, Ricardo da S Torres, and Nelson LS da Fonseca. On the classification of fog computing applications: A machine learning perspective. *Journal of Network and Computer Applications*, page 102596, 2020.

[56] Hui Guo, Lan-lan Rui, and Zhi-peng Gao. V2v task offloading algorithm with lstm-based spatiotemporal trajectory prediction model in svcns. *IEEE Transactions on Vehicular Technology*, pages 1–16, 2022.

[57] Ashish Gupta, Hari Prabhat Gupta, Bhaskar Biswas, and Tanima Dutta. A fault-tolerant early classification approach for human activities using multivariate time series. *IEEE Transactions on Mobile Computing*, 20(5):1747–1760, 2020.

[58] Florian Hagenauer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Efficient data handling in vehicular micro clouds. *Ad Hoc Networks*, 91:101871, August 2019.

[59] Florian Hagenauer, Christoph Sommer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Vehicular micro cloud in action: On gateway selection and gateway handovers. *Ad Hoc Networks*, 2018.

[60] Ghaith Hattab, Seyhan Ucar, Takamasa Higuchi, Onur Altintas, Falko Dressler, and Danijela Cabric. Optimized Assignment of Computational Tasks in Vehicular Micro Clouds. In *2nd International Workshop on Edge Systems, Analytics and Networking (EdgeSys 2019)*, pages 1–6. ACM, 2019.

[61] Khaled Hejja, Sara Berri, and Houda Labiod. Network slicing with load-balancing for task offloading in vehicular edge computing. *Vehicular Communications*, 34:100419, April 2022.

[62] Nguyen Tien Hoa, Bui Duc Son, Nguyen Cong Luong, Dusit Niyato, et al. Dynamic offloading for edge computing-assisted metaverse systems. *IEEE Communications Letters*, 2023.

[63] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[64] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of network and computer applications*, 98:27–42, 2017.

[65] Rasheed Hussain and Sherali Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys & Tutorials*, 21(2):1275–1313, 2018.

[66] Tesla Inc. *Model X Owner's Manual*. Tesla Inc. 2021.

[67] André Ip, Luis Irio, and Rodolfo Oliveira. Vehicle Trajectory Prediction based on LSTM Recurrent Neural Networks. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*. IEEE, April 2021.

[68] Bingcheng Jiang, Qian He, Peng Liu, Sabita Maharjan, and Yan Zhang. Blockchain empowered secure video sharing with access control for vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[69] Ying Ju, Yuchao Chen, Zhiwei Cao, Lei Liu, Qingqi Pei, Ming Xiao, Kaoru Ota, Mianxiong Dong, and Victor C. M. Leung. Joint Secure Offloading and Resource Allocation for Vehicular Edge Computing Network: A Multi-Agent Deep Reinforcement Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, 24(5):5555–5569, May 2023.

[70] Aguimar R Júnior, Joahannes BD da Costa, Geraldo P Rocha Filho, Leandro A Villas, Daniel L Guidoni, and Rodolfo I Meneguette. Alocaçao de tarefas em nuvens veiculares utilizando jogos de mercado. In *Anais do XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 210–223. SBC, 2022.

[71] Wellington Lobato Junior, Joahannes Costa, and Eduardo Cerqueira. Avaliação exploratória da disseminação de vídeos para aplicações see-through em redes veiculares. In *Anais do XXIII Workshop de Gerência e Operação de Redes e Serviços*. SBC, 2018.

[72] Wellington Lobato Junior, Joahannes Costa, Denis Rosário, Eduardo Cerqueira, and Leandro A Villas. A comparative analysis of dsrc and vlc for video dissemination in platoon of vehicles. In *2018 IEEE 10th Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE, 2018.

[73] Subramanyan Kamakshi and V. S. Shankar Sriram. Modularity based mobility aware community detection algorithm for broadcast storm mitigation in VANETs. *Ad Hoc Networks*, 104:102161, July 2020.

[74] Supun Karunarathna, Shalitha Wijethilaka, Pasika Ranaweera, Kasun T Hemachandra, Tharaka Samarasinghe, and Madhusanka Liyanage. The role of network slicing and edge computing in the metaverse realization. *IEEE Access*, 11:25502–25530, 2023.

[75] Mostafa Haghi Kashani and Ebrahim Mahdipour. Load Balancing Algorithms in Fog Computing: A Systematic Review. *IEEE Transactions on Services Computing*, 16(2):1505–1521, March 2023.

[76] Abhay Katiyar, Dinesh Singh, and Rama Shankar Yadav. State-of-the-art approach to clustering protocols in VANET: a survey. *Wireless Networks*, 26(7):5307–5336, June 2020.

[77] SM Ahsan Kazmi, Tai Manh Ho, Tuong Tri Nguyen, Muhammad Fahim, Adil Khan, Md Jalil Piran, and Gaspard Baye. Computing on wheels: A deep reinforcement learning-based approach. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):22535–22548, 2022.

[78] John B Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.

[79] DataCenter Knowledge. Report: Google uses about 900.000 servers, 2011.

[80] Stefan Krauß, Peter Wagner, and Christian Gawron. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5):5597, 1997.

[81] Ping Lang, Daxin Tian, Xuting Duan, Jianshan Zhou, Zhengguo Sheng, and Victor CM Leung. Blockchain-based cooperative computation offloading and secure handover in vehicular edge computing networks. *IEEE Transactions on Intelligent Vehicles*, 2023.

[82] Luc Le Mero, Dewei Yi, Mehrdad Dianati, and Alexandros Mouzakitis. A Survey on Imitation Learning Techniques for End-to-End Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[83] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[84] Jiewu Leng, Douxi Yan, Qiang Liu, Kailin Xu, J Leon Zhao, Rui Shi, Lijun Wei, Ding Zhang, and Xin Chen. Manuchain: Combining permissioned blockchain with a holistic optimization model as bi-level intelligence for smart manufacturing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(1):182–192, 2019.

[85] Xi Li, Andres Garcia-Saavedra, Xavier Costa-Perez, Carlos J. Bernardos, Carlos Guimarães, Kiril Antevski, Josep Mangues-Bafalluy, Jorge Baranda, Engin Zeydan, Daniel Corujo, Paola Iovanna, Giada Landi, Jesus Alonso, Paulo Paixao, Hugo Martins, Manuel Lorenzo, Jose Ordonez-Lucena, and Diego 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):84–90, March 2021.

[86] Zhong Li, Cheng Wang, and Chang-Jun Jiang. User association for load balancing in vehicular networks: An online reinforcement learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):2217–2228, 2017.

[87] Douglas D. Lieira, Matheus S. Quessada, Joahannes B. D. da Costa, Eduardo Cerqueira, Denis Rosário, and Rodolfo I. Meneguette. TOVEC: Task Optimization

Mechanism for Vehicular Clouds using Meta-heuristic Technique. In *2021 International Wireless Communications and Mobile Computing (IWCMC)*, pages 358–363. IEEE, June 2021.

[88] Jinshi Liu, Manzoor Ahmed, Muhammad Ayzed Mirza, Wali Ullah Khan, Dianlei Xu, Jianbo Li, Abdul Aziz, and Zhu Han. Rl/drl meets vehicular task offloading using edge and vehicular cloudlet: A survey. *IEEE Internet of Things Journal*, 9(11):8315–8338, 2022.

[89] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. *Mobile Networks and Applications*, pages 1–24, 2020.

[90] Yi Liu, Huimin Yu, Shengli Xie, and Yan Zhang. Deep Reinforcement Learning for Offloading and Resource Allocation in Vehicle Edge Computing and Networks. *IEEE Transactions on Vehicular Technology*, 68(11):11158–11168, November 2019.

[91] Yibing Liu, Lijun Huo, Jun Wu, and Ali Kashif Bashir. Swarm learning-based dynamic optimal management for traffic congestion in 6g-driven intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[92] Wellington Lobato, Joahannes BD Da Costa, Allan M de Souza, Denis Rosário, Christoph Sommer, and Leandro A Villas. Flexe: Investigating federated learning in connected autonomous vehicle simulations. In *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)*, pages 1–5. IEEE, 2022.

[93] Wangchen Long, Tao Li, Zhu Xiao, Dong Wang, Rui Zhang, Amelia C. Regan, Hongyang Chen, and Yongdong Zhu. Location Prediction for Individual Vehicles via Exploiting Travel Regularity and Preference. *IEEE Transactions on Vehicular Technology*, 71(5):4718–4732, May 2022.

[94] Quyuan Luo, Changle Li, Tom Luan, and Weisong Shi. Minimizing the Delay and Cost of Computation Offloading for Vehicular Edge Computing. *IEEE Transactions on Services Computing*, 1374:1–12, 2021.

[95] Naercio Magaia, Pedro Ferreira, Paulo Rogério Pereira, Khan Muhammad, Javier Del Ser, and Victor Hugo 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.

[96] Zineb Mahrez, Essaid Sabir, Elarbi Badidi, Walid Saad, and Mohamed Sadik. Smart urban mobility: When mobility systems meet smart data. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):6222–6239, 2021.

[97] Usman Gidado Manzo, Haruna Chiroma, Nahla Aljojo, Saidu Abubakar, Segun I Popoola, and Mohammed Ali Al-Garadi. A survey on deep learning for steering angle prediction in autonomous vehicles. *IEEE Access*, 2020.

[98] Arooj Masood, Demeke Shumeye Lakew, and Sungrae Cho. Security and Privacy Challenges in Connected Vehicular Cloud Computing. *IEEE Communications Surveys & Tutorials*, 22(4):2725–2764, 2020.

[99] Sarah McClure, Amy Ousterhout, Scott Shenker, and Sylvia Ratnasamy. Efficient scheduling policies for Microsecond-Scale tasks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 1–18, 2022.

[100] Tesnim Mekki, Issam Jabri, Abderrezak Rachedi, and Maher ben Jemaa. Vehicular cloud networks: Challenges, architectures, and future directions. *Vehicular Communications*, 9:268–280, 2017.

[101] Peter M Mell and Timothy Grance. Sp 800-145. the nist definition of cloud computing. *National Institute of Standards & Technology*, 2011.

[102] Rodolfo Meneguette, Robson De Grande, Jo Ueyama, Geraldo P. Rocha Filho, and Edmundo Madeira. Vehicular Edge Computing: Architecture, Resource Management, Security, and Challenges. *ACM Computing Surveys (CSUR)*, 55(1), January 2023.

[103] Rodolfo I Meneguette and Azzedine Boukerche. A cooperative and adaptive resource scheduling for vehicular cloud. In *Computers and Communications (ISCC), 2017 IEEE Symposium on*, pages 398–403. IEEE, 2017.

[104] Rodolfo I. Meneguette, Diego O. Rodrigues, Joahannes B. D. da Costa, Denis Rosario, and Leandro A. Villas. A Virtual Machine Migration Policy Based on Multiple Attribute Decision in Vehicular Cloud Scenario. In *IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, May 2019.

[105] Microsoft. Plataforma de veículos conectados, 2017.

[106] Suchintan Mishra, Manmath Narayan Sahoo, Sambit Bakshi, and Joel J. P. C. Rodrigues. Dynamic Resource Allocation in Fog-Cloud Hybrid Systems Using Multicriteria AHP Techniques. *IEEE Internet of Things Journal*, 7(9):8993–9000, September 2020.

[107] Sudip Misra and Samaresh Bera. Soft-VAN: Mobility-Aware Task Offloading in Software-Defined Vehicular Network. *IEEE Transactions on Vehicular Technology*, 69(2):2071–2078, February 2020.

[108] Mahmudun Nabi, Robert Benkoczi, Sherin Abdelhamid, and Hossam S Hassanein. Resource assignment in vehicular clouds. In *IEEE International Conference on Communications (ICC)*, pages 1–6. Ieee, 2017.

[109] Eugene Ndiaye, Tam Le, Olivier Fercoq, Joseph Salmon, and Ichiro Takeuchi. Safe grid search with optimal complexity. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4771–4780. PMLR, 2019.

[110] Tri DT Nguyen, Van-Nam Pham, Luan NT Huynh, Md Delowar Hossain, Eui-Nam Huh, et al. Modeling data redundancy and cost-aware task allocation in mec-enabled internet-of-vehicles applications. *IEEE Internet of Things Journal*, 2020.

[111] Stephan Olariu. A survey of vehicular cloud research: Trends, applications and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 21(6):2648–2663, 2019.

[112] Stephan Olariu, Tihomir Hristov, and Gongjun Yan. The next paradigm shift: from vehicular networks to vehicular clouds, 2013.

[113] Gurjashan Singh Pannu, Seyhan Ucar, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Dwell time estimation at intersections for improved vehicular micro cloud operations. *Ad Hoc Networks*, 122:102606, November 2021.

[114] Gurjashan Singh Pannu, Seyhan Ucar, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Vehicular Virtual Edge Computing using Heterogeneous V2V and V2C Communication. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–2. IEEE, May 2022.

[115] Maycon Leone Maciel Peixoto, Adriano HO Maia, Edson Mota, E Rangel, Daniel G Costa, Damla Turgut, and Leandro Aparecido Villas. A traffic data clustering framework based on fog computing for VANETs. *Vehicular Communications*, 31:100370, October 2021.

[116] R. Pereira, D. Lieira, M. da Silva, A. Pimenta, Joahannes B. D. da Costa, D. L. Rosario, R. I. Meneguette, , and Leandro A. Villas. A novel fog-based resource allocation policy for vehicular clouds in the highway environment. In *2019 IEEE 11th Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE, 2019.

[117] Rickson Pereira, Azzedine Boukerche, Marco A. C. da Silva, Luis H. V. Nakamura, Heitor Freitas, Geraldo P. Rocha Filho, and Rodolfo I. Meneguette. FORESAM—FOG Paradigm-Based Resource Allocation Mechanism for Vehicular Clouds. *Sensors*, 21(15):5028, July 2021.

[118] Rickson S Pereira, Douglas D Lieira, Marco AC da Silva, Adinovam HM Pimenta, Joahannes BD da Costa, Denis Rosário, Leandro Villas, and Rodolfo I Meneguette. Reliable: Resource allocation mechanism for 5g network using mobile edge computing. *Sensors*, 20(19):5449, 2020.

[119] Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito, and Omer Rana. Fog computing for the internet of things: A survey. *ACM Transactions on Internet Technology (TOIT)*, 19(2):1–41, 2019.

[120] Qualcomm. Connecting vehicles to everything, 2018.

[121] Matheus S Quessada, Douglas D Lieira, Joahannes BD da Costa, Geraldo P Rocha Filho, E Robson, and Rodolfo I Meneguette. Arcane: Algoritmo metaheurístico para alocação de tarefas em nuvens veiculares. In *Anais do VI Workshop de Computação Urbana*, pages 140–153. SBC, 2022.

[122] Iftikhar Rasheed and Fei Hu. Intelligent super-fast Vehicle-to-Everything 5G communications with predictive switching between mmWave and THz links. *Vehicular Communications*, 27:100303, January 2021.

[123] Salman Raza, Shangguang Wang, Manzoor Ahmed, and Muhammad Rizwan Anwar. A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. *Wireless Communications and Mobile Computing*, 2019, 2019.

[124] Zeineb Rejiba, Xavier Masip-Bruin, and Eva Marín-Tordera. A Survey on Mobility-Induced Service Migration in the Fog, Edge, and Related Computing Paradigms. *ACM Computing Surveys (CSUR)*, 52(5), September 2020.

[125] Ju Ren, Deyu Zhang, Shiwen He, Yaoxue Zhang, and Tao Li. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys (CSUR)*, 52(6):1–36, 2019.

[126] Aguimar Ribeiro, Geraldo P. Rocha Filho, Daniel L. Guidoni, Robson E. de Grande, Sandra Sampaio, and Rodolfo I. Meneguette. A Shapley Value-based Strategy for Resource Allocation in Vehicular Clouds. In *IEEE Global Communications Conference (GLOBECOM 2022)*, pages 5801–5806. IEEE, December 2022.

[127] Aguimar Ribeiro Jr, Joahannes BD da Costa, Geraldo P Rocha Filho, Leandro A Villas, Daniel L Guidoni, Sandra Sampaio, and Rodolfo I Meneguette. Harmonic: Shapley values in market games for resource allocation in vehicular clouds. *Ad Hoc Networks*, page 103224, 2023.

[128] Lucas B. Rondon, Joahannes B. D. da Costa, Geraldo Pereira, and Leandro A. Villas. A distance and position-based caching discovery protocol for vehicular named-data networks. In *2019 IEEE 11th Latin-American Conference on Communications (LATINCOM)*, pages 1–6. IEEE, 2019.

[129] Lucas B Rondon, Joahannes BD da Costa, Geraldo P Rocha Filho, Denis Rosário, and Leandro A Villas. Degree centrality-based caching discovery protocol for vehicular named-data networks. In *2020 IEEE 91st vehicular technology conference (VTC2020-spring)*, pages 1–5. IEEE, 2020.

[130] Petra Schwerin and Gerhard Wäscher. The Bin-Packing Problem: A Problem Generator and Some Numerical Experiments with FFD Packing and MTP. *International Transactions in Operational Research*, 4(5–6):377–389, November 1997.

[131] Vivek Sethi and Sujata Pal. Feddove: A federated deep q-learning-based offloading for vehicular fog computing. *Future Generation Computer Systems*, 141:96–105, 2023.

[132] Shanil Sharma, Dheerendra Mishra, and Saurabh Rana. A privacy oriented authorized key agreement framework for vehicular edge computing. *Security and Privacy*, 6(1):e277, 2023.

[133] SINDIPECAS. Relatório da frota circulante edição de 2023, 2023.

[134] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing (TMC)*, 10(1):3–15, January 2011.

[135] Ibrahim Sorkhoh, Dariush Ebrahimi, Ribal Atallah, and Chadi Assi. Workload Scheduling in Vehicular Networks With Edge Cloud Capabilities. *IEEE Transactions on Vehicular Technology*, 68(9):8472–8486, September 2019.

[136] Gang Sun, Liangjun Song, Hongfang Yu, Victor Chang, Xiaojiang Du, and Mohsen Guizani. V2V Routing in a VANET Based on the Autoregressive Integrated Moving Average Model. *IEEE Transactions on Vehicular Technology*, 68(1):908–922, January 2019.

[137] Peng Sun, Noura Aljeri, and Azzedine Boukerche. Machine Learning-Based Models for Real-time Traffic Flow Prediction in Vehicular Networks. *IEEE Network*, 34(3):178–185, May 2020.

[138] Yuxuan Sun, Sheng Zhou, and Zhisheng Niu. Distributed Task Replication for Vehicular Edge Computing: Performance Analysis and Learning-Based Algorithm. *IEEE Transactions on Wireless Communications*, 20(2):1138–1151, February 2021.

[139] Tarik Taleb, Konstantinos Samdanis, Badr Mada, Hannu Flinck, Sunny Dutta, and Dario Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681, 2017.

[140] Fengxiao Tang, Bomin Mao, Nei Kato, and Guan Gui. Comprehensive survey on machine learning in vehicular network: Technology, applications and challenges. *IEEE Communications Surveys & Tutorials*, 23(3):2027–2057, 2021.

[141] DXC Technology. Racing to build autonomous cars. Technical report, DXC Technology, 2019.

[142] Xiang Tian, Baoxian Zhang, and Cheng Li. Throughput-Optimal Dynamic Broadcast for SINR-Based Multi-Hop Wireless Networks With Time-Varying Topology. *IEEE Transactions on Vehicular Technology*, 70(11):11962–11975, November 2021.

[143] Abdul Waheed, Munam Ali Shah, Abid Khan, Saif ul Islam, Suleman Khan, Carsten Maple, and Muhammad Khurram Khan. Volunteer Computing in Connected Vehicles: Opportunities and Challenges. *IEEE Network*, 34:212–218, 2020.

[144] Jingjing Wang, Chunxiao Jiang, Kai Zhang, Tony QS Quek, Yong Ren, and Lajos Hanzo. Vehicular sensing networks in a smart city: Principles, technologies and applications. *IEEE Wireless Communications*, 25(1):122–132, 2018.

[145] Junhua Wang, Tingting Liu, Kai Liu, BaekGyu Kim, Jiang Xie, and Zhu Han. Computation offloading over fog and cloud using multi-dimensional multiple knapsack problem. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[146] Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, James Tu, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. *16th European Conference on Computer Vision*, 2020.

[147] Wei Wang, Feng Xia, Hansong Nie, Zhikui Chen, Zhiguo Gong, Xiangjie Kong, and Wei Wei. Vehicle Trajectory Clustering Based on Dynamic Representation Learning of Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22(6):3567–3576, June 2021.

[148] Xiaojie Wang, Zhaolong Ning, Song Guo, and Lei Wang. Imitation Learning Enabled Task Scheduling for Online Vehicular Edge Computing. *IEEE Transactions on Mobile Computing*, 21(2):598–611, February 2022.

[149] Alan Weissberger. Juniper research: 5g connectivity opportunity for the connected car market, 2023.

[150] Honghai Wu, Jichong Jin, Huahong Ma, and Ling Xing. Federation-based deep reinforcement learning cooperative cache in vehicular edge networks. *IEEE Internet of Things Journal*, 2023.

[151] Xianjing Wu, Shengjie Zhao, Rongqing Zhang, and Liuqing Yang. Mobility Prediction-Based Joint Task Assignment and Resource Allocation in Vehicular Fog Computing. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, May 2020.

[152] Xiaobei Yan, Maode Ma, and Rong Su. Efficient group handover authentication for secure 5g-based communications in platoons. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[153] Wenyuan Zeng, Shenlong Wang, Renjie Liao, Yun Chen, Bin Yang, and Raquel Urtasun. Dsdnet: Deep structured self-driving network. *16th European Conference on Computer Vision*, 2020.

[154] Jie Zhang, Hongzhi Guo, Jiajia Liu, and Yanning Zhang. Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution. *IEEE Transactions on Vehicular Technology*, 69(2):2092–2104, February 2020.

[155] Shangwei Zhang, Jiajia Liu, Hongzhi Guo, Mingping Qi, and Nei Kato. Envisioning device-to-device communications in 6g. *IEEE Network*, 34(3):86–91, 2020.

[156] Haitao Zhao, Jiawen Tang, Bamidele Adebisi, Tomoaki Ohtsuki, Guan Gui, and Hongbo Zhu. An Adaptive Vehicle Clustering Algorithm Based on Power Minimization in Vehicular Ad-Hoc Networks. *IEEE Transactions on Vehicular Technology*, 71(3):2939–2948, March 2022.

[157] Pan Zhao, Lei Feng, Peng Yu, Wenjing Li, and Xuesong Qiu. A Social-Aware Resource Allocation for 5G Device-to-Device Multicast Communication. *IEEE Access*, 5:15717–15730, 2017.

[158] Hong Zhong, Li Wang, Jie Cui, Jing Zhang, and Irina Bolodurina. Secure edge computing-assisted video reporting service in 5g-enabled vehicular networks. *IEEE Transactions on Information Forensics and Security*, 2023.

[159] Sheng Zhou, Yuxuan Sun, Zhiyuan Jiang, and Zhisheng Niu. Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks. *IEEE Communications Magazine*, 57(5):49–55, 2019.