Mobility-Oriented Data Retrieval for Computation Offloading in Vehicular Edge Computing.

by

Victor Soto Garcia

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science (EECS)
Faculty of Engineering
University of Ottawa

Glossary of Terms

CO Computation Offloading

CPU Central Processing Unit

DBF Distance-based Forwarding protocol

DSRC Dedicated Short-Range Communication

EC Edge Computing

FC Fog Computing

GPS Geographical Positioning System

GF Geographic Forwarding

IoT Internet of Things

ITS Intelligent Transportation Systems

LAN Local Area Network

MCC Mobile Cloud Computing

MEC Mobile Edge Computing

MPR Mobility Prediction Retrieval protocol

QoE Quality of Experience

QoS Quality of Service

REPRO Result Retrieval Protocol

RSU Road Side Unit

V2X Vehicle-to-Anything Communication

V2V Vehicle-to-Vehicle Communication

V2I Vehicle-to-Infrastructure Communication

VANET Vehicular Networks

VCC Vehicular Cloud Computing

VCO Vehicle-assisted Computation Offloading protocol

VM Virtual Machine

WAN Wide Area Network

TP Topology protocol

Abstract

Vehicular edge computing (VEC) brings the cloud paradigm to the edge of the network, allowing nodes such as Roadside Units (RSUs) and On-Board Units (OBUs) in vehicles to perform services with location awareness and low delay requirements. Furthermore, it alleviates the bandwidth congestion caused by the large amount of data requests in the network. One of the major components of VEC, computation offloading, has gained increasing attention with the emergence of mobile and vehicular applications with high-computing and low-latency demands, such as Intelligent Transportation Systems and IoT-based applications. However, existing challenges need to be addressed for vehicles' resources to be used in an efficient manner. The primary challenge consists of the mobility of the vehicles, followed by intermittent or lack of connectivity. Therefore, the MPR (Mobility Prediction Retrieval) data retrieval protocol proposed in this work allows VEC to efficiently retrieve the output processed data of the offloaded application by using both vehicles and road side units as communication nodes. The developed protocol uses geo-location information of the network infrastructure and the users to accomplish an efficient data retrieval in a Vehicular Edge Computing environment. Moreover, the proposed MPR Protocol relies on both Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication to achieve a reliable retrieval of data, giving it a higher retrieval rate than methods that use V2I or V2V only. Finally, the experiments performed show the proposed protocol to achieve a more reliable data retrieval with lower communication delay when compared to related techniques.

Acknowledgements

I would like to thank my supervisor, Prof. Azzedine Boukerche, for the patient guidance, encouragement, advice, and for the continuous financial support he has provided throughout my time as his student.

To PARADISE laboratory members. Dr. Robson De Grande, thank you for the discussion sessions we have had.

To my wife, who gave me nothing but support throughout these challenging times.

Publications Related to Thesis

Victor Soto, Robson E. De Grande, and Azzedine Boukerche. 2017. "Repro: Time-constrained Data Retrieval for Edge Offloading in Vehicular Clouds". In Proceedings of the 14th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks (PE-WASUN '17). ACM, Miami, FL, USA, 21-25.
DOI=http://dx.doi.org/10.1145/3134829.3134834

A. Boukerche, and V Soto Garcia, "Computation Offloading in Vehicular Edge Computing", submitted to ACM Computing Surveys.

A. Boukerche and V. Soto, "Mobility-Oriented Retrieval Protocol for Computation Offloading in VEC", submitted to IEEE Transactions on Intelligent Transportation Systems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Vehicular Edge Computing (VEC) allows resource-limited devices, such as mobile phones, to execute compute-intensive applications by using the available processing capabilities of cloud servers through Computation Offloading (CO). CO refers to the process of partitioning an application into smaller "offloadable" tasks [51], sending them to the cloud and then retrieving the output processed data. This approach helps devices achieve the computation with lower energy consumption and lower delay. However, the increasing number of devices requesting and sending data to the core layer of the cloud results in performance drawbacks such as bandwidth congestion, which leads to delay, packet loss and blocking of new connections.

Therefore, edge computing aims to alleviate the bandwidth congestion and give a performance boost to cloud computing by processing data at the edge of the network. Edge computing architectures [13] propose the use of nodes at the edge of the network for computing instead of processing all the data in the back end of the network. These edge nodes are smaller servers that are strategically located in places of interest. Considering the recent development in vehicular industry and the capability of vehicles to communicate with Road-Side Units (RSUs), in this context, using vehicles as support infrastructure to form VEC is a viable approach [42, 85] to provide a performance boost to computation offloading. By using the nodes at the edge of the network, instead of sending them to a data center or cloud, decreases the distance traveled by the data packets. Therefore, the delay is reduced and the network congestion is alleviated to avoid issues such as data loss.

In addition, vehicular networks generally rely on three different types of communication: Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), and Vehicle-to-Anything

(V2X) communication. However, each type of vehicular communication results in benefits and challenges for VEC. For instance, V2V communication helps to alleviate infrastructured networks and reduce the communication delay. However, the output retrieval from the computing offloading will be challenging as the high mobility can lead to losses of connectivity. Concerning V2I communication, the RSU improves connectivity performance using more stable access points with less mobility, but it might be congested due to the high volume of data. In regards of V2X, although it is the most versatile way of communication it has challenges for managing communication among heterogeneous entities.

The primary challenge in this situation is related to the mobility and connectivity of the nodes. This retrieval strategy starts from the assumption that when the vehicle to which a task was assigned finishes the computation, it will want to return the processed data to the sender RSU. Depending on some parameters, such as the speed and the processing time of the task, the vehicle's position might be different from the position it occupied when the task was received. The proposed technique to address these challenges extends our previous work [80]. The algorithm used in this work, for the proposed MPR data retrieval protocol, is influenced by geographic routing and utilizes the position and direction of movement of the user at the moment of offloading the tasks in order to estimate its future position for a more reliable and efficient data retrieval in vehicular edge computing. To address the challenges caused due to the mobility of vehicles, a hybrid of V2V and V2I communication is used, which has proven to be more reliable than the techniques that use V2V or V2I only.

1.1 Thesis Statement

The computation offloading technique is a robust solution that gives devices with low resources the ability to execute compute-intensive applications, such as image rendering, games, and live video streaming in social media. With the quick development and evolution in technology, the number of existing mobile devices and applications keeps increasing, opening some challenges to computation offloading. Bandwidth congestion and connectivity between mobile device and server are the primary ones. Using nodes at the edge of the network, such as vehicles which commonly have underused resources, is a viable approach for alleviating the bandwidth workload. However, challenges such as high mobility, and short-lived or intermittent connectivity between them makes computing offloading to and task retrieval from vehicular edge computing a daunting task. There-

fore, this work proposes an efficient data retrieval protocol for a Mobile Edge Computing scenario using vehicles as support infrastructure, which adapts to a realistic scenario and addresses both of the above-mentioned challenges.

1.2 Motivation

Edge computing aims to alleviate the congestion on the core layer of the cloud networks or data centers. With the development of smart vehicles and applications such as Intelligent Transportation Systems, achieving a seamless communication between these nodes is a prominent research topic. However, challenges exist due to the high-mobility of vehicles and intermittent connectivity between them. Although there are existing studies on Vehicle-to-Anything communication, there is limited research on V2X towards computation offloading. Considering devices with high mobility, such as vehicles with On-Board Units, as infrastructure for applications like Computation Offloading is a viable option. However, offloading and retrieving data involving high-mobility nodes makes it a daunting task. Therefore, a forwarding technique for the retrieval of processed data is of vital importance. The systems proposed shall use vehicles in a Mobile Edge Computing environment as support infrastructure, helping alleviate the load on the core cloud servers, reduce communication delay for CO localized tasks, and to assist on the retrieval of the output processed data.

1.3 Objectives

As this thesis aims to propose to enhance the computation offloading paradigm incorporating new technologies, such as using vehicles as support infrastructure, some parameters need to be taken into consideration. The more critical one is the mobility of the nodes because not only it makes the retrieval process more complex, but it also brings other challenges like intermittent or null connectivity from the side of the vehicles. Apart from this, there is also the case of obstacles in communication when considering an urban scenario, which makes necessary to come up with a solution that can apply to a realistic scenario. To achieve this aim, the objective of this work is to collect information about current protocols that can be used with the purpose of data retrieval, and find an efficient way to retrieve the processed data, while considering the challenges of connectivity, delay, and successful retrieval.

1.4 Contribution

The main contribution of this thesis consists of a new algorithm for a data retrieval protocol that aims to address mobility and intermittent connectivity in a vehicular edge architecture, development and integration of a movement prediction algorithm to provide a more efficient and reliable retrieval, and an evaluation through simulation of different retrieval protocols to show the proposed MPR protocol achieves a more reliable retrieval of the processed data to give the best overall performance.

1.5 Outline

The thesis is organized as follows, Chapter 2 shows the background and related work. The background work describes the process of computation offloading, as well as applications related to this and other topics involved. Chapters 3 and 4 describe the proposed data retrieval protocols used in the scenario presented for the analysis. Chapter 5 illustrates the scenario used in the experiments. The analysis of the results obtained is also discussed in this chapter. Finally, Section 6 concludes this work with the key findings and future work.

Chapter 2

Background and Related Work

Mobile Cloud Computing (MCC) uses the computation offloading (CO) technique to assist devices on executing compute-intensive applications. To achieve this, CO is generally divided in three main steps, partitioning, resource allocation or scheduling, and offloading, where partitioning consists in the division of the application into smaller chunks, scheduling concerns with resource allocation for processing of the chunks that will be offloaded and offloading comprises the transfer of the application's chunks to be processed in a cloud server. Therefore, computing offloading helps to reduce computation delay and conserve energy in resource-constrained mobile devices. However, the proliferation of smart mobile devices and the increased demand for computing intensive application in these devices will lead to huge amount of data offloading to the cloud, resulting in drawbacks for the system such as poor performance for the mobile application and network congestion, which leads to communication delay, packet loss and blocking of new connections [17, 18, 90].

In this context, Mobile Edge Computing (MEC) appears as a viable solution to reduce the amount of data to be transfered for cloud servers. Computation in the edge takes advantage of nodes, like traffic light controllers or Road-side Units (RSUs), to alleviate the load from the cloud servers in the core layer. By bringing the processing nodes to the edge (i.e. closer to the user) communication times are decreased. Therefore, computation cost decreases making the CO technique a more efficient one. Moreover, as data can be analyzed at the edge, network traffic to a data center or cloud is reduced as well.

Recently, the development in the automotive industry has brought a considerable improvement to vehicles in terms of storage and processing power, and with vehicles spending a significant amount of time idle (parked), most of these resources are not

exploited. Several works [42, 85] have evaluated the possibility of using vehicles as infrastructure by creating Cloud servers out of vehicle clusters. If the vehicles are static (e.g. in a parking lot), it becomes logic to consider them as Roadside Units (RSUs). Moreover, when moving on the streets, vehicles' resources still could be reached by V2V or V2I communication. Consequently, connected vehicles have enabled very promising solutions in assisting edge computing with underutilized, capable resources of vehicles. This strategy allows decreasing the workload of cloud servers at the network's core layer by sharing tasks and assembling cloud clusters with vehicles at the edge of the network [88]. Vehicular Edge Computing (VEC), composed by these vehicle cloud clusters, aims to use these idle resources and support resource-constrained devices using the computation offloading technique and it can facilitate many applications in the context of traffic efficiency and management [4], active road safety [4], and infotainment [11], for instance.

In this section, we study proposed solutions for computation offloading in vehicular edge. We propose to classify task partition techniques in filter-based and automatic, based on the parameters each technique aims to improve. Moreover, we classify scheduling algorithms in adaptive, social-aware-based and deadline-sensitive, based on the environment the methods are considered for. Finally, the data retrieval techniques are classified in distance-based, social-based and mobility prediction-based algorithms, based the distance references that the methods use to achieve an efficient retrieval. There are different approaches that have been taken for classifying the process of computation offloading [7, 69]. However, to the best of our knowledge there is no literature that classifies the works for computation offloading in Vehicular Edge Computing, which is essential to reveal the relationship between connectivity, communication capability, and mobility, as well as the benefits of exploiting vehicular resources.

The remainder of this chapter is organized as follows. A background to computation offloading, the process it follows and existing literature on platforms that take advantage of the CO technique is presented in Section 2.1. The classification proposed of existing techniques for partitioning, scheduling, and data retrieval for CO in a VEC environment are discussed in Section 2.3, 2.4, and 2.5, respectively. Furthermore, existing techniques that can be adapted for computation offloading in a vehicular environment are discussed in Section 2.6. Finally, Section 2.7 gives the concluding remarks.

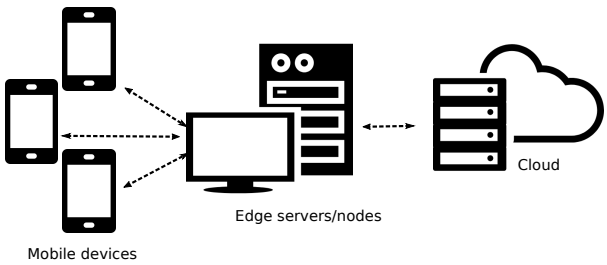


Figure 2.1: Mobile Edge Computing architecture.

2.1 Computation Offloading in a Nutshell

In this section, we will discuss the main building blocks for computing offloading from mobile devices to vehicular edge computing. First, we highlight the three main steps for computing offloading of a computation intensive task. After that, we discuss the potential of vehicular edge computing for reducing computing offloading for cloud servers and, consequently, network overhead.

With the development of the technology of mobile devices, applications are evolving into the creation of more and more compute-intensive applications to give the user a better experience. For example, in the case of gaming and social media, devices are capable of rendering video streaming [14] with such good quality graphics. While all of these applications may improve the user experience, there is a disadvantage to their execution; the high consumption of resources on the mobile devices side. Moreover, the increasing usage of mobile devices presents the possibility of an overload in the bandwidth of the network.

Devices using the computation offloading model and sending significant amounts of data to the core layer of the network can create bandwidth congestion. Mobile Edge Computing (MEC) extends the cloud paradigm by processing data in edge nodes, which are located at the edge of the network as shown in Figure 2.1. This approach alleviates the congestion in the bandwidth and at the same time reduces the communication cost by reducing the delay in the data retrieval process.

A potential approach for this disadvantages is computation offloading. Figure 2.2 shows the main components and process of computation offloading, where instead of completely executing the application locally, it allows the device to transfer part of the tasks to a remote server. This approach aims to achieve a reduced execution time and energy consumption from the overall process.

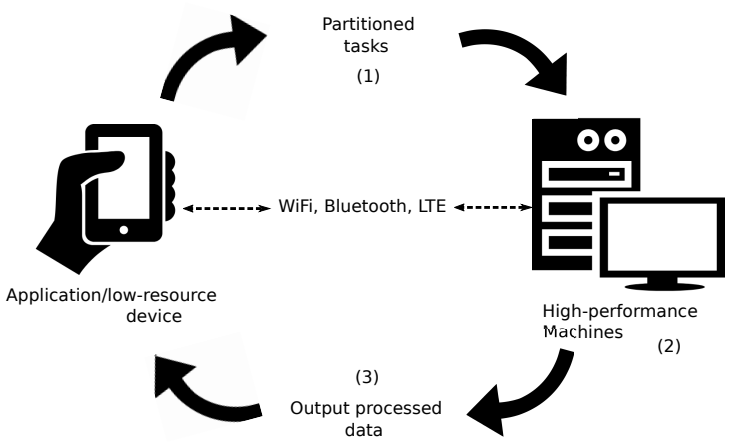


Figure 2.2: Typical computation offloading architecture.

The major steps for computation offloading in a vehicular edge computing environment, represented in Figure 2.2, are regarded as follows:

1. *Partitioning*, the application is sliced into smaller offloadable tasks, usually on the mobile device. This step of the process may seem unaffected by the fact that a high-mobility environment is considered. However, if the partitioning is done efficiently according to the constant changing of nodes and resources, the following steps can have a better performance.

2. *Scheduling*, the server considers the available resources and policies to process the offloaded tasks in the most efficient way. Considering the mobility of nodes in the case of a vehicular edge computing system, resource allocation becomes a more complex process. Therefore, a scheduling algorithm that can adapt to such environment can result in an enhanced QoS.

3. *Data Retrieval*, where the output processed data is collected by the mobile device. The fact that the offloading is in an environment with high-mobility makes this step an essential one. While the two previous steps can help to achieve a quicker task execution, there are still challenges to consider for the retrieval, such as intermittent connectivity and message redundancy leading to a broadcast storm.

Based on the MEC architecture (Figure 2.1) and following the process of computation offloading (Figure 2.2) it is a viable option to consider vehicles as high-performance ma-

chines, which are most of the time idle and have unused resources. Compared to MEC, using vehicles as support infrastructure in an edge computing (VEC) is less costly, considering there is no need for deployment. Moreover, considering the capability of vehicles to communicate with RSUs, which are at the edge of the network, using vehicles as processing nodes greatly improves resource utilization and performance of MEC by reducing delay and capacity limitations when data traffic rises to create network congestions.

2.1.1 Vehicular Edge Computing

Vehicular edge computing is an extension of cloud computing that aims to alleviate the overload from the core cloud servers and lower the offloading cost of mobile applications. Figure 2.3 represents a typical architecture for VEC, where the resources of vehicles are exploited as edge nodes.

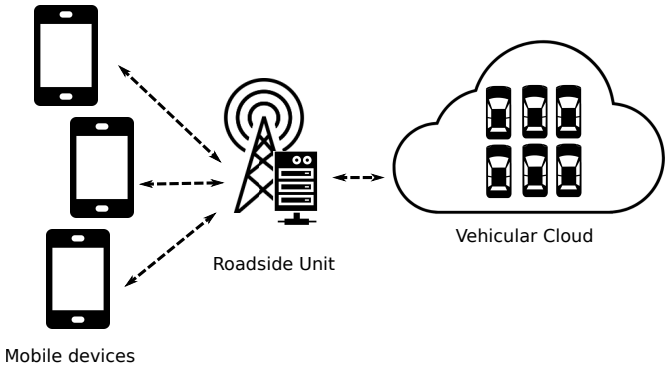


Figure 2.3: General architecture for computation offloading in vehicular networks.

The three layers of the VEC architecture can be explained as follows.

- *Mobile devices:*comprise applications being executed by users, which can be pedestrians or vehicles with onboard units
- *Roadside Units:*represent the fixed infrastructure of the network that receive and process the tasks offloaded by the mobile devices.
- *Vehicular cloud:* is a grid formed by vehicles capable of communication with each other and an RSU, which acts as controller node of the cloud

The main objective of VEC is for resource-constrained mobile devices to offload applications to vehicular nodes in the edge of the network, in order to alleviate the congestion generated in the core cloud servers. The computation offloading to vehicular edge computing, instead of the cloud, results in energy conservation in the mobile devices and reduction of networking congestion [9]. Several studies were proposed in the literature to address different aspects for enabling edge computing in vehicular networks.

In an attempt to alleviate the load in cellular networks, [91] presents a mathematical framework to study time and space constrained data in a Vehicular Ad-hoc Network (VANET) scheme. Considering vehicular clouds are relatively new, there is still a number of challenges to address. The mathematical framework presented here contemplates the probability of contact between vehicles, so they can share content. Even if several users subscribe to the same content, they have different delivery deadlines, which makes such a solution possible. The analysis and experiments prove this framework to perform well and achieve high offloading efficiency.

Regarding offloading in VEC, [8] studies different classical assignment policies together with their performance and task failure, the policies studied were: best fit, first fit, and last fit. The analysis shows that last fit minimizes the total task failures, and best fit achieves a better task assignment failure rate.

A cloud offloading framework for VANET, which aims to enhance the performance of this technique by using different scheduling policies to adapt for interactive applications is presented in [10]. Three policies are used for the study and each aim for a different objective: minimize CPU expense, minimize network expense and minimize the maximum lateness of applications. This solution for offloading interactive applications is tested using a prototype of a cloud system, it shows to efficiently adapt scheduling while applications meet their deadlines.

Concerning location-aware content is presented in [55], it focuses on caching architectures for highly localized content and the impact that growing traffic demand has on them. This work uses an app, which collects location and connectivity information from the user. The assumptions of the architecture are as follows. The app can be installed on a mobile device, the user can be a static or moving pedestrian, either walking or inside a vehicle, and the content is categorized in types. The analysis presented shows that the cache sizes are influenced by the architecture used, and the cost of moving from a centralized to a distributed architecture is not significant [20, 21]. This study shows that caching architectures can be adapted to the edge computing scheme. However, a more thorough analysis of different communication technologies is essential to have a more

clear overview. Working with delay-tolerant data, a system with offloading spots for data storage is proposed in [12], it uses strategic locations for vehicles to offload massive delay-tolerant data to obtain a scalable and centralized offloading architecture.

Furthermore, in the context of VEC, a study [72] of virtual machine (VM) migration, together with its challenges in performance. This work presents an algorithm for VM migration for vehicular clouds, which detects an exiting vehicle from the grid. The grid refers to vehicles in communication with an RSU, and exiting vehicle means a vehicle that is leaving this grid. To avoid data loss, it must be retained within the grid. Therefore, if an exiting vehicle is not able to find a destination vehicle for the data migration then it is stored in the RSU. The authors propose three different modifications to this algorithm: start looking for a destination vehicle at street level, select the destination vehicle based on the workload of the nodes, and select the destination vehicle based on the time they have left in the grid. While the idea presented is promising, the analysis showed that the results are still not satisfactory for an efficient system.

Despite the significant advances encountered in the literature, there still are several challenges to be addressed towards the fully development of vehicular edge computing. The primary challenge in VEC is communication between nodes for task offloading and retrieval, due to the fact that mobility is highly present and topologies are constantly changing. Considering there are no standard parameters yet to evaluate performance in vehicular clouds, using vehicles as infrastructure show a great potential to boost the performance of computation and communication for an edge environment [42]. The study in [85], shows there is a direct relation between the delay tolerance, connectivity, the mobility of the vehicles, and the serviceability of VEC. There are other parameters to consider, such as geographical location and security, but overall VEC can be expected to enhance the performance of cloud systems.

Vehicular edge computing generally relies on three different types of communication:

- *Vehicle-to-Vehicle Communication.* V2V communication allows vehicular nodes to exchange data directly without the need of infrastructure. Vehicles with On-board Units (OBUs) are high-resource nodes, which are generally underused and they have the capability to communicate with other OBUs using Dedicated short-range communications (DSRC).

- *Vehicle-to-Infrastructure Communication.* V2I refers to vehicles and roadside infrastructure being able to communicate with each other. This communication is

mainly to collect data from applications or any information of interest for the network to perform optimally.

- *Vehicle-to-Anything Communication.* Both Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication play an important role in VEC. Therefore, V2X is a hybrid architecture that allows vehicles to communicate with any node in the network, such as mobile devices, pedestrians and roadside units

Each type of vehicular communication results in benefits and challenges for VECs. For instance, V2V communication helps to alleviate infrastructured networks and reduce the communication delay. However, the output retrieval from the computing offloading will be challenging as the high mobility can lead to losses of connectivity. Concerning V2V communication, the RSU improves connectivity performance using more stable access points with less mobility, but it might be congested due to the high volume of data. In regards of V2X, although it is the most versatile way of communication it has challenges for managing communication among heterogeneous entities. Even though communication is a major challenge for VANET and can cause performance drawbacks for the process of CO, considering it is still a recent topic, using vehicles as edge nodes is definitely a viable solution for a more efficient computation offloading.

2.2 Proposed classification

In the context of Vehicular Ad-hoc Networks, a number of surveys [27, 29, 36, 70] have studied data dissemination [47, 63, 97]. Nevertheless, these surveys study dissemination methods as part of an information diffusion process and not in terms of retrieval, as it is needed for a computation offloading system supported by vehicles and RSUs.

The classifications found in the literature are as follows:

- Akherfi et al. [7] defines the process of computation offloading with the following steps:
  - *Application partitioning*, where the application is divided into offloadable and non-offloadable tasks.
  - *Preparation*, to arrange everything that is needed for the partitioned tasks to be offloaded, including server selection.
  - *Offloading decision*, which is the final step of the process and it refers to sending the tasks to the remote server.

- Orsini et al. [69], considers a design guideline for the offloading system as follows:
  - *Partitioning or pre-phase to offloading*, to define and divide what parts of the application need to be synchronized after the offloading process finishes.
  - *Offloading*, referred as the most important step of the process, it is where the tasks are sent to a remote server.
  - *Handling of a global state*, where the system stays aware of the mobile nodes and considers the rapid change of the environment.

Although these categorizations consider the essential steps for computation offloading, if the system used is the one of Vehicular Edge Computing there are some steps that have a significant influence on the performance of the technique.

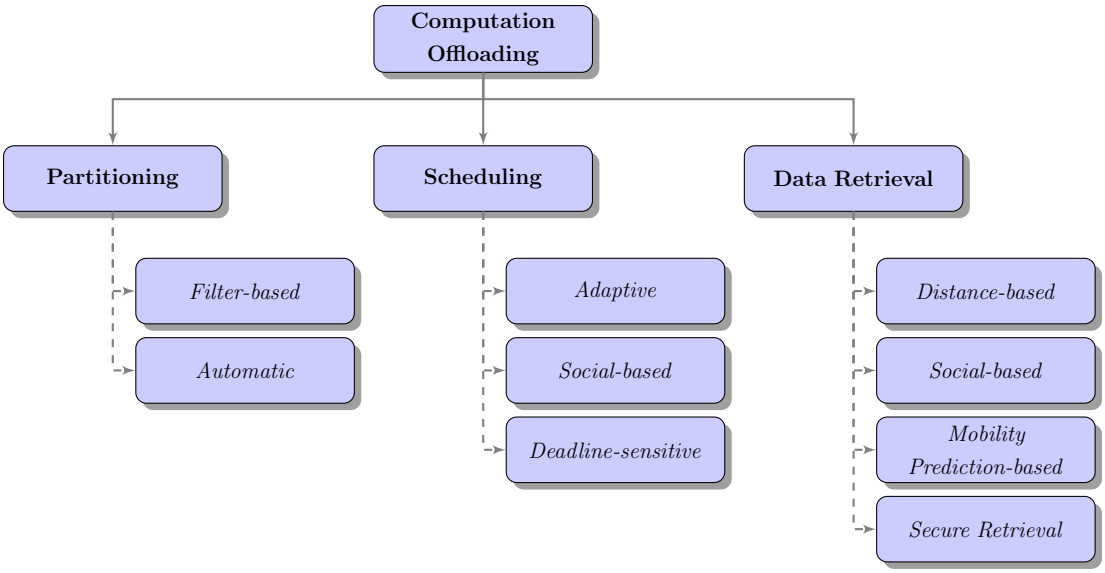


Figure 2.4: Taxonomy of the process of computation offloading.

In this work, we propose to classify computation offloading as follows: partitioning, scheduling, and data retrieval. The classification is based on the three major steps that are required for the process to be efficient. Furthermore, the steps are classified into subcategories based on the existing works and future directions for vehicular edge computing. The proposed taxonomy is represented in Figure 2.4. A brief description of the categories is presented below:

- *Partitioning*: Partition techniques are divided in two subcategories based on their objectives. *Filter-based*, where the algorithm aims to enhance performance regarding a single parameter like energy consumption or delay, and *automatic*, where the objective of the technique is to enhance overall performance in the designated environment, which in this case is edge computing.

- *Scheduling*: Scheduling mechanisms are separated into three subcategories. *Adaptive*, where the technique aims to adjust to the opportunistic nature of a mobile network to achieve an optimum performance, *social-based*, this method takes advantage of the mobility of the nodes and the frequent contacts of the users to reliably allocate resources according to their physical and social distances, and *delay-sensitive*, which refers to the algorithms that have the only purpose of enhancing the performance of the system, regarding deadlines and allocation delay.

- *Data Retrieval*: Data retrieval methods are divided into three subcategories. *Distance-based*, where the geographical location of the nodes is the main parameter considered for a successful retrieval of the output data, *social-based*, which with the same principle of the distance-based uses social relationships to obtain a more secure and efficient data retrieval, finally, *mobility prediction-based* techniques that take advantage of the mobility of nodes to predict the location of the nodes to calculate the best path for the user to retrieve the data packets.

2.3 Partitioning

The rapid development of technology in mobile devices has brought an increasing popularity of developing compute-intensive applications. Even when portable devices, such as mobile phones, tablets, laptops or even watches have new and more powerful computing resources every year, sometimes these are not enough for high-performance applications such as gaming, live-streaming or virtual reality applications. To support all of these applications, the works in [15, 32, 34] suggest task offloading to cloud servers. To do this, the application must be partitioned into smaller chunks of data, so there is a lower communication delay involved in the offloading process.

For applications to be divided into smaller tasks, first those tasks have to be classified into classes. This classes can be defined based on many criteria, for example, the importance of the task in the application. In the case of computation offloading in a mobile environment, such as vehicular edge computing, a major parameter to consider is

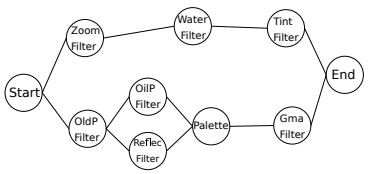


Figure 2.5: Topology structure of an eight-node application.

whether the task can be executed externally. Some applications may need information from sensors, such as GPS or the camera, that can only be obtained from the mobile device. Once these parameters are defined, and the application partitioned into smaller tasks, it is much simpler to offload them to be executed in an external server or device. Figure 2.5 represents the topology of an application used in [65] as part of a partitioning study.

Being the first step of the process, it is essential for the partitioning algorithm used to analyze the environment before starting to define parameters. In the case of vehicular edge computing, the most important characteristic to evaluate is the mobility of the nodes and the possibility of connectivity failure. Considering these two constraints, it is logic that a partitioning algorithm that breaks compute-intensive tasks into smaller tasks can help the nodes to reduce the execution delay and make the following steps more efficient. For scheduling, having an efficient partitioning algorithm means a more seamless resource allocation and task execution. Furthermore, for data retrieval, by reducing the overall delay implies that the mobile node will have fewer drawbacks considering that the distance traveled while processing the task is short.

Partitioning can be categorized in different ways [51], and while there are no standard parameters to evaluate vehicular edge computing, the works reviewed in the proposed subcategories are considered to be potential techniques to perform well if used for computation offloading with mobile nodes. The categorization of the existing work in partitioning for computation offloading is summarized in Table 2.1. Furthermore, the proposed solutions for each category are qualitatively compared in Table 2.2.

2.3.1 Filter-based Partitioning

Filter-based partitioning in the sense of computation offloading refers to the process of identifying the more convenient tasks to offload. Different parameters can be considered for filter-based partitioning, but in general, the reduction of resource consumption is the

Table 2.1: Partitioning approaches for computation offloading.

Approach	**Advantages**	**Disadvantages**
Filter-based	Identifies the more convenient tasks to offload, based on a specific preset parameter	-Does not consider any parameters outside of the preset ones -May not obtain the best performance
Automatic	Identifies tasks to offload based on the available resources	-May consume additional resources to verify availability

main objective. The process for a filter-based partitioning begins with defining what is the objective of the technique. For example, if the more complex tasks are going to be offloaded, then partition the application accordingly. The advantages of this approach are that the algorithms can be adapted for any specific constraint in the scheme it is used. However, because of the same reason the drawback can present when the environment considered has a few open issues and it becomes problematic for the partition algorithm to achieve an optimal performance. This method becomes practical for applications, such as language translation and voice recognition. There are several techniques that can serve for filter-based application partitioning [26, 44, 45]. However, most of them are designed for wired networks and may not perform as well in a wireless scheme [37], which is the one considered in this study. Because the amount of work related to edge computing is limited, the following works focus on different task profiling and are considered to potentially enhance the performance in vehicular networks with some adaptations.

Elicit [40] is a framework proposed to efficiently identify the compute-intensive tasks in mobile applications for offloading. This framework considers not only the profile of the mobile application but the environmental setup. The bandwidth between the server and the mobile device, network latency, and size over the overhead data are contemplated to make the optimal partition of the task based on the results obtained. The purpose of Elicit is to improve the execution time of the tasks. To do this, they partition the mobile application and offload the most resource-consuming ones, which are assumed to be known already. To select which method is to be offloaded, they first divide the processes into classifications. One of these is the Constrained Class where they have the methods that cannot be offloaded. After the separation process, the algorithm filters the methods by examining some parameters like execution cost in the mobile device, execution time on the server, and the class of the process. These filters are there to check none of the tasks are constrained to be offloaded. For the transmitting mechanism, they follow the same principles as the POMAC transparent mechanism proposed in [41],

which can offload methods that do not access any global or class variables. In addition to the method input parameters, Elicit can offload processes that access global or class variables. After the evaluation, the results show that Elicit can work with existing mobile applications and find the best method to offload the most compute-intensive part to the cloud or nearby servers via edge computing and obtain a reduced execution time and energy consumption.

An approach for reducing network traffic is proposed in [43]. The algorithm proposed models the applications as a graph with classes to then group these classes into clusters for a partition solution. The solution proposed in this work is categorized as a filter-based technique due to the fact that it has the main purpose of minimizing network traffic to enhance performance of pervasive computing schemes. The partitioning process starts with modeling the applications using Graph Theory, which models them as undirected graphs with a set of vertices, that represents the class or granularity of the applications, and a set of edges representing interaction between classes. After the application is modeled the clustering algorithm proposed aims to achieve a balanced load and memory usage for the CPU, therefore obtaining an efficient usage of resources and network traffic. In order to evaluate the performance of the proposed solution, experiments are carried out using a prototype. Although offloading to a remote server is not evaluated in these experiments, the partitioning phase shows to improve CPU and memory usage which should lead to the minimization of network traffic.

A similar work [6] proposes a system to optimize the performance of a network with long-duration streaming applications. In the presented scheme a partition algorithm is developed, which models the applications as a graph and it adapts to the current topology of the system to achieve the optimal distribution of tasks [83]. This partitioning mechanism aims to minimize the partitioning complexity and to balance the load in a changing topology.

Regarding filter-based techniques in terms of user preferences, the work in [65] presents a partition algorithm that aims to achieve an efficient personalized division of tasks for users. The objective of the proposed algorithm is to get the optimal performance in partitioning and offloading tasks regarding execution costs. The proposed approach makes use of a user profile model, which includes parameters like how often the mobile device is charged and memory cleaning frequency. These parameters are used as a training set to build the execution cost model. The cost model calculates the average delay and energy consumption for processing a task. In addition, all the information gathered by these models is then used by the max-cut partitioning algorithm, which finds the opti-

mal partition plan. In order to assess the performance of the partition algorithm, the authors compare it to some other techniques and it shows to outperform them. Although the algorithms used for comparison are not based on user preferences, the behavior of the user is an essential parameter to consider when the main objective of the proposed solutions is to achieve the best QoS.

According to the literature reviewed, many parameters can be considered for filter-based partitioning. It is always essential to understand the application before selecting the partitioning algorithm. The approach followed in the first work reviewed [40] is based on splitting the more resource-consuming tasks. However, considering that in vehicular edge computing some of the tasks can be executed in vehicles, it is important to recognize that the less complex the task is, the lower is the less execution delay. That is something to look for considering that vehicles are moving while processing the tasks. On the other hand, the technique proposed in [65] is based on a profile created for every user. In the context of vehicular edge computing, this approach seems more appropriate for the environment. The parameters considered for the profile are related to the behavior of the user. Therefore, each device will have different classes of tasks to offload and an adaptable QoS can be provided for the users. Even though the studies in [6, 43] do not perform experiments specifically on computation offloading, the partition solution aiming to enhance the performance in memory and CPU usage makes sense to lower the network traffic. These parameters are essential for a partitioning scheme to be considered in an overloaded environment, such as edge computing.

2.3.2 Automatic partitioning

In the context of computation offloading, automatic partitioning refers to identifying the tasks that can be offloaded to achieve an efficient task offloading based on the available resources at runtime. Unlike the filter-based technique, this method focuses on enhancing performance based on more general parameters, such as execution cost and optimal throughput. The advantage of automatic partitioning is that the algorithm considers the available resources, which is useful in a scenario with high mobility nodes. However, this technique aims to enhance the overall performance and cannot focus on single constraints. It is good for applications that need a rapid response, such as augmented reality or data streaming. There is little literature focused on this type of partitioning. However, representative work is discussed here.

The authors in [25], present a distributed architecture for mobile-to-mobile offloading

with an automatic splitting and scheduling algorithm. The scenario presented is with an offloader, referring to the mobile device that wants to push compute-intensive applications to another mobile device, and the neighboring offloadee nodes [24], which refers to the mobile devices that are in range at a time t. The splitting/scheduling algorithm designed and implemented in the experiments estimates the time for a task being run locally, and the required time to offload the task to an offloadee. Each device keeps a history of the tasks that have been executed locally, together with time attributes for each one in a vector. This vector is exchanged and updated every time a new offloadee device is found. Additionally, it is utilized in a linear equation where they consider data transmission delay, execution time, bandwidth and CPU speed to get the optimum splitting/scheduling for each task. After conducting analysis and experiments using real applications and data, their results show a 99.7% of efficiency for the splitting and offloading part and up to an 80% of energy saved when offloading a task to be executed.

Regarding task partitioning for multi-user scenarios, the work in [87] proposes ParGen, a computation partitioning scheme for data streaming applications, such as augmented reality. These applications use the camera on the mobile devices to collect images as an input and after analyzing them, an output is presented. The partitioning approach proposed is based on a building recognition application for students, where new students point their cameras at buildings and the name of the building and some relevant information is presented on their screens. The ParGen method, based on a genetic algorithm, separates users requesting the data into groups. Next, it adjusts the bandwidth of the server according to the groups. Finally, the best partition and allocation path is selected, based on average overhead. The proposed solution in this work is compared against other genetic algorithms and shows to outperform them. However, the simulation environment used for the experiments assumes that all the mobile devices are executing the same application, hence the same amount of data is being requested. In a MEC environment is important to consider that a significant part of the nodes have mobility, and it is not realistic to assume all of them will request the same data.

A bandwidth-adaptive partitioning scheme is presented in [64]. It aims to find an optimal partitioning solution for small applications and to reduce execution cost. After a model of the application is built using a weighted object relation graph, the partitioning algorithms, Branch-and-Bound based Application Partitioning (BBAP) and Min-Cut based Greedy Application Partitioning (MCGAP) algorithms, check the bandwidth changes and generates the partition models to minimize the execution costs. In order to evaluate the performance of the algorithms, simulations were performed using both of

them in different scenarios. While BBAP shows to perform well with small applications, the MCGAP algorithm shows to be fit for larger ones and both of them achieve to reduce the execution cost.

The work in [49] proposes an algorithm for application partitioning that focuses on multi-site distribution. Although it aims for energy efficiency in a mobile cloud computing scenario, it can be considered as an automatic partitioning scheme due to the fact that it considers different locations and adapts for execution in heterogeneous devices [22]. The process adopted for application partitioned in this work follows the next steps: application modeling, cost modeling, and partitioning solving. An energy model is used to achieve the optimum energy consumption. This model uses a data cost graph, which involves data from the CPU workload, the bandwidth rate and performance of the different locations considered for offloading [3, 19]. The authors propose a cyclic random movement genetic algorithm (CRMGA) to find the optimal partitioning solution for mobile applications. This algorithm first clusters all the unoffloadable tasks and assigns them to the local storage and then finds the optimal solution to partition the offloadable tasks. The experiments to evaluate this scheme are carried out on a prototype and the parameters used are based on energy consumption and the time spent obtaining the partitioning solutions.

Table 2.2: Representative algorithms for partitioning in computation offloading.

Proposal	**Category**	**Objective**	**Method**	**Potential Application**
Hassan et al., [40]	Filter-based	Reduce response time & energy consumption	Identify the most compute-intensive tasks	Mobile edge computing (Language translators)
Niu et al., [65]	Filter-based	Reduce computation costs	Exploit user preferences to define a more personalized partition plan	Mobile applications (multimedia, gaming)
Jungum et al., [43]	Filter-based	Reduce network traffic	Use graph theory and clustering to enhance the usage of memory and CPU	Pervasive computing

Ajwani et al., [6]	Filter-based	Minimize partitioning cost	Use graph theory together with an architecture-oblivious partitioning	Long-duration streaming applications
Wen et al., [87]	Automatic	Obtain optimal throughput	Enhance overall performance based on a genetic algorithm	Data streaming
Calice et al., [25]	Automatic	Reduce computation time & energy consumption	Split tasks based on the available resources	Augmented reality
Zhang et al., [49]	Automatic	Reduce energy consumption	Cluster offloadable and non-offloadable tasks to obtain the optimal partitioning solution	Speech recognition
Niu et al., [64]	Automatic	Minimize execution cost	Use a weighted object relation graph to model the applications before partitioning	Image detection

The proposed solutions for automatic partitioning reviewed here to consider whether it is convenient to execute the tasks on a remote server. The first work discussed [25] keeps a history of the tasks that have been offloaded and the ones executed locally. Furthermore, the approach in [87] creates clusters of nodes based on the requests received and adjusts them depending on the available bandwidth. Moreover, the work in [64] also adapts to the changes in bandwidth and can be used in a vehicular edge computing as it is designed for small applications, which translate into low execution time. Although both are different solutions for the same problem, the two seem to achieve the expected performance. In the context of vehicular edge computing, both of the techniques discussed show to be useful for nodes with mobility. Although the proposed algorithm in [49] is directed for mobile cloud computing, it considers mobile applications and the possibility of offloading them to different locations with heterogeneous devices. Based

on these characteristics this last work is categorized as automated partitioning and we consider it is a solution that can help further enhance the performance of edge computing if adapted adequately. Creating clusters according to the requests is favorable because different nodes are active at different times. On the other hand, keeping track of where the tasks were executed and their performance can significantly improve forthcoming application requests.

2.4 Scheduling

The task offloading technique is suitable for enabling mobile users to offload their compute-intensive tasks to a remote machine. However, because the number of mobile devices and users in this technology, the network is susceptible to a bandwidth congestion. Therefore, causing an impact on the quality of service. Therefore, edge computing aims to alleviate the overload by processing tasks at the edge of the network and only sending the necessary information to the core of the cloud. Using vehicles as edge nodes can result complicated at the time of scheduling. After partitioning the application into smaller tasks, the next step is to allocate them to the edge nodes for processing. However, the nodes considered in this scenario are vehicles, and it is essential to address the mobility issues. Many scheduling policies include a decision process that, if it is not automated, the user can decide if it is convenient to offload the tasks to a remote server or to execute locally.

There are some parameters to consider for this decision process, such as the delay of communication and processing in a remote server, and sometimes is not worth it. Typically a decision manager collects information about the resources available, resources needed and deadline constraints for the tasks to execute seamlessly. Then, depending on the scheduling policy the resources are allocated accordingly. In the context of vehicular edge computing, using a capable scheduling algorithm can significantly enhance the performance of computation offloading. Assuming the application is optimally partitioned, an efficient scheduling policy can guarantee a low-delay task execution. For both a high and low-mobility scenarios, an efficient execution means that the difference of position of the nodes is not substantial. Therefore, the retrieval of the output data can be seamless and efficient as well.

Using the same example from Section 2.3 and assuming that the tasks have been partitioned into different classes, Figure 2.6 shows the model of how the partitioned tasks are sent to a decision manager, which is responsible to allocate them to a vehicle

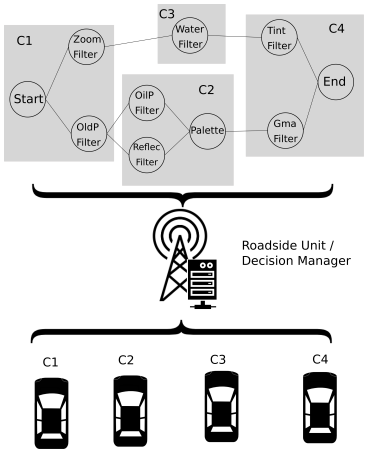


Figure 2.6: General model of scheduling using a decision manager.

by a scheduling algorithm used in vehicular edge computing. Here, we survey some of the existing works for scheduling algorithms in computation offloading, which aim to give the user the optimum Quality of Service (QoS). Moreover, some of the discussed works are not directed for vehicular edge computing but the main idea can be adapted to the system. The categorization proposed for scheduling in computation offloading is summarized in Table 2.3. Furthermore, the proposed solutions for each category are qualitatively compared in Table 2.4.

Table 2.3: Scheduling approaches for computation offloading.

Approach	**Advantages**	**Disadvantages**
Adaptive Scheduling	-Can adjust to changes, such as the frequent search of resources in a mobile scenario	-Complexity may generate execution delay -Difficult to guarantee an optimum performance
Social-based	-Exploits close contacts, therefore reliable nodes, for a balanced task offloading. -Considers priority for tasks.	-Security challenges -May be considered invasive by users.
Deadline-sensitive	-Aims to decrease the execution delay of the task -Can help to decrease communication cost in retrieval	-May not be the best option to consider for a high number of task requests

2.4.1 Adaptive Scheduling

The main motivation for adaptive scheduling is the constant search for resources caused by the dynamic nature of networks with mobile nodes. In a Mobile Edge Computing with support from vehicular infrastructure, both mobile devices and vehicles have mobility, which results in the frequent change of available resources for any region of interest considered. Using the proper scheduling policy has a significant influence on the performance of every application. Consuming time searching for available resources can translate to an increased computation delay, which means a poor QoS. The advantage of this method comes from schemes with mobile nodes, such as vehicular or mobile edge computing. However, because the main objective of adaptive scheduling is to adjust to opportunistic available resources needs to be available at all times, which can be complicated in such environment as vehicular networks. There are different approaches of using an adaptive scheduling algorithm depending on the scheme. In this section, we review some of the existing works that can be used in networks with mobility.

The work presented in [58] proposes a resource allocation and task scheduling policy for a vehicular cloud. The framework proposed consists of four major components: client manager, device manager, network manager, and decision engine. The client manager is in charge of the task information that arrives. The device manager controls the information about the available CPUs. The network manager collects information about the vehicular cloud and the infrastructure involved. The decision engine receives all the information collected by the managers and uses it to calculate the optimum decision about the assignment of the tasks. This approach aims to increase the number of available virtual machines and to reduce task completion time and energy consumption. It is assumed that vehicles with onboard units have access to GPS [1]. To achieve the multiple objectives, a heuristic hybrid Particle Swamp Optimization (PSO) [31] algorithm is used. The algorithm proposed adapts with the updated information obtained from the managers before making a decision on the fittest node to achieve the lowest task completion and energy consumption. Compared to standard and self-adaptive PSO, the proposed framework shows to enhance performance based on the parameters defined.

Related to node mobility and adaptive resource allocation, the work in [62] aims to use mobility prediction to avoid the negative impact of using an opportunistic network. The system used for this approach is the one of VCC, where groups of vehicles are within the communication range of the RSU to create a cloudlet. The mobility prediction model proposed is based on an artificial neural network, and the datasets used for training in-

clude the following parameters: resource lifetime, the position of vehicles, average speed of vehicles, and the number of vehicles ahead and in the same area. The performance of the prediction model was evaluated using four methods, R-squared, Pearson's correlation coefficient, mean absolute percentage error and root mean square error. Although the mobility prediction model showed to outperform other efficient approaches, there are parameters that were not considered during the evaluation. Moreover, significant parameters, such as communication cost, workload, and network overhead, need to be considered for resource allocation to achieve an adequate performance when taken to a more realistic scenario.

The authors in [30] present a scheme to optimize the offloading decision and resource allocation for tasks in a general multi-user system where the user has several independent tasks. To do this, they propose a heuristic algorithm based on Separate Semidefinite Relaxation (SDR) [71]. The system considered is the one of mobile cloud computing, where they have one cloud server, one access point, and N mobile users. Each one of them with a certain number of independent tasks. The algorithm proposed first solves the parameters for local execution and the settings for offloading and then compares them to see which option is more convenient. After that, it obtains the proposed offloading solution and the optimal resource allocation. The algorithm was simulated and compared to three different methods; local processing only, cloud processing only and lower bound of optimum. The lower bound is obtained by the SDR method. After getting the results, the analysis shows that the proposed algorithm gives nearly the optimum solution to get a reduced general offloading cost (regarding energy consumption, execution time, bandwidth). However, the optimum solution is the one obtained by the same semidefinite relaxation, and while it is a good solution, it is necessary to compare it to other similar algorithms or methods and evaluate its performance in a real scenario.

The work in [75] presents a solution for an overloaded mobile edge computing environment. It proposes two schemes for recovery when there is failure to find available resources and it fits the category of adaptive scheduling due to the fact that it can adjust to new parameters to avoid complete loss of service in a MEC cell. The first of the two schemes focuses on recovery when there is only one available neighbor within range and the second scheme is directed to the cases when there is absolutely no neighboring node available for wireless transfer. The recovery algorithm finds the adjacent MEC cells, clusters them with the disconnected nodes, and chooses a cluster head to manage the recovery system to ensure it works properly. After the offline nodes are connected to

new MEC cells, they can make requests through ad-hoc relay nodes, which will calculate the amount of data demanded and pass the information to the cluster head. To evaluate the capabilities of the proposed recovery mechanism in a MEC environment, simulations where performed using throughput and delay of transfer. Although the experiments were successful, it is worth mentioning that more realistic characteristics for the environment can be useful to evaluate the performance of the system in different scenarios.

From the literature review, it is evident that there are few existing works on scheduling for vehicular edge computing. However, due to the similarity in the architecture of the systems considered, vehicular and mobile cloud computing, it is not difficult to consider them for a vehicular edge computing scheme. One of the works reviewed [58], proposes having a manager for each of the major parts in the process, which in regarding such opportunistic network seems like an adequate decision. On the other hand, a different work [30] proposes to have a prediction mechanism to know the potentially available resources at all times. Choosing the adequate approach regularly depends on the application. For example, using a prediction mechanism seems suitable for an application that needs a quick response, such as video streaming or infotainment [73], while having a manager promises a more reliable and robust solution for applications such as virtual machine migration. Another work reviewed in this section [30] proposes to optimize scalability in a multi-user system, which is a mechanism that benefits the majority of applications for the environment considered in this survey. The last work studied in this section [75] is one that focuses on recovery of the network in situations of failure or lack of connectivity, which is a challenge that is expected in a mobile edge computing system. Moreover, the nodes used in vehicular edge computing often have high mobility, which can directly affect connectivity between nodes. Therefore, a solution as the one mentioned is essential for resource allocation and scheduling in computation offloading.

2.4.2 Social-based Scheduling

In the context of scheduling, the mobility of the nodes can be used as an advantage. Social-based techniques use data obtained from social media to develop an approach where the routine users follow day to day can be exploited to give them a better QoS on different applications. In the context of computation offloading in MEC with the support of vehicular networks, the nodes that have frequent contact can be used to balance the load of task requests. The advantage of these methods is that they use data that is already available to create clusters of nodes to use as reliable resources. However, the

drawback of it is that using that information from users can sometimes be considered somewhat invasive. Here we discuss existing works that use social connections to propose scheduling algorithms that aim to enhance performance for the computation offloading technique.

The authors in [89], propose a load balancing scheme based on the ball and bin theory [74], which is a phenomenon that can be applied to traffic balancing in distributed networks. They apply this theory in a scenario where a user needs to offload a task to a friend (mobile device contact). But considering the distribution of the relationship between contacts is not uniform, because it is expected that the user will spend more time with some contacts than with others, and the ball and bin theory needs a uniform distribution, then using this theory will not work. The authors came up with a new concept for balanced task offloading which they called Your friends are more powerful than you. This new concept considers the closer social relationships between users and keeps the record of the users that have a more frequent contact. Multi-hop schemes are not considered in the proposed method, and each task is regarded as a vital issue for each user. So, if multi-hop was to be used, task completion time will significantly increase. Furthermore, tasks with a high priority are to be executed locally. The proposed social-relationship-based algorithm was compared regarding efficiency [5], and it shows a good performance, as well as a significant difference in execution time between strangers and friends.

In the context of the geographical location of the nodes, a resource allocation scheme is presented in [98]. This approach aims to enhance the throughput of the network by using a social-aware algorithm to fairly allocate resources. This work is directed for device-to-device applications in a 5G network. However, because the architecture considered is similar to the one of MEC we review it in the sense of computation offloading for vehicular edge computing. The allocation algorithm proposed first groups the nodes in clusters, based on the content preferences of the users. The node that is physically closer to the other nodes is considered the cluster head, which is selected to receive the data and multicast it to the rest of the nodes in the cluster [83]. Furthermore, to add a node to the cluster the physical and social distance between it and the cluster head are examined. After the clusters are formed, the allocation algorithm follows two steps: power allocation and channel allocation. The experiments realized in order to analyze the performance of the proposed scheme showed that using a multicast technique has advantages in terms of maintaining a good balance between throughput and fairness. Although the technique reviewed showed to perform well, the authors consider the nodes

to have low mobility and to be used in an environment with vehicles, mobility is an essential parameter to consider.

Regarding user behavior in scheduling, the work in [86] proposes a task assignment solution that considers users mobility and resource distribution to achieve a reduction in task execution delay. This proposed solution examines execution time of tasks and mobility of the users between the base stations of a mobile edge computing system. Offloaded application should be executed in the amount of time that the user remains connected to a single base station or less. However, if that is not possible then the application is partitioned and transfered wirelessly to the next base station considering the movement trajectory of the user. The algorithm presented in this work checks the queue of tasks and assigns them to the base station that will perform the execution with the lowest delay possible. In order to evaluate the performance of the task assignment mechanism, simulation were executed to measure network parameters, such as delay, task acceptance rate, and number of users. When compared to schemes that do not consider mobility of the users, the mechanism proposed shows an enhancement in performance.

The literature reviewed here proposes using social information to optimize the performance of scheduling mechanisms in an edge computing environment with mobile nodes. The objective of both of the works reviewed here is to use frequent contacts to select them as secure nodes and reliably offload tasks. However, it is something to examine in more detail because in both approaches [89, 98] the available resources depending on whether the other users are open to making their resources available for other devices. Considering this, the solutions proposed can turn into an unfair system and lead to unbalanced workload problems. The study in [86] studies a key characteristic in edge computing networks, mobility. Specially for an edge computing scheme in vehicular networks, it is essential to develop solutions that consider mobility of the nodes to achieve an adequate quality of service. Even though this work is focused on mobile edge computing, the task assignment algorithm proposed can easily be adapted to a vehicular edge computing system, as the base stations used in both work in a similar way.

2.4.3 Deadline-sensitive Scheduling

Regarding computation offloading, delay is always a major constraint to consider. The main objective of schemes like Mobile Edge Computing is to provide a seamless experience to the user. Extra delay added to the process by not using the proper scheduling policy can also affect the communication cost when the output data is retrieved by the user.

The main focus of deadline-sensitive scheduling is to reduce delay as much as possible in order to meet the deadlines set for tasks. The advantages of this methods are that by reducing delay at the time of scheduling, the overall delay is also decreased, which helps to achieve a better performance. However, because the only focus of this methods is to meet deadlines, other issues, such as connectivity, may be unattended. This method is particularly useful for applications that are time-constrained, such as video streaming. In this section, we review existing works related to scheduling algorithms that aim to reduce the delay of the process.

The authors in [52] present a Mobile-Edge Computing (MEC) scheme. Where the system proposed uses a task buffer, and the tasks are scheduled based on several parameters, such as the queuing state of the mentioned buffer and the processing and transmission state of the device used. As the system model considered is based on the edge computing paradigm, to take advantage of this, the applications targeted are compute-intensive and delay-sensitive. For the task queuing, the model used first divides the time into time-slots with a same predefined length, where at the beginning of each slot a task is generated. Then, the created jobs are queued in the task buffer for the mobile device to decide if the task will be scheduled and executed locally or sent to a remote server. If the decision is to execute the task locally, then, the mobile device calculates the actual processing state, to know if there is a task already being executed; to define in what time-slot the arriving task will begin. In contrast, if the decision is to send it to the MEC server, where concurrent task execution is permitted. Then, the states of transmission, power consumption and delay are calculated. To evaluate the performance of the proposed scheme, the authors simulate three scheduling policies and compare them to their proposed one. The scheduling policies used are as follow, local execution policy (all the tasks are executed locally), cloud execution policy (all the tasks are sent to a remote server), and greedy execution policy where whenever the local CPU or the remote server is idle, any task waiting in the buffer is scheduled. After obtaining the results, the analysis shows that the proposed algorithm can achieve a lower average delay in task execution.

The work in [94] proposes an offloading scheme for cloud-based mobile edge computing in vehicular networks. The system presented, focuses on the offloading of delay-constrained applications for MEC in vehicular networks. There are some assumptions considered for the scenario that is worth mentioning. Firstly, a unidirectional road is studied, where RSUs equipped with MEC servers are deployed and individually identified. The vehicles are considered to have mobility, therefore they establish communication

with the RSU that has the strongest signal at the moment. Lastly, only homogeneous communication is established within the network. Considering there are many types of vehicles based on their resource capacity, a contract based scheduling policy is developed. The RSUs in the network broadcast information about the contracts, which the vehicles can obtain. Based on the information collected, the nodes process the utility for offloading a task. If the contract brings negative utilities, such as computation delay, then it is not convenient for the vehicle to send the task and it is processed locally instead. The offloading system aims to satisfy the task requirements and exploit the utility of the MEC scheme. Even though the analysis shows that the proposed approach maximizes the use of resources, an important parameter to consider in vehicular networks is scalability. It is essential to analyze the scenario with a high traffic density of nodes and study the effect this has on bandwidth consumption because of the contract information being broadcast.

The work in [93] proposes a resource allocation and task scheduling scheme that aims to optimize the performance of mobile cloud systems. However, the logic of the mechanism presented can be adapted to vehicular edge computing. The load-aware resource allocation and task scheduling (LARTS) algorithm is meant to give its best performance when the cloud is overloaded. In this situation, all the tasks are distributed again so the delay-tolerant tasks are given a lower priority than the deadline-sensitive ones. The algorithm uses directed acyclic graphs to model applications and takes advantage of the differences and independence of both delay-tolerant and delay-sensitive tasks to do the load-balancing and scheduling based on a genetic algorithm. In order to evaluate the performance of the algorithm proposed the authors realize experiments using parameters, such as arrival rate and cost of migration. The mechanism proposed shows to enhance the overall performance of the cloudlet-based system which proves that using it on an edge computing environment, which has a similar architecture, can potentially improve the quality of service.

A load balancing scheme is presented in [82], it aims to finish tasks as fast as possible in cloud computing systems. The authors use an online and offline method to to the task assignment in order to achieve the optimal scheduling solution. The online algorithm is randomized and follows a Poisson distribution. The offline mechanism is based on a bacterial foraging optimization algorithm, which provides high robustness and high speed. Therefore, it potentially reduces delay overall. In order to evaluate the performance of both algorithms simulations are performed using heterogeneous nodes [16]. The two algorithms show to be efficient, the online algorithm dealing with the task assignment and the offline one providing stability on the system.

According to the literature reviewed, delay has a significant influence on the performance of a system such as vehicular edge computing. Both of the works reviewed in this section are proposed for edge computing and focus on delay-constrained applications. The first work [52] proposes to create a queue with time-slots, and then the user can decide whether is convenient to offload. The authors in [94] propose a similar solution for a vehicular environment with the same purpose of reducing delay. In their case the use a contract-based scheduling, where the utilities for offloading a task are calculated before making the decision to offload or to execute locally. Both solutions make possible for the user to achieve the lowest delay possible. Deadline-sensitive algorithms, such as the one discussed in [93] for scheduling are important for applications like data streaming that need a low delay to meet the deadline for the tasks. The idea of proposing an offline algorithm [82] is resourceful because connectivity issues in an edge computing system are common, specially in vehicular networks where high mobility is involved.

Table 2.4: Representative algorithms for scheduling in computation offloading.

Proposal	**Category**	**Objective**	**Method**	**Potential Application**
Midya et al., [58]	Adaptive	Scalability and Flexibility	Use particle swarm optimization to allocate a huge number of task requests	Vehicular Cloud Computing
Mustafa et al., [62]	Adaptive	Reduce impact of mobility in performance	Use an artificial neural network-based mobility prediction model for resource allocation	Virtual machine migration in VCC
Chen et al., [30]	Adaptive	Maximize overall performance (energy consumption and delay)	Use semidefinite relaxation and random mapping to allocate resources	Mobile Cloud
Satria et al., [75]	Adaptive	Recovery from failure	Cluster nodes with neighboring MEC cells	Real-time applications

Yang et al., [89]	Social-based	Reduce execution time	Load balancing by offloading tasks to friends or nodes with frequent contact	Data sharing
Zhao et al., [98]	Social-based	Balance throughput performance and fairness between multi-cast nodes	Create clusters based on physical and social distances to multi-cast data	Data streaming
Wang et al., [86]	Social-based	Lower execution delay	Achieve an optimal task assignment considering mobility of the users	Mobile applications
Liu et al., [52]	Delay-sensitive	Decrease power consumption and computation delay	Analyzes the average delay and energy consumption of tasks	Mobile applications
Zhang et al., [94]	Delay-sensitive	Successful execution of delay-constrained tasks	Uses a contract based policy to maximize the resources available	Mobile Edge Computing
Zhanget al., [93]	Delay-sensitive	Meet deadlines in an overloaded system	Model and schedule applications based on the independence of their tasks	Image/video processing
Tang et al., [82]	Delay-sensitive	Reduce execution time	Use an online and an offline algorithm for task assignment to provide robustness	Cloud computing

2.5 Retrieval

Computation offloading, previous to mobile devices, did not have to deal with connectivity issues or communication delay due to the mobility of the nodes. However, the rapid development of communication capabilities of vehicles and mobile devices makes data dissemination an essential topic of research for the development of applications in vehicular networks. Considering the research area is relatively new, the number of proposed solutions to the challenges in retrieval for computation offloading is limited. The typical scenario and communication links for data retrieval in a Vehicular Edge Computing scheme, where a pedestrian offloads a partitioned task to an RSU and it is assigned to a vehicle, is represented in Figure 2.7.

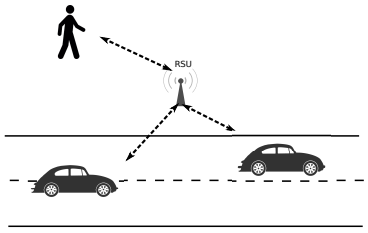


Figure 2.7: Typical scenario for data retrieval in MEC with vehicular support.

Regarding computation offloading in a mobile edge computing scheme with support from vehicular networks, both Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication play an important role.

The retrieval process in computation offloading starts when the offloaded tasks are processed and the output data needs to be collected by the user. Depending on the approach taken for the offloading, the node responsible for the output data can be an RSU or a vehicle. The main challenges for data retrieval presented in an environment where nodes have high-mobility are intermittent connectivity and broadcast storm. There is literature that proposes dissemination techniques for vehicular applications. Nevertheless, these papers focus on dissemination as a way of diffusion and not for retrieval as it is needed in computation offloading. In the remainder of this section data retrieval techniques for computation offloading are surveyed.

Although a considerable amount of research is directed to dissemination techniques for applications in vehicular networks, such as content delivery, emergency notification, traffic control, and intelligent transportation systems, the area of retrieval in computation offloading requires more study. The majority of the literature is focused on data

Approach	Advantages	Disadvantages
Distance-based	Uses location based on GPS from OBU and RSU to accurately forward data packets	Beacons to obtain location add delay. Obtained location needs to be precise. May generate message redundancy leading to broadcast storm.
Social-based	Uses contact between nodes to categorize them into clusters and use nodes with frequent contact as reliable forwarders	Relies on neighboring nodes to forward data. May be considered invasive. Not good for low traffic densities.
Mobility prediction-based	Using traffic prediction can lead to a better use of resources, therefore a more efficient retrieval of data.	Accuracy of prediction algorithms may not always be accurate, leading to data loss.
Secure Retrieval	Protect users privacy and security by encrypting information	Failure of system may lead to information leaks

Table 2.5: Categorization for data retrieval techniques in computation offloading.

dissemination as a mean to spread messages. However, computation offloading requires data forwarding in the sense that a node retrieves the output data of a task that was offloaded to an external server. There are many existing dissemination techniques that can be adapted to CO and, as mentioned before, most of them are classified into push, pull, hybrid. Therefore, here we discuss existing techniques that can potentially be adapted to data retrieval for computation offloading with vehicular infrastructure as support. The categorization proposed for data retrieval in computation offloading for vehicular edge is summarized in Table 2.5. Furthermore, the proposed solutions for each category are qualitatively compared in Table 2.6.

2.5.1 Distance-based Data Retrieval

Considering the main challenge in the scenario of computation offloading for MEC in vehicular networks is mobility, it makes sense to use location as the key parameter of the retrieval. Assuming GPS is available for the vehicles and the RSUs in the network, which is feasible, using location can help the retrieval method to forward data packets to the accurate location. Hence, achieving an efficient retrieval with low communication delay. The advantages of this method are that by using distance as a parameter, the probability of a broadcast storm due to message redundancy can be reduced by sending

the data packets to a particular location. However, the drawbacks are that broadcasting the data can lead to security issues [23]. Although there is a lack of research literature in this area, a representative work is reviewed here. It is essential to mention that some of the surveyed works are not directly related to data retrieval for computation offloading but, based on the technique used, can be adapted to such a system.

The REPRO protocol presented in [79] uses a distance-based method for data retrieval in MEC with the support of vehicular networks. This protocol aims to address connectivity, mobility, and delay issues by using the idle cars as support infrastructure whenever the vehicle that is processing the task is out of reach of any RSU. Once an RSU is reached, using the advantage that the network topology is known, the best path for retrieval is calculated, allowing the sender RSU to retrieve the processed data with the least delay possible. Assuming the vehicles and RSUs have access to GPS and the nodes are aware of the existing topology, REPRO uses a beaconless, timer-based data forwarding technique. Once the data is ready, it is broadcast. The vehicles that receive the broadcast use the location information to set a timer, which varies in value and is greater the further away it is from the destination. This means that the node that is closer to the destination has the lowest timer. When the timer expires the data packet is broadcast by the forwarding vehicle. If the broadcast is received by a vehicle that had a greater timer, this vehicle cancels the forwarding process to avoid redundancy. This process is followed until the data reaches its destination or until the data finds an RSU, which then is responsible to forward the packet to the destination.

The advantage of using vehicles as support infrastructure in the context of data retrieval for computation offloading is that they have a significant amount of resources that are commonly unused. There are two scenarios that can be considered for vehicles: moving and parked vehicles. The advantages of a moving vehicle are that energy consumption is not a priority because as long as the car is running the battery is charging.However, in the case of a parked vehicle, it is important not to use all the energy. There are some research works that focus on dissemination of content by adapting the transmission power to reduce energy consumption and avoid a broadcast storm.

The work in [28] presents Adaptive Forwarding message and Cooperative Safe driving (AFCS), an efficient pull-based technique for emergency notification applications. Most of the dissemination techniques used in these type of applications rely on broadcasting the messages, which leads to a high bandwidth consumption due to the redundancy of the packets. It uses a power adjusting technique based on the distance of the nodes to save energy and at the same time reduce the possibility of a broadcast storm. Although

this technique is directed to emergency notification messages, the same principle can be used for computation offloading.

A data dissemination protocol for traffic safety applications is proposed in [67]. The scheme presented uses a multi-hop broadcast protocol, which based on the distance between nodes it adapts and uses a candidate selection algorithm to choose the appropriate node to relay the messages. The presented scheme was compared to other beaconless and beacon-based protocols and it showed an acceptable performance.

Finally, three different forwarding techniques are presented in [39], this analysis is focused on enhancing the performance regarding transmission delay and quality of service. The framework presented in this work is directed towards networks with massive coexisting devices, like any Internet of Things application. The results of the analysis show that the quality of service provided is based on the delay tolerance of the tasks.

It is evident that there is a lack of research work in the area of data retrieval for computation offloading. However, the work reviewed in this section follows a logical solution, which is using the distance between the processing node and the host that needs to retrieve the output data. The advantages of this approach are two. First, message redundancy is avoided by using a timer that cancels any forwarding repetition. Second, using distance as a parameter for directing the broadcasts helps to reduce the bandwidth consumption by sending the data packets to a specific area. Finally, by using several nodes to forward the output data it is easier to recover from any issues that may present with connectivity. Moreover, it aims to achieve the lowest retrieval delay possible by using both vehicles and infrastructure, such as RSUs.

2.5.2 Social-based Data Retrieval

In general, social-based approaches exploit information obtained from the user, such as location, frequently visited places, frequent contacts, and interests. Considering social media and the number of sensors mobile devices have these days [66], this data is not complicated to obtain and it can surely assist many applications to achieve an enhancement of performance and QoS. The advantages of using social-based methods is that information about the relation of the nodes is already available and can be easily obtained from social media with permission of the users. However, sharing such information can be considered invasive by the users and refusing to share it can negatively affect the performance of the system. Because there is a lack of research work focused on data retrieval for computation offloading in general, it is important to mention that not all of

the works reviewed here are directly related to computation offloading, but are to data dissemination with social-based techniques that can be adapted.

The work in [48] proposes a social based mechanism for data forwarding with mobile ad hoc networks in mobile cloud computing. This mechanism is based on Distributed Hash Table (DHT), which is a technique developed for supporting a high number of nodes dynamically interacting with the network. The proposed approach aims to enhance the efficiency of data forwarding by considering users as nodes and using social relationships of these nodes to create clusters. A parameter, social distance, is introduced to represent the relationship between two users after an ID is assigned to each node. The objective of the forwarding mechanism is to find forwarders until the social distance is equal to zero, which means the data has reached its destination. The case that there are no neighbors around can be presented, in which the node sends the data packet to the closest base station and it is responsible to find a suitable forwarder. However, if there still no nodes nearby to forward the data, the process is aborted.

In the context of computation offloading for MEC in vehicular networks, low-complexity data retrieval techniques are of significant importance. Considering that the more complex an algorithm is, it will take more time to process, complexity translates to delay and communication costs in the process of CO.

There can be many approaches for dissemination techniques with low-complexity algorithms, for example, an algorithm which first establishes a multicast tree to transmit data packets with a low computation complexity, to achieve low communication costs is presented in [53]. This dissemination algorithm does not use RSUs and it takes advantage of opportunistic networks to share data. Experiments were realized using a prototype, and the analysis shows an enhanced performance as the number of nodes increases.

Although the approach proposed studies multicast in mobile opportunistic networks, the prototype used for the experiments exploits Bluetooth and WiFi connections to disseminate the packets, which are available in vehicles with on-board units. Moreover, nodes with high-mobility are not studied in the literature reviewed, which is an essential parameter to consider if a dissemination technique, such as this, is considered to be adapted to a vehicular environment.

A Data Dissemination protocol Based on Centrality is presented in [33], which aims to reduce the number of relay nodes without diminishing the coverage area or increasing delay. Every car has a list of the one-hop and two-hop neighbors at all times to select the best relay node. The experiments showed that the protocol did not achieve the expected results. Even though this technique shows to achieve an acceptable performance in the

reviewed literature, it is important to consider that most of the applications that use it are to diffuse content. This means that the objective of this technique is to reach the most nodes possible. A multi-device data retrieval technique can be useful for computation offloading, particularly for tasks that need more time to be processed. However, it is essential to mention that redundancy is not acceptable for data retrieval in CO and that the proper adjustments should be considered to achieve an efficient performance.

Multi-device dissemination refers to using more than one device to forward messages from one place to another. With multi-hop routing, a data packet can travel long distances using different nodes as relays. This technique is commonly used in ITS applications, such as traffic safety and emergency messages. However, this method can help computation offloading by expanding the coverage range of RSUs, using vehicles as relay nodes to retrieve the output processed data and achieving a low delay.

Although the reviewed work was considered as a social-based data retrieval technique, the approach focuses on the selection of the nodes to forward the data and not on the retrieval rate. However, because there is a lack of research works in the area of data retrieval for computation offloading, it is essential to study all the available literature. Similar to the approaches in scheduling and partitioning, the authors in the discussed work [48] use social connections between users to select a reliable forwarding node. In this case, the parameter of social distance is introduced, which is an advantage considering there are no standard parameters for vehicular edge computing. However, the drawback of depending on the availability of other users and their eagerness to provide personal information exists in this approach too.

2.5.3 Mobility prediction-based Data Retrieval

In data retrieval schemes that use vehicles as infrastructure, mobility prediction can be considered essential for the successful delivery of information. Either high or low mobility, knowing where the destination will be beforehand can save a significant amount of communication delay. Moreover, mobility prediction not only concerns the specific location of a node, but also the different levels of traffic flow existing during the day in an urban scenario. The advantages of mobility prediction are similar to the ones of distance-based techniques. However, this approach proposes to calculate the potential location of the node or the user after a determined time, which can guarantee a more accurate retrieval. On the other hand, the drawbacks of this method are that depending on the mobility algorithm used, the complexity and the accuracy of it can lead to a

decrease in the retrieval rate.

An approach for Offloading of cellular networks through ITS is presented in [54]. The scheme presented relies on a Directed Short-Range Communications (DSRC) vehicular network to alleviate the cellular network from content delivery. It is assumed that the vehicles are able to communicate with both vehicular and cellular networks. Therefore, the users with mobile devices can take advantage of the ITS network to obtain content by downloading it directly from the RSUs or using vehicles as relay nodes to forward the data packets. It is considered that a traffic manager, part of the ITS environment, can predict the mobility of the nodes. Using this prediction, the traffic manager defines a link between two nodes that are within the maximum radio range. Two models are considered for the evaluation of the scheme proposed, RSU-driven relaying and greedy relying. The objective of the RSU-driven approach is to use the nodes contact prediction values to schedule the content delivery, whereas the greedy relying follows three steps: the downloaders broadcast a list of the content presently downloading, the relay nodes filter the requests for content that they do not have, and finally the relay decides how to deliver the available content requested.

A cloud-assisted safety message dissemination framework for VANETs is proposed in [50], which aims to reduce packet loss and message redundancy. It uses buses, which are equipped with mobile devices and VANET interfaces, as a gateway for the message dissemination. While the framework presented shows a satisfactory performance, it was only compared to flooding techniques and the simulations were performed on a single road, which needs more realistic scenarios. Even though an approach like this seems to have only advantages, there are still some open challenges to consider if a data retrieval technique for computation offloading using public transport is to be developed. Firstly, there are not many nodes in a public transport network, which, depending on the application, may not be reliable enough. Lastly, security issues might present [23], considering the number of public transport users it might be a vulnerability.

The work [54] reviewed in this section aims to offload cellular traffic through vehicular networks. Although it is not an approach for edge computing directly, the concept and architecture used are similar. Considering the lack of work in the area, it can be considered for data retrieval in a vehicular edge computing. As discussed before, there are no standard parameters to evaluate the performance of such a technique. However, the scheme presented uses a mobility prediction algorithm for the delivery of location-based content, which can significantly optimize the performance of data retrieval protocols by calculating the possible location of the user at the time of retrieval. The drawback, in

this case, is relying on the mobility prediction algorithms to obtain an accurate and successful retrieval rate. Moreover, mobility prediction algorithms can sometimes be too complex and add significant delay to the process.

A thing to consider when using vehicles as support infrastructure in computation offloading is that vehicles are mobile nodes in an opportunistic network. It may be complicated to know what kind of computational resources are available on each of them, or at least it is something that consumes time, which is essential in offloading. The idea of using public transport as the support nodes is practical because of two reasons. The first reason is that in many cities they have schedules, which makes it more simple to predict their availability. The second reason is that, because they are very similar vehicles and many times deployed by the same company, permission issues like the ones that may present in a social-based approach are avoided and it is easier to know their resource capacity. Although the work presented is not used in a computation offloading scheme, based on the architecture used it can be adapted for it.

2.5.4 Secure Data Retrieval

Security in vehicular networks is an issue of significant importance in vehicular networks. The idea of intelligent transportation systems, which is to have automated vehicles on the streets, makes way for the vulnerability of hackers accessing the controls of vehicles in the network and possibly cause accidents [56]. Besides this, there is also the theft of information by intercepting messages, such as emergency messages or shared data in computation offloading, being broadcast throughout the network. Because of this reasons, security has become a popular topic in message dissemination for vehicular networks.

For example, the work in [2] proposes a secure message dissemination broadcasting with encryption and policy enforcement, computation cost of decryption is delegated to the nearest RSU. Policy enforcement means that messages should be enforced with an access policy and should be selectively delivered to the vehicles according to said policy. The proposed scheme provides fast decryption and reduces the overhead of storage in the vehicle.

Another scheme proposed in [59], uses time stamp defined Message authentication codes (TD-MAC) for data dissemination in VANET. The analysis presented studies security in data dissemination concerning the resiliency and communication overhead of the algorithm proposed. The results show that it outperforms other efficient MAC

schemes.

The scheme proposed in [60] detects vehicles that revokes potential fake vehicle nodes to ensure the safety of data dissemination in VANETs. Although the proposed scheme shows adequate performance against the models that it is compared to, it needs more realistic scenarios for experiments.

As discussed before, security is an important parameter to consider for data dissemination in vehicular networks. The works presented above propose different methods of making data sharing more secure. However, security measures still need to be adapted for different purposes, for example aiming for a lower network overhead in the case of computation offloading. Moreover, it is essential to have experimented with realistic scenarios with the objective to build a more robust system.

Table 2.6: Existing algorithms for data retrieval.

Proposal	**Category**	**Objective**	**Method**	**Potential Application**
Soto et al., [79]	Distance-based	Alleviate MEC network bandwidth using vehicles as support infrastructure	Uses a timer based on the distance between node and destination to avoid message redundancy and achieve a low communication delay.	Computation offloading for mobile applications

Chang et al., [28]	Distance-based	Energy consumption,	Use power adjusting technique to reduce energy consumption	Emergency Notifications
Oliveira et al., [67]	Distance-based	Reliability on dissemination	Use distance between nodes to select the optimal relay node	Safety message broadcasting
Farooq et al., [39]	Distance-based	Interference, packet delay	Use delay-tolerance of applications to ensure every message reaches its destination on time	massive IoT networks
Li et al., [48]	Social-based	Improve service availability/data forwarding performance	Introduces social distance as a parameter, the forwarding algorithm finds nodes with the lowest distance to use as a forwarder.	Computation offloading for mobile applications
Liu et al., [53]	Social-based	Communication cost	Uses a low-complexity algorithm to reduce overall cost	P2P, mobile social networks
Costa et al., [33]	Social-based	Overhead	Reduce the number of relay nodes without diminishing range using knowledge of neighoring nodes	ITS
Liu et al., [50]	Mobility prediction	Dissemination delay	Uses public transport as a gateway for message dissemination	Safety message distribution
Malandrino et al., [54]	Mobility prediction	Alleviate cellular network	Uses mobility predictions from a traffic manager to select forwarding nodes.	ITS content download

Liu et al., [2]	Secure Retrieval	Fast decryption and low storage overhead	Proposes a security policy which delegates encryption/decryption to the nearest RSU	Emergency messages
Mondal et al., [59]	Secure Retrieval	Scalability in the network	Uses time stamp defined message authentication to achieve a more secure dissemination	Message dissemination
Mondal et al., [60]	Secure Retrieval	Reduce network and communication overhead	Revokes nodes based on their behavior	Message dissemination

2.6 Future Directions

This section presents a general discussion and open challenges found in the areas of partitioning, scheduling and data retrieval for computation offloading. Moreover, there is an evident lack of research works in the area of data retrieval for computation offloading. Because of that, dissemination techniques that can potentially be used in computation offloading for vehicular edge computing are presented here.

2.6.1 Partitioning

Regarding a computation offloading scheme after reviewing the literature, selecting the partitioning parameters for the tasks to enhance performance has proven to be useful. However, some of the newer applications developed have tasks that require access to sensors from the mobile device or to the camera, which makes them difficult to offload. An interesting approach is reviewed in this survey, where a user profile model is used to deliver the optimal performance for every user. Many applications these days are developed with the purpose of giving the users a more automated lifestyle, therefore considering their behavior for decision-making algorithms may lead to achieving an optimal QoS and not only an overall adequate performance. The entire process of computation

offloading can be a more efficient one if from the start the adequate approach is used. If the partitioning algorithm can guarantee a good performance, the scheduling algorithm can achieve a better performance as well, and the same happens with the retrieval algorithm. However, it is important to perform experiments in the proper environment and make the necessary adjustments to the system.

2.6.2 Scheduling

Adaptive scheduling techniques, such as the ones reviewed, can prove to be effective for a more general approach. However, as the solutions proposed tend to be heuristic techniques, the scheduling performance is not always optimal. There are several parameters to consider in scheduling, like execution delay, load balancing, and energy consumption. Furthermore, if the focus of the algorithm is to adapt to the constant change in topology and available resources, it makes it more complex to have multiple objectives and aim at the same time for reduction of computational cost for example. Moreover, security remains an open challenge in the reviewed work.

The reviewed literature on a social-based load balancing scheme that considers nodes with frequent contact to be reliable regarding data sharing. This approach can become useful considering these days almost every application on mobile devices is related to social media in some way. However, they do not present experiments or analysis using real data which is necessary to see if this algorithm is functional in practical scenarios. Regarding computation offloading with the support of vehicular networks, a social-based approach can be exploited if we consider that most of us have a daily routine. The same vehicles are regularly parked at the same places, hence using resources from idle vehicles can significantly enhance the performance of the system and alleviate bandwidth consumption from the core servers.

Although the proposed ideas seem to obtain good results, there are some parameters to consider to prove that it is an efficient and feasible scheme to be used in a practical scenario. For example, the number of tasks generated should be more prominent and for a system like this one, using edge computing, scalability should be an essential parameter to consider. Using a scheduling algorithm to reduce the computation delay can also contribute to the performance of the data retrieval. In the context of computation offloading in vehicular networks, if a task is allocated to a vehicle, the best approach is to assign it to the most capable node. Not only to reduce the execution time of the task but to ensure that the vehicle has not moved too far by the time it finishes

the computation. If the mobility parameter of the vehicles is considered at the time of scheduling, the data retrieval process can be made more efficient.

2.6.3 Data Retrieval

In the context of data retrieval for computation offloading, the distance-based forwarding mechanism is a promising technique. Using distance and speed parameters with high-mobility nodes is the most logical approach. However, depending on the application and the environment considered, vehicles may have low or no mobility at all and if that is the case, the parameters for the forwarding method should be adjusted. Furthermore, the scenario considered may be urban or rural, which changes the settings to considerate for the development of a new technique. Usually, an urban environment can be assumed to have more infrastructure to be used, such as RSUs, traffic control managers, and base stations. But, for a rural scenario or a highway for example, in the majority of the cases, there is only roads and a few vehicles that can be helpful to the system.

Moreover, a major parameter considered for the development of data forwarding in vehicular networks is traffic flow. Although the same scenario can be considered, the traffic flow of it may change with the time of day or even the seasons. The solution reviewed in this section offers a retrieval technique, which uses a distance-based timer. The use of this timer at the time of forwarding the output data from the offloaded task addresses two of the main challenges mentioned before, the broadcast storm and the intermittent connectivity. Moreover, using a timer that considers the distance between the host and the destination, ensures that the maximum coverage is exploited. At the same time, by using both vehicles and RSUs as forwarders it enhances the probability of a successful retrieval with a low traffic flow scenario.

Some of the proposed technique considers mobility on the nodes and how it impacts QoS on data forwarding for mobile cloud computing. Using social contacts as a resource to improve the performance of data retrieval, is a smart way of exploiting all the available resources to achieve a secure delivery. However, an approach of this category, that relies on other nodes to forward data, may result impractical for an environment with a low traffic flow. Moreover, the social-based approaches reviewed here assumes that those nodes that have frequent contact can rely on each other, therefore, can be efficiently used as relay nodes. However, as with most of the social-based approaches, this can be considered slightly invasive in the sense that there is a record kept of frequent contacts and locations of users.

2.7 General Discussion

Computation offloading is essential for resource-constrained devices to improve their processing and storage capabilities, as well as to provide a more reliable service. Vehicular edge computing uses vehicles at the edge of the network to offer this. Furthermore, by processing tasks at the edge of the network it alleviates the load of the core layer cloud servers and reduces communication delay.

This paper presents existing work on the processes of partitioning, scheduling and data retrieval for computation offloading in a VEC architecture. However, there is a significant lack of literature in the area of data retrieval. Because of this, existing dissemination techniques used in vehicular applications, such as emergency messages and traffic control, are considered for potential data retrieval methods in a VEC environment.

Considering the lack of research there is in data retrieval for computation offloading in VANET, due to being a recent topic, based on the knowledge obtained from the reviewed work, a possible path for future work is the development of an efficient and reliable data retrieval protocol for computation offloading in a VEC environment. However, it is reasonable to expect vehicular edge computing to be a solution to provide a more reliable service for mobile and vehicular applications in the future.

Chapter 3

Preliminaries

The VEC paradigm helps resource-constrained devices enhance performance with the computation offloading scheme, where the user can send computationally-intensive applications to a remote server to be processed. With this, the user saves energy while also being capable of executing said applications. The computation offloading process follows a series of steps. First, the application is partitioned, or broken into smaller, "offloadable" tasks. Then, the device finds and establishes a connection with a server to finally send the tasks that need to be offloaded.

In this section a model of the computation offloading system considered is presented with the purpose of understanding the system's behavior, as well as to asses the ability to simulate the elements that contribute to enhancing its performance in a vehicular edge computing environment. For the environment contemplated in this work, vehicles and RSUs are considered as support infrastructure in VEC; vehicles as infrastructure support help the cloud network distribute the load, while efficiently processing the offloaded tasks using commonly underused resources from vehicles. It is essential to mention that the model used for experimentation includes the 802.11p protocol together with dedicated short-range communications (DSRC), which allows periodic beaconing, switching between control and service channels, and Wave Short Message (WSM) handling. This environment considers a set of assumptions. First, a set of applications (A) exist, where $A = \{A_1, A_2, ..., A_i\}$ is composed of i applications. Each application (A_m) is effectively partitioned into smaller tasks (c), where $A_m = \{c_1, c_2, ..., c_j\}$ is composed of j tasks.

Second, a set of RSUs (M), where $M = \{M_1, M_2, ..., M_k\}$ is composed of k RSUs. Every RSU use a queue to allocates the tasks that were offloaded using a FIFO scheduling policy. A queue, Q, is used for the scheduling policy and each task in the queue can

Table 3.1: Variables of the model

Notation	Description
A	Applications
M	RSUs
N	Vehicles
P	Pedestrians with a mobile device
c	Components or tasks of an application
V	Vehicles available for new tasks
R	Total retrievals
Q	Queue of tasks
t, x, s, d, a, g	Time, position, speed, distance, availability and range
j_r^n	Results for job n
$d_{v,p}$	Distance from vehicle to pedestrian's last known location
L	Total communication delay
q	Communication delay for a single node
T	Timer
t_p	Processing time
x'	Estimated position
$Z_{b,d,j,r}$	Messages (beacon, data, job, results)

belong to a different application. Moreover, the queue is limited by q which represents the available resources of each RSU. In addition, it is assumed that a wireless network of RSUs is distributed across an urban area.

Finally, the network elements are aware of geographical locations. The location information of the RSUs, which is kept in an address table by each of the static nodes, is obtained using beacon messages. Mobile devices and vehicles are assumed to have a global positioning system. Since the RSUs are static nodes, their positions, together with the network topology, are also known.

The model used, for which the notations are listed in Table 3.1, considers a set of vehicles (N) as defined in Equation 3.1,

$$N = \{N_1, N_2, ..., N_l\} \tag{3.1}$$

where N is composed of l vehicles and each one has information of their current status defined by x, t, s, and a as the current position, time, speed, and availability parameters, respectively. Since the area considered for the experiments is a section of an urban area, the number of vehicles is variable and all of them have different speeds. However, it is relevant to mention the vehicles are not connected to the network at all times. The availability parameter is to show if the vehicle is already processing a task or if it has processing resources available to receive a new one. These vehicles have on-board units, which, like the RSUs, send 1-second interval beacons with information about them such as their position and their availability to receive a new task, based on whether they are currently processing a task.

A set of pedestrians (P), where $P = \{P_1, P_2, ..., P_m\}$ is composed of m pedestrians that own a mobile device and have the need to offload a task to a server. These mobile devices receive the beacons sent by RSUs and vehicles, thus are considered to be aware of the nodes nearby.

For the RSU to assign an application component to a vehicle, it must first consider the availability information of the vehicle which was received with the beacon message. The set of tasks to be assigned to a vehicle can be described as in Equation 3.2.

$$S = \sum_{1}^{q} (c_j(loc = 0)) \tag{3.2}$$

Where S represents the total tasks ready to be assigned, $\{c_j, ..., c_q\}$ constitutes the tasks in the queue that are not being executed locally, represented by $loc = 0$. Furthermore, V represents the set of vehicles that are available to receive a task, as defined in Equation 3.3.

$$V = \cup N_l(a \neq 0) \tag{3.3}$$

Once the set of ready tasks and the set of available vehicles are defined, the RSU follows a selection process, where the best available vehicle is given priority based on Equation 3.4. This equation is used to estimate if, based on simple calculations, the vehicle will still be in the coverage area of the connectivity range from the RSU.

$$v_i \in V = x_{v_i} < M_i\{g\} \tag{3.4}$$

Where v_i is the selected vehicle, x_{v_i} is the estimated position of the vehicle and g is the connectivity range parameter of the RSU (M_i). A sample of calculations for the estimated position are defined in Equations 3.5, 3.6, and 3.7.

$$x'_x = \{x + [(s \cdot t_p) \cdot cos(md)]\} \tag{3.5}$$

$$x'_y = \{x + [(s \cdot t_p) \cdot sin(md)]\} \tag{3.6}$$

$$x' = (x'_x, x'_y) \tag{3.7}$$

Where x is the current position of the vehicle, t_p is the processing time of the task and md is the movement direction angle. Different parameters can affect the retrieval of the processed data. However, successful retrieval can be defined as a set of total processed data retrieved (R), where $R = \{R_1, R_2, ..., R_n\}$ is composed of n outputs from every task that was assigned to a vehicle.

The output processed data is defined in Equation 3.8, where if the position of the pedestrian ($P\{x\}$) at the time the task is processed and ready for retrieval is inside of the coverage area of the RSU ($M_j\{g\}$), the retrieval is a successful one.

$$R \Rightarrow P\{x\} \leq M_j\{g\} \tag{3.8}$$

The processing time of a task j in the Edge Cloud is defined in Equation 3.9. It involves the scheduling time, $s(j, V')$, the migration of the task to the selected vehicle, $off_d(j, V')$, the execution time, $e(t, V', l_{V'})$, and finally the retrieval, $r_d(dat_j, V', protocol)$. In this case each time variable depend on the size of the tasks being offloaded and it is obtained after the offloading process.

$$t_{p_{EC}} = s(j, V') + off_d(j, V') + e(t, V', l_{V'}) + r_d(dat_j, V', protocol) \tag{3.9}$$

Evidently, the gain of offloading a task to a remote server can be represented in terms of time and energy consumption, but since this work is focused on the retrieval process, we will focus on the gain in terms of time as represented in Equation 3.10. The equation describes the difference of running the task locally, t_{p_m}, against running it in the Edge Cloud, $t_{p_{EC}}$. Both time variables are assumed to be obtained after the offloading process, as both of them depend on the size of the offloaded tasks.

$$g(t) = t_{p_m} - t_{p_{EC}} \tag{3.10}$$

The computation offloading scheme can be represented with Equation 3.11, where every considered element in the network is integrated: pedestrians (P_m), vehicles (N_l)

and RSUs (M_k). The set of applications (A_i) belong to the mobile devices, which are owned by pedestrians . These applications are partitioned into smaller tasks and offloaded to a server in the edge of the network, an RSU in this case. Once the tasks need to be processed, the RSU looks for available resources in vehicles ($v_i \in V = x_{v_i} < M_i\{g\}$) and allocates tasks to them. As defined before, S represents the tasks that are not executed locally and the retrieved output data is represented by R.

$$O = \sum_{1}^{i,m} A_i \in P_m + \sum_{1}^{k,l} M_k, N_l + \sum_{1}^{i} (v_i \in V = x_{v_i} < M_i\{g\}) \cup S, R \tag{3.11}$$

Chapter 4

Data Retrieval in Vehicular Environments

From this point on, the computation offloading scheme O is separated into two parts. In the first (RSU-Triggered Task Assignment), the "sender" RSU, which is the RSU that will allocate the received task to a vehicle, receives a task to process, and is assigned to a vehicle using three different protocols for retrieving processed data. In the second (Mobile Device-Triggered Task Assignment), the "sender" pedestrian can decide to offload the task directly to a nearby vehicle or send it to a broker (RSU).

4.1 RSU-Triggered Task Assignment

Usually, RSUs are strategically located near nodes, such as vehicles, traffic lights, or traffic controllers, or are meant for applications, such as emergency message distribution or traffic coordination [38]. Therefore, with the massive and increasing number of mobile devices that might take advantage of the computation offloading paradigm, it is easy to assume that at some point an RSU can get overloaded. In that case, the network will need assistance to process these offloaded tasks; as a consequence, high-resource elements such as vehicles are considered the support infrastructure.

The primary challenge in this situation is related to the mobility and connectivity of the nodes. This retrieval strategy starts from the assumption that when the vehicle to which a task was assigned finishes the computation, it will want to return the processed data to the sender RSU. Depending on some parameters, such as the speed and the processing time of the task, the vehicle's position might be different from the position it

occupied when the task was received. To address this challenge, three different protocols are proposed below. First the Topology protocol, which uses V2I communication only and takes advantage of the network's topology to find the shortest path to retrieve the output processed data. Second, Distance-based forwarding, that relies on V2V communication alone for the retrieval. Finally, the REPRO protocol, which uses a hybrid of V2I and V2V communication to achieve the most efficient retrieval of the output processed data. For the following protocols, it is relevant to mention that whenever a vehicle receives a beacon message from an RSU, the vehicle answers with a data message that carries parameters, such as speed, movement direction, and availability.

4.1.1 Topology-based Retrieval Protocol

With the topology of the network and the known location of the RSUs, the Topology-based Retrieval Protocol follows a logical retrieval solution. The vehicles following this protocol receive and process a task; at the moment of sending the processed data back, they broadcast it in a single attempt to reach an RSU in the network. If the message does not reach any RSU, the processed data is lost. If it does, the RSU that receives the processed data first checks the destination information in the message. After that, if it matches with its ID, it finishes the process and retrieves the processed data in a single-hop manner. If the destination does not match, the RSU forwards the message following the network topology to the shortest path possible, until the sender RSU retrieves it with the fewest hops possible.

For the algorithms in this work, messages sent and received are represented by Z_y, where y is substituted by b, j, and r, for beacons, jobs and results respectively. In addition, *nmap* is used to represent the mapping of the network, once the beacon messages are received and answered. The Topology-based Retrieval Protocol, which we refer to as Topology Protocol from now on, is presented in Algorithm 1. Lines 1-4 show the RSUs sending beacons to perform the network mapping, the status of the job queue is shown as well. Lines 5-7 show how the RSU assigns a task to an available vehicle based on the answer received from the beacon. In lines 8-14, the process of retrieving results in a single-hop or many-hop manner is delineated, where in the latter, the RSU forwards the message following the best path in the topology. The same Topology Protocol, but from the perspective of the vehicles, is presented in Algorithm 2. This algorithm shows the vehicle answering the beacon message from the RSU in lines 1-2. In addition, lines 4-5 show that if the vehicle is available to process a task, it will receive it, process it and

Algorithm 1: Topology Algorithm: RSU

```
1  M^n ∪ (Q ∋ {c_j})                          # Queue status
2  Send: Z_b_msg(N)                           # Send beacons
3  Receive: Z_d_msg(N ∈ V)                    #Receive node information
4  nmap = M ∪ N{x, t, q, a}                   # Network mapping
5  if v_i{a} ≠ ∅ then
6      Send: Z_j_msg(V)                       # If vehicle is available
7      V ∩ j^i                                assign task
8  Receive: Z_r_msg(V)
9  if M_d{ID} = M^n{ID} then
10     {c^n_M ⊂ A^n})                         # Receive output data
11                                            and verify RSU's ID
12 else
13     nmap ∋ M_d                             #If not mine, forward to next RSU
14     Forward: Z_r_msg(M_d)
```

send the output data back to the destination RSU, represented by M_d.

4.1.2 Distance-based Forwarding Protocol

The Distance-based forwarding (DBF) protocol is based on geographic routing using mostly V2V communication to retrieve the processed data. As with the previous protocol, the RSU network topology is known due to the beacon messages sent between each other, and the vehicles have a geographical positioning system on-board. Once a vehicle becomes available to receive a task and finishes executing it, the processed data must be retrieved by the sender RSU. As the name suggests, this protocol uses the distances between nodes for that. To avoid confusion, the car that receives the task is referred to as a "processing vehicle" and as for any other cars in the process, the term "forwarding vehicle" is used. The processing vehicle contacts only the sender RSU. First, the processing vehicle checks if the sender RSU is within range distance. If it is, the processed data is retrieved in a single-hop manner. If it is not, it broadcasts the message to any nearby vehicle. Second, the forwarding vehicle receives the data and calculates the distance from its position and to the location of the sender RSU, to compare if it is closer to the destination than the vehicle that previously forwarded the data. If it is not, then

Algorithm 2: Topology Algorithm: Vehicle

```
if Receive:Z_b_msg(M_k ∈ M) then
    Send:Z_d_msg{x, t, q, a}(M_k)        #If beacon is received,
                                          answer with info message
if Receive:Z_c_msg(M_k) then
    Send:Z_M_msg{c_M^j}                  #If task is received,
                                          process and answer with output
```

the message is discarded and the data is not forwarded. The distance ratio is calculated using Equation 4.1.

$$d_R = \frac{d_{v_f,r_d}}{d_{v_p,r_d}} \tag{4.1}$$

where d_R is the distance ratio, d_{pv} is equal to the distance from the previous vehicle that forwarded the message to the destination RSU, and d_{fv} is the distance from the currently forwarding vehicle to the RSU.

Based on this ratio, the forwarding vehicle sets a timer to send the processed data one hop forward. This process is followed by every vehicle that received the data. The value of the timer is calculated with Equation 4.2 where T_{max} is a constant and T is the timer value.

$$T = T_{max} \cdot d_R \tag{4.2}$$

It is relevant to mention that with shorter distances, the timer is also shortened. Hence, once the forwarding vehicle broadcasts the processed data, every vehicle that receives it and has a timer running cancels both the timer and the forwarding process. To achieve that, only the closest car to the sender RSU forwards the processed data to avoid message repetition, leading to a broadcasting storm. This operation is followed until the sender RSU has retrieved the processed data. Once the destination RSU has received the output processed data, a message is broadcast to stop the forwarding process of this data. It is assumed that the last vehicle to forward the output will receive it as it is still in the communication range.

To better understand this process, an example can be given following Figure 4.1. Let us assume that the user in the figure has an application that can be partitioned into jth

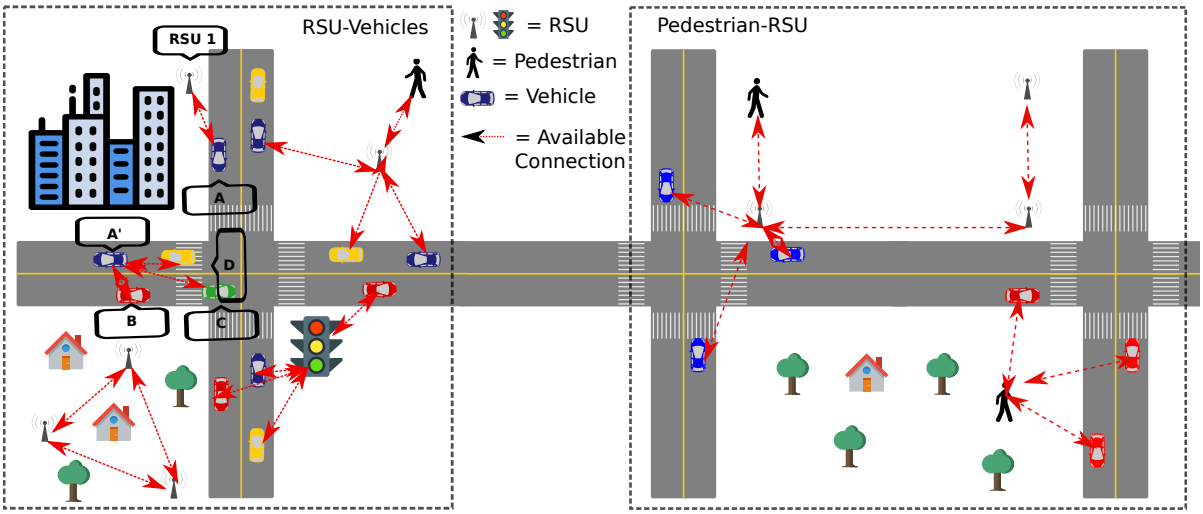


Figure 4.1: Data retrieval scenarios.

pieces. The components of the application can be offloaded to the RSU 1, when the user moves and enter in the RSU 1's coverage. Considering RSU 1 assigned a task to vehicle A, its location is now where vehicle A' is by the time it finishes processing it. At this point the sender RSU is unreachable, and the processed data is to be received by vehicles B, C and D.

Upon the receipt of the message, the three of them sets a timer based on their distance, as explained above. Because vehicle D is the closest one to the sender RSU, it is the first one to forward the processed data; hence, cars B and C cancel the forwarding process when the new message is received.

Algorithm 2 shows the process followed by the vehicle that processes the task while using the DBF. The process that forwarding vehicles use for retrieval is shown in Algorithm 3. Lines 1-4 show the cancellation of the retrieval when the timer is running, and a repeated message is forwarded by a vehicle that is closer to the sender RSU. Lines 6-10 show that when a set of processed data is received and no timer was set, it is the first time the processed data is received. The values of both the distance ratio and the timer are calculated to set it. Finally, lines 10-11 show the forwarding request after the timer expires, in case the forwarding process was not canceled.

In Algorithm 4, where the process followed by an RSU is shown, lines 1-4 show the exchange of information between RSU and vehicles to map the network. Lines 5-7 show that if a vehicle is available, a task is assigned to it. Finally, in lines 8-10 show the RSU receiving the processed data and verifying the destination's ID. Unlike the topology

Algorithm 3: DBF Algorithm: Forwarding Vehicle

```
1  if Receive: Z_r_msg(N) then
2      info = (N){x, t}                          #Output message is received
3      if (Forward:Z_r_msg(N) = isScheduled) then
4          Cancel:Z_r_msg(N)                     #If the same message is already being
5                                                forwarded, cancel the process
6  else
7      info = v_n(d_{v_f}, d_{r_d})
8      d_R = d_{v_f,r_d} / d_{v_n,r_d}           #Calculate distance ratio
9      T = T_max · d_R                           and start timer
10     Schedule:Z_r_msg(v^n) T_ON()
11 if T_OFF then
12     Forward:Z_r_msg(v_f)                      #Forward output message when timer expires
```

protocol, the only one that collects the processed data with this protocol is the sender RSU.

4.1.3 REPRO

From the previously discussed protocols, we can deduce that the Topology Protocol is the best choice regarding retrieval delay when the coverage of RSUs does not pose an issue. On the other hand, DBF copes with connectivity and mobility issues. REPRO (Result rEtrieval PROtocol) protocol is motivated by these deductions. It is developed as a hybrid strategy, where the processed data is retrieved using an adaptation of the DBF protocol up until an RSU is reached. Then, it follows the Topology Protocol until the sender RSU retrieves the processed data.

The following outlines more details of an offloading scheme in which the REPRO protocol is followed. The RSUs are aware of the network topology, the availability of the nodes, and parameters such as speed and moving direction. The sender RSU assigns a task to an available vehicle; when it is processed, the car starts following the DBF retrieval protocol. This means that the processed data is broadcast and the vehicles that receive it (forwarding vehicles) set a timer based on the distance ratio, as previously explained in Section 4.1.2 (using Equations 4.1 and 4.2). The forwarding vehicle broadcasts the data again once the timer is done, if the process was not interrupted by a repeated

Algorithm 4: DBF Algorithm: Sender RSU

```
1  M^n ∪ (Q ∋ {c_j})                          # Queue status
2  Send: Z_b_msg(N)                           # Send beacons
3  Receive: Z_d_msg(N ∈ V)                    #Receive node information
4  nmap = M ∪ N{x, t, q, a}                   # Network mapping
5  if v_i{a} ≠ ∅ then
6      Send: Z_j_msg(V)                       # If vehicle is available
7      V ∩ j^i                                assign task
8  Receive: Z_r_msg(V)
9  if M_d{ID} = M^n{ID} then
10     {c_M^n ⊂ A^n})                         # Receive output data
11                                            and verify RSU's ID
```

message. This method is followed until any RSU from the network is reached. Once an RSU receives the processed data, the Topology Protocol is followed. As explained in Section 4.1.1, when an RSU gets a processed data message, it calculates the best path to reach the sender RSU.

This protocol aims to address connectivity, mobility, and delay issues by using idle cars as support infrastructure whenever the vehicle that is processing the task is out of reach of any RSU. Once an RSU is reached, the best path for retrieval is calculated using the advantage that the network topology is known, allowing the sender RSU to retrieve the processed data with the least possible delay.

Algorithm 5: REPRO Algorithm: RSU

```
M^n ∪ (Q ∋ {c_j})                                  # Queue status
Send: Z_b_msg(N)                                   # Send beacons
Receive: Z_d_msg(N ∈ V)                            #Receive node information
nmap = M ∪ N{x, t, q, a}                           # Network mapping
if v_i{a} ≠ ∅ then
    Send: Z_j_msg(V)                               # If vehicle is available
    V ∩ j^i                                        assign task
Receive: Z_r_msg(V)
if M_d{ID} = M^n{ID} then
    {c^n_M ⊂ A^n})                                 # Receive output data
                                                   and verify RSU's ID
else
    nmap ∋ M_d                                     #If not mine, forward to next RSU
    Forward: Z_r_msg(M_d)
```

Algorithm 6: REPRO Algorithm: Vehicle

```
1  if Receive:Z_b_msg(M_l ∈ M) then
2      Send:Z_d_msg{x, t, q, a}(M_l)                    #If a beacon is received,
3                                                        answer with my info
4  if Receive:Z_j_msg(M_l) then
5                                                        #If a task is received
6      if nmap = (M_l{g} = N{g}) then
7                                                        and the destination RSU is still in range
8          Send:Z_r_msg{M_l})                            #Answer with output message
9      else
10         Send:Z_r_msg(N_l)                             #Broadcast it to a vehicle
11 if Receive: Z_r_msg(N) then
12     info = (N){x, t}                                  #Output message is received
13     if (Forward:Z_r_msg(N) = isScheduled) then
14         Cancel:Z_r_msg(N)                             #If the same message is already being
15                                                       forwarded, cancel the process
16 else
17     info = v_n(d_vf, d_rd)
18     d_R = d_vf,rd / d_vn,rd                           #Calculate distance ratio
19     T = T_max · d_R                                   and start timer
20     Schedule:Z_r_msg(v^n) T_ON()
21 if T_OFF then
22     Forward:Z_r_msg(v_f)                              #Forward output message when timer expires
```

Algorithm 5 shows the side of the RSU retrieving processed data while using REPRO protocol. The mapping of the network is shown in lines 1-4. Lines 5-7 show the task assignment to a vehicle. Finally, lines 8-13 show the final retrieval part, where the RSU decides if the results need to be forwarded or if they have reached the sender RSU.

Algorithm 6 shows the process followed by a vehicle using the REPRO protocol. The identification process is shown in lines 1-2, where the vehicle receives a beacon and answers back with a data message. Lines 4-10 show the decision process followed when the processed data is ready to be broadcast. In lines 11-20, the vehicle receives a processed data message and goes decides whether to forward it or cancel the forwarding

process that has been initiated. Finally, lines 21-22 show the message being forwarded if the timer was not interrupted.

4.2 Mobile Device-Triggered Task Assignment

Having understood the retrieval protocols from the point of view of the RSU in the previous section, we deduce that the REPRO is the best choice for retrieval concerning mobility, connectivity and communication delay. Because the studied environment is the one of Edge Computing with support of vehicular infrastructure, it is necessary to analyze the offloading scheme from the side of the mobile devices. In this work, it is assumed that mobile devices are carried by pedestrians. As before, vehicles are used as support infrastructure in the VEC environment. It is also assumed that all the elements, RSUs, vehicles, and pedestrians are aware of locations via geographical positioning system. Pedestrians are considered to own a mobile device, and the task from an optimally partitioned application needs to be offloaded.

With the purpose of obtaining a better performance from the mobile devices, pedestrians have two options. The first is comprised of offloading the task to an RSU, which acts as a broker or manager for the entire process. Considering that RSUs have the vehicles as support, in this case the RSU then assigns the task to a vehicle for later retrieval, after it is processed. The second option in this scheme consists of skipping the RSU and offloading the tasks directly to a vehicle, saving at least two hops of communication delay.

Three protocols are discussed in this section. Geographical Forwarding uses only vehicles to both process and retrieve the processed data, and takes advantage of the location awareness of the nodes in the network. The Vehicle-assisted Computation Offloading (VCO) protocol uses an RSU as the broker and aims to achieve a smother offloading process. Finally, the Movement Prediction Protocol (MPR) protocol also uses RSUs as a broker, with the purpose of achieving a higher retrieval rate of processed data and lower delay. It also uses a prediction algorithm to acquire a more precise estimated location of the sender pedestrian at the time the vehicle finishes processing the respective assigned task.

4.2.1 Geographic Forwarding Protocol

The Geographic Forwarding (GF) protocol relies on V2V communication. It is a beaconless protocol, and it uses the last known position of the pedestrian as a reference, which is the location at the time the task was offloaded and it is transmitted jointly with the offloaded task. The process followed by the GF begins after the task is computed, and the processed data needs to be retrieved by the sender pedestrian. Therefore, the vehicle broadcasts the processed data at this point in time. Similar to the DBF protocol from Section 4.1.2, the vehicles that receive the broadcast data use the last location known to the sender pedestrian to calculate a distance ratio using Equation 4.3.

$$R = \frac{d_{v,p}}{d'_{v,p}} \tag{4.3}$$

where R is the distance ratio, the distance between the last known location of the sender pedestrian and the processing vehicle is represented by $d_{v,p}$, and $d'_{v,p}$ represents the same variable, but from the vehicle that is currently forwarding the processing data. Using this ratio value, following Equation 4.2 ($T = T_{max} \cdot d_R$), a timer is calculated and set. If the same message is received during that time, the forwarding process is canceled to avoid message repetition, leading to a broadcasting storm. This process is followed by every vehicle following the GF protocol. It is relevant to remember that the vehicle that is closer to the sender pedestrian is the only one that forwards the processed data.

The communication delay considered for this scenario is based on the model from Equation 4.4.

$$D = \sum_{i=1}^{n} d^i_{v^i} + d^i_{p^i} \tag{4.4}$$

where D is the total delay, and d is the delay for every single node involved in the process.

The forwarding process followed by the vehicles using the GF protocol for processed data retrieval is shown in Algorithm 7. In lines 1-3, the vehicle obtains the location information from the message received and calculates the distance ratio. In lines 5-7, the timer, together with the forwarding of the processed data is canceled if a repeated message is received. If the message received is new, the calculated timer is set (lines 9-10). Lines 11-12 show the forwarding event being scheduled after the timer is done.

Algorithm 7: GF Algorithm: Vehicle

```
1  Receive:Z_msg                                  #Receive message including
2  R = d_{v,p}/d'_{v,p}                           location information
3  T = T_max · R                                  #Calculate distance ratio (pedestrian-vehicle)
4                                                 and timer value
5  if Data:Z_r_msg(N_l) then
6      if Forward:Z_f_msg(N_l) = isScheduled then
7          Cancel:Z_f_msg(N_l)                    #If it is a repeated result message,
8                                                 cancel forwarding
9      else
10         Schedule:Z_r_msg(N_l)@T                #Else, forward it
11 if Data:Z_j_msg(P_m) then
12     Schedule:Z_r_msg(N_l, P_m)@T               #If it includes a task,
13                                                process it and answer with the output
```

4.2.2 VCO

The VCO protocol, unlike the GF, uses both vehicles and RSU to retrieve the processed data, but it relies more on V2I communication to achieve a more efficient retrieval. It is based on a more realistic adaptation of the REPRO protocol, explained in Section 4.1.3, where the sender pedestrian uses an RSU as a broker to manage the entire offloading scheme.

In more detail, the RSU is aware of the available vehicles at all times. The moment a pedestrian offloads a task via its mobile device, the task is assigned to the closest RSU. In this protocol the sender RSU, or broker, is the only node aware of the last known location of the pedestrian. This is the single node that needs to retrieve the processed data. To avoid a broadcasting storm, the same process explained in Section 4.2.1 is followed, which involves a timer and distance relation calculated with the same equations. Once the vehicle finishes processing the task, and it is retrieved by any RSU, it decapsulates the information to check if the last known location of the pedestrian is within range. If it is, the message is sent to the pedestrian. If not, the processed data is forwarded until it reaches the RSU closest to that location. For this scenario, the communication delay follows the model in Equation 4.5.

Algorithm 8: VCO Algorithm: RSU

```
1  nmap = M ∪ N{x, t, q, a}                          #Network mapping
2  Receive:Z_msg                                     #Receive message including
3                                                    location information
4  if Data:Z_t_msg(P_m){l = 0} then
5      Send:Z_t_msg(N_l)                             #If the task received is not executed locally,
6                                                    assign to a vehicle
7  if Data:Z_r_msg(N_l) then
8                                                    #If a result message is received,
9      if (P_m{g}) = M{g} then
10         Send:Z_r_msg(P_m)                         #and the destination is in range, send it
11     else
12         Forward:Z_r_msg(M)                        #Else, forward it via RSU
```

$$D = \sum_{i=1}^{n} d^i_{v^i} + d^i_{r^i} + d^i_{p^i} \tag{4.5}$$

where the hop delay for every hop is represented by $d^i_{v^i}$ for vehicles, $d^i_{r^i}$ for RSUs and $d^i_{p^i}$ for pedestrians until the processed data reaches its destination.

The vehicles under the VCO protocol follow a similar process to the one shown in Algorithm 7. RSUs behave according to Algorithm 8, where line 1 shows the mapping of the network. Line 2 shows the RSU receiving a message, which includes location information from the sender. Lines 4-5 show the task is assigned to a vehicle when it is received. Line 7 shows a processed data message is received, where a decision process begins. If the last known location of the pedestrian is within range, then the processed data is retrieved in a single-hop manner (lines 9-10). If not, then it is sent to the closest RSU to said location (lines 11-12).

4.2.3 Mobility Prediction Retrieval Protocol

The Mobility Prediction Retrieval (MPR) protocol is an extension of the VCO protocol. It aims to enhance the performance and reliability regarding the success of retrieved processed data and delays. The primary problem in retrieving processed data in a computation offloading environment, using vehicles as infrastructure to support the scheme,

is the mobility and the connectivity of the nodes. These problems are addressed using a prediction algorithm, which uses the information collected from the sender at the moment the task is offloaded. With a simple technique such as Dead Reckoning, which uses parameters such as speed and direction of movement, it is possible to calculate the estimated position of the sender after a specific time (processing time), and with this information obtain a more accurate routing at the moment of retrieval.

The process of prediction is based on Equations 4.6 and 4.7.

$$X'_x = X_x + ([s \cdot t_p] \cdot [\cos(md)]) \tag{4.6}$$

$$X'_y = X_y + ([s \cdot t_p] \cdot [\sin(md)]) \tag{4.7}$$

where (X'_x, X'_y) are the coordinates of the estimated position, (X_x, X_y) are the last known coordinates of the sender pedestrian and s, t_p, md are the speed, processing time, and movement direction, respectively. The calculation is done at the moment the sender RSU retrieves the processed data; the moment before it forwards it to the sender pedestrian, it compares the estimated position against its own. If the distance between them exceeds the communication range, the processed data is forwarded to the next closest RSU to try and reach the sender pedestrian at the new position. If not, the usual process is followed, and the sender pedestrian retrieves the processed data from the sender RSU.

Algorithm 9 shows the process followed by a sender RSU using MPR. In line 1, the network topology is obtained, including static nodes and one-hop mobile nodes. A message is received in line 2, which includes location information from the sender. Lines 4-5 show that if the message received contains a task, then it is assigned to a vehicle from the neighbourhood. If the message received contains processed data (line 7), the estimated position of the sender pedestrian is calculated (lines 9-10). Lines 11-12 show that if the estimated position is within the range, the processed data is forwarded to be retrieved by the sender pedestrian. If not (lines 13-14), the message is forwarded to the RSU that is closest to the calculated location, to then be retrieved by the sender pedestrian.

Algorithm 9: Prediction Protocol

```
nmap = M ∪ N{x, t, q, a}                                  #Network mapping
Receive:Z_msg                                             #Receive message including
                                                          location information
if Data:Z_t_msg(P_m){l = 0} then
    Send:Z_t_msg(N_l)                                     #If the task received is not executed locally,
                                                          assign to a vehicle
if Data:Z_r_msg(N_l) then
                                                          #If a result message is received,
    X'_x = X_x + ([s · t_p] · [cos(md)])                  #Calculate new potential position
    X'_y = X_y + ([s · t_p] · [sin(md)])                  of pedestrian
    if (P_m{g}) = M{g} then
        Send:Z_r_msg(P_m)                                 #if destination is in range, send it
    else
        Forward:Z_r_msg(M)                                #Else, forward it via RSU
```

Chapter 5

Performance Evaluation

To evaluate the performance of data retrieval protocols in an edge computing environment assisted by vehicles as infrastructure it is essential to conduct experiments measuring diverse parameters. These experiments were realized using Veins [78], an open source framework for vehicular network simulations, which includes two-ray path loss models [77], SUMO [46] for road traffic simulation, and OMNeT++ [84] for network simulation.

Figure 5.1: Example of map used in experiments.

Although three different scenarios were utilized for the different types of protocols, all of them were realized in a similar map. The selected map is located in a section of the downtown area of Ottawa, Ontario, Canada, as shown in Figure 5.1. The map used in the experiments includes a section of the highway, which allows us to further evaluate the protocols at very high speeds. The red dots shown on the map are an example of how the RSUs were distributed with an approximate separation of 300m between each

Table 5.1: Simulation Parameters

Parameter	**Value**
Frequency	5.89 GHz
Sensibility	-89 dBm
Tx Power	20 mW
Beacon Interval	1 s
Vehicle traffic density	$6, 12$ $v/km/hr$
Pedestrian traffic density	10 $p/km/hr$
Vehicle speed (avg/max)	$50 - 120$ km/hr
Pedestrian speed (avg/max)	$5 - 10$ km/hr
Map size	~ 9 km^2
Task processing times	$5 - 145$ s
T_{max}	100 ms
Simulation time	1000 s
Number of simulation runs	35

other, enough distance for them to be connected wirelessly for 1-hop communication. The RSUs are static nodes, but for both the vehicles and pedestrians, two different classes of random routes were generated. Every route has different parameter values for vehicles and pedestrians: speed, lanes, paths, and so on. Fifteen types of tasks with varied processing times were used. The processing times range from 5 to 145 seconds and are assigned randomly, following a uniform distribution. Also, for the protocols that use a timer, the value of T_{max} is $100ms$.

The analysis examines distinct aspects, such as the number of processed data retrieved, the communication delay, and the number of hops it takes to retrieve the output data. It also evaluates how all of these are affected by parameters such as the processing time of the tasks and the density of vehicles in the map. The purpose of this analysis is to compare all these criteria to measure the performance and efficiency of all the discussed protocols.

Table 5.2: Overall processed data retrieved

Protocol	Regular Traffic Density	Low Traffic Density
Topology	73.18%	75%
DBF	75.37%	63.99%
REPRO	88.93%	86.52%

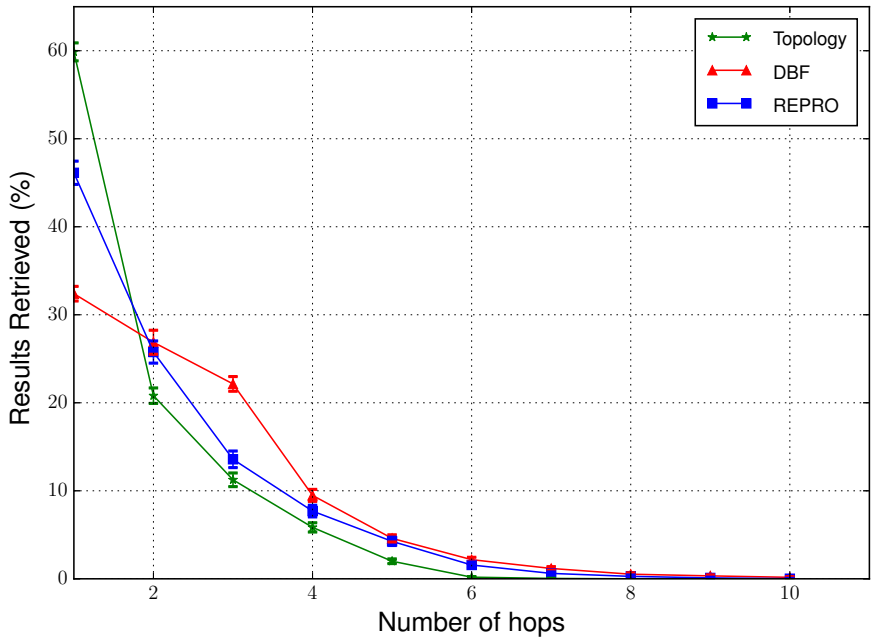
5.1 Results

In order to assess the effectiveness of the proposed MPR protocol in providing successful retrieval of the processed data, experiments have been realized within a realistic urban environment. After the results of both RSU-Triggered and Mobile Device-Triggered Task Assignment retrieval protocols were obtained, the following analysis was realized to define which protocol performs best. Similar to before, RSU-Triggered Task Assignment experiments are presented first to get a better understanding of the experiments realized. Similar parameters were used for both parts summarized in Table 5.1, but the minor changes are mentioned below. All results contain a 95% confidence interval obtained from an average of at least 35 samples.

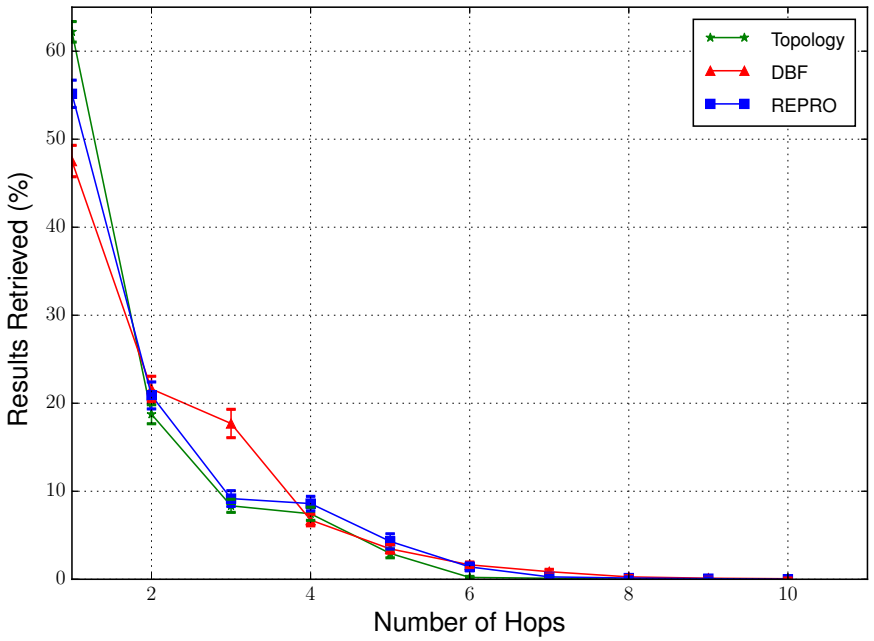
5.1.1 RSU-Triggered Task Assignment

The routes generated for the simulations include a parameter of traffic density, which refers to the number of vehicles per kilometer of lane or road per hour. Two different values were used for the experiments: a low traffic density (6 v/km/h) and a normal traffic density (12 v/km/h). These values allowed us to observe how the retrieval protocols are affected by the density of vehicle traffic. These values were chosen empirically through extensive experiments. For the sake of analysis, the experiments were also realized with a T_{max} value of $500ms$, but the results obtained were very similar to the ones of $T_{max} = 100ms$. Because the difference between the two results was an offset of unnecessary extra delay, only the graphs for $T_{max} = 100ms$ are shown. All results contain a 95% confidence interval obtained from an average of at least 35 result samples.

Table 5.2 shows the amount of processed data that was retrieved out of all the tasks that were assigned to a vehicle. According to the analysis, the only protocol that is significantly affected by the traffic density is DBF, which is to be expected because it relies mostly on V2V communication. If there are too few vehicles to forward the

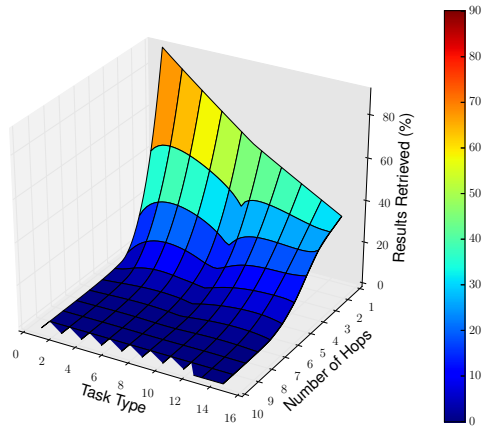


(a) Traffic density = 12 v/km/hr

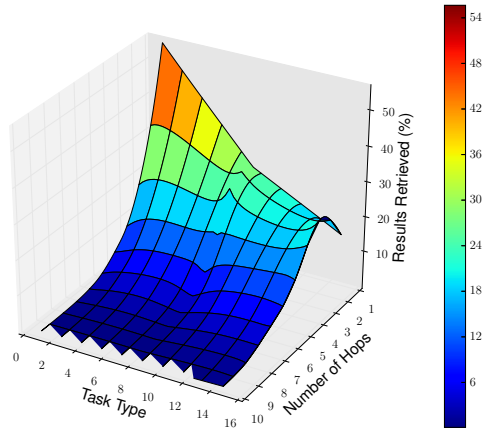


(b) Traffic density = 6 v/km/hr

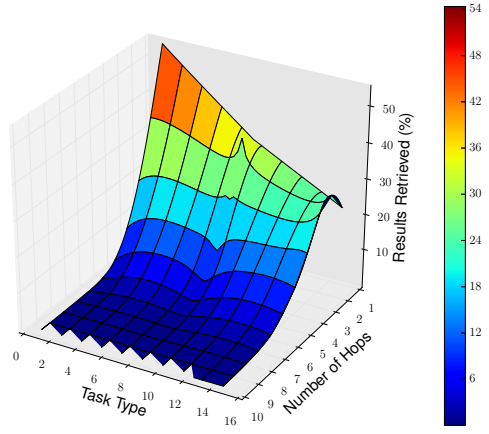
Figure 5.2: Effect of traffic density on results retrieved.



(a) Topology Protocol



(b) Distance-based Forwarding



(c) REPRO

Figure 5.3: Effect of processing time on the number of hops (density of 12 v/km/hr).

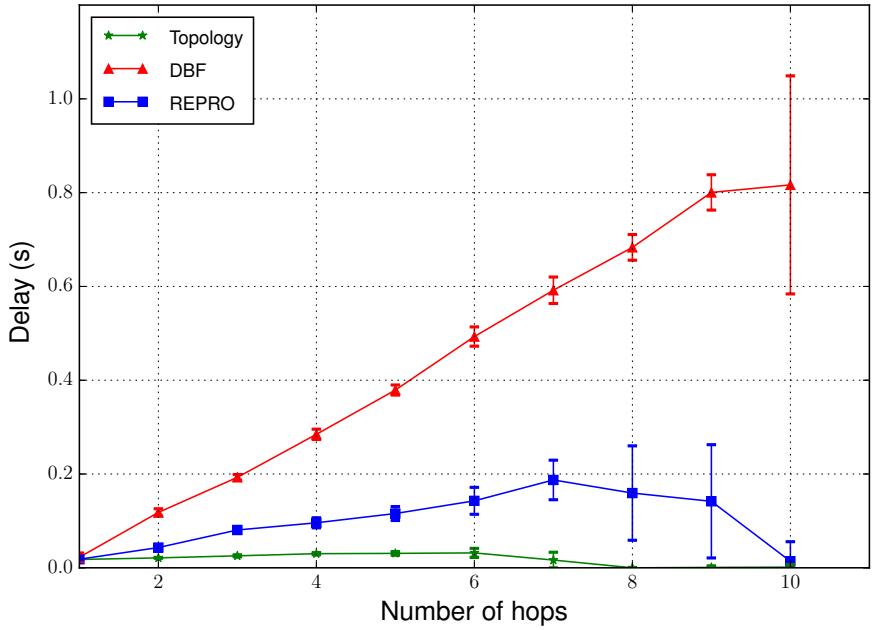
processed data, the most probable situation is that the data message will get lost at some point before reaching the sender RSU. Both Topology and REPRO protocols seem to be more balanced when concerning the traffic density in the scenario. Out of the three protocols, REPRO shows the best performance regarding the amount of processed data retrieved out of all the tasks received by a vehicle.

The following results show a more detailed analysis of the results obtained for the three protocols.

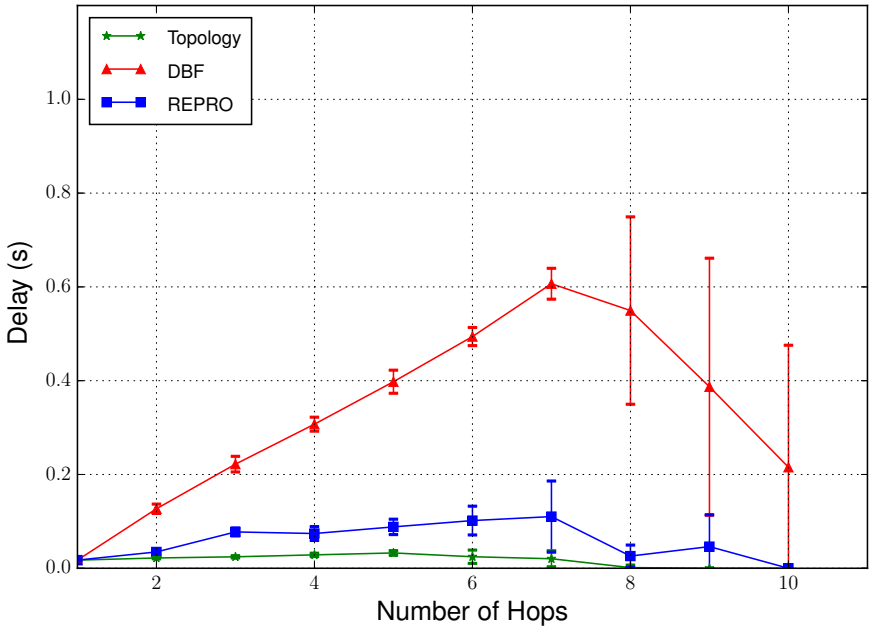
Figure 5.2 describes the effects of the variations in traffic density on the retrieval of the processed data. If we take Figure 5.2a and focus on the REPRO curve, we can observe that out of the 88.93% of results from retrieved processed data, approximately 45% was retrieved with a single hop. This behaviour repeats for the three protocols due to the fact that the tasks that only need one hop to be retrieved are the ones that have the shortest processing time. In other words, the vehicles did not travel far before they finished the computation and forwarded the processed data back to the sender RSU.

Figure 5.3 confirms the results mentioned above. These results show the average hops needed to retrieve the different types of tasks, from 1 to 15, and up to ten hops of communication. As it can be inferred, the tasks with a higher processing time need the most hops to retrieve the processed data. For instance, observing Figure 5.3c, we can see that for type 15 tasks, only about 20% of the processed data was retrieved in a single hop while for type 1 tasks, it took only one hop to retrieve more than 50% of them. As explained before, the mobility of vehicles produces this behaviour. In other words, the distance traveled while processing a task depends on the travel speed.

Figure 5.4 presents the average retrieval delay against the number of hops. This analysis measured the number of seconds required for the processed data to be retrieved in X number of hops. In this case, the behaviour of the various protocols differs. It is worth noting that as more hops are needed for retrieval in DBF, the communication delay is greater because every hop is a new vehicle that needs to perform various processes such as checking the information encapsulated in the messages. Consequently, as more hops are needed, the delay is higher. However, the behaviour is slightly different for the Topology and the REPRO protocols. Because the topology structure of the network is shaped like a square or matrix, the delay value reaches a maximum value before it begins to descend. Therefore, it makes sense that there is a maximum number of hops between RSUs before the processed data reaches its destination. Finally, it can be observed that the Topology Protocol achieves the lowest delay, followed by REPRO, with a small increase.

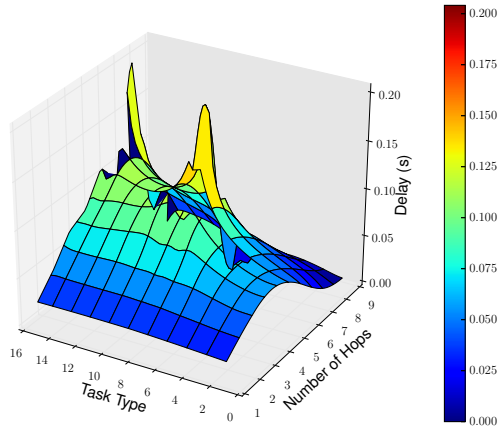


(a) Traffic density = 12 v/km/hr

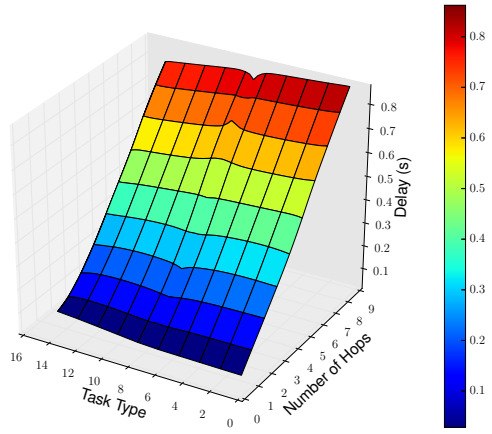


(b) Traffic density = 6 v/km/hr

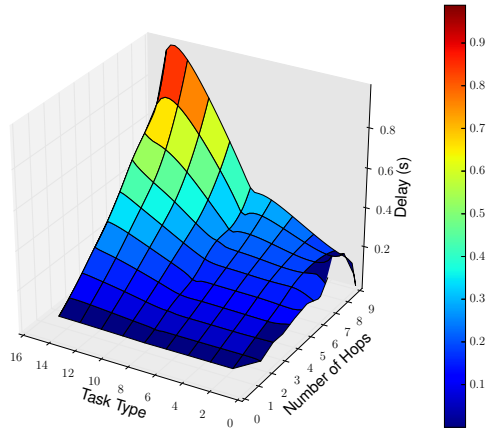
Figure 5.4: Mean delay for the different number of hops.



(a) Topology Protocol



(b) Distance-based Forwarding



(c) REPRO

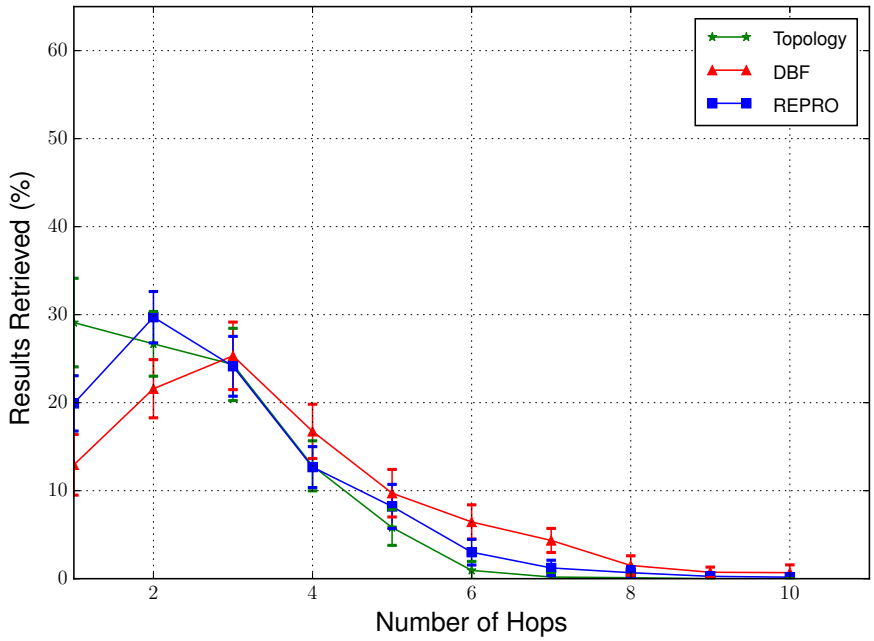
Figure 5.5: Effect of processing time on the retrieval delay (density of 12 v/km/hr).

Once again, Figure 5.5 presents a more detailed version of the retrieval delay for every type of task. It shows the effect the processing time has over the retrieval delay, and it confirms previous discussion. We can observe that the communication delay increases together with the number of hops for the DBF protocol in Figure 5.5b. Also, Figures 5.5a and 5.5c show how the delay value begins to decrease after reaching a maximum value, for the Topology-based retrieval and the REPRO protocols, respectively.

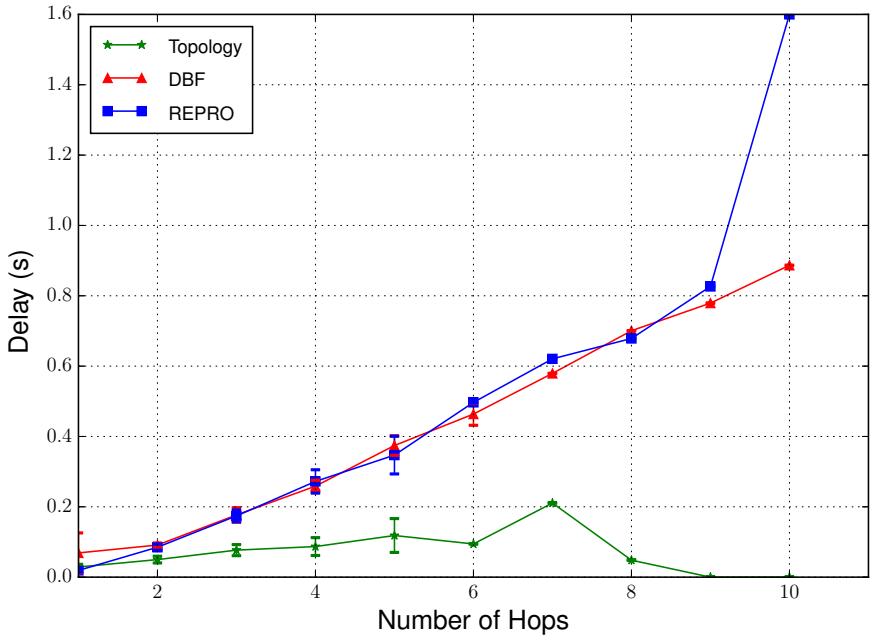
Figure 5.6 presents the results obtained in a worst-case scenario, in which a vehicle is assigned a type 15 task for the three protocols shown. These tasks have a random processing time, whose value ranges from 140 to 150 seconds [76]. On the other hand, regarding the processed data retrieved in Figure 5.6a, the three protocols show a similar behaviour. In other words, the protocols that rely more on V2V communication, REPRO and DBF, retrieve most of the processed data in two and three hops, respectively. The Topology Protocol changes its behaviour minimally, when less processed data is retrieved in overall. Regarding delay, Figure 5.6b shows that REPRO presents some changes, and it starts behaving more like DBF. This is expected, because a car can travel farther when a task takes longer to process. Therefore, it needs more hops for the processed data to be retrieved.

With the purpose of comparing the performance of the three protocols in terms of the effect of communication delay, three different deadline values were used for the experiments, *1%, 0.5%,* and *0.1%* of the processing time. Figure 5.7 shows the results obtained. As expected, fewer tasks meet the deadlines when the deadline values are lowered. Accordingly, if we focus on Figure 5.7a, we can observe that the DBF protocol is the most affected by deadlines for the three scenarios shown. DBF was the protocol with the highest retrieval delay values. Figure 5.7b shows the average time the retrievals missed the task deadlines. That being the case, the Topology Protocol shows the best performance overall, followed by REPRO.

The results shown are for the experiments realized. The REPRO protocol achieved the best retrieval rate, at more than 10% greater than the other two compared protocols. While the Topology Protocol rendered the best performance concerning the delay of retrieval, REPRO was never too far behind it. In other words, a hybrid solution, which mixes the best features of both the Topology-based retrieval and DBF protocols, achieves an efficient and more stable data retrieval performance.



(a) Results retrieved



(b) Delay

Figure 5.6: Worst-case scenario for RSU-Triggered retrieval protocols using regular traffic density.

5.1.2 Mobile Device-Triggered Task Assignment

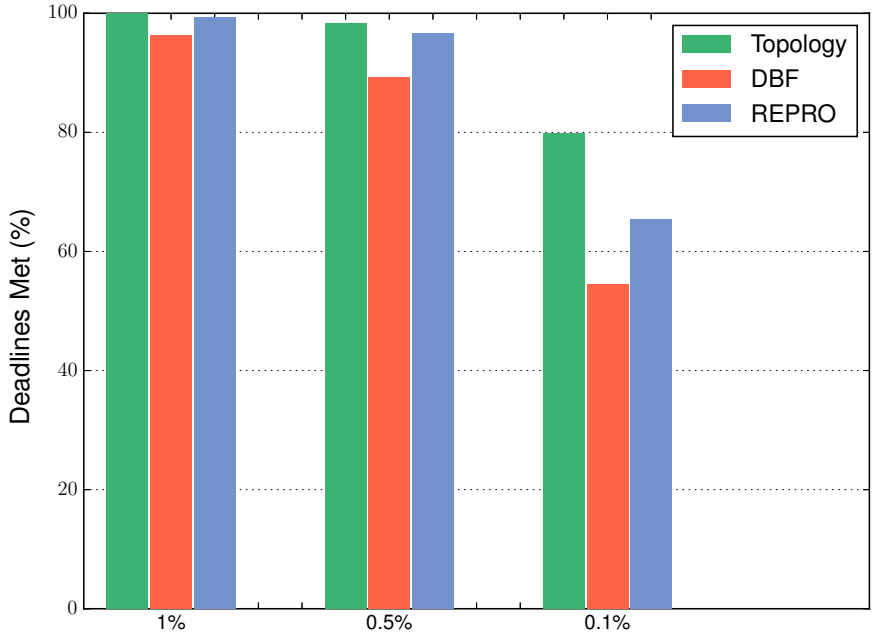
Mobile device-initiated experiments evaluate the performance of the protocols explained in Section 4.2. The density of pedestrians in the scenario is 10 $p/km/h$, which is a regular value based on the experiments.

Figure 5.8 shows the average processed data retrieved out of all the tasks that were offloaded to an RSU. In Figures 5.8a and 5.8b, we can see the general values, while in Figures 5.8c and 5.8d, a more detailed version is presented. These figures show that the GF protocol has the worst performance in terms of processed data retrieved, and retrieval delay. This is because the GF protocol only relies on vehicles to offload and retrieve the data, which makes it unreliable in scenarios with a low density of vehicles. Although the traffic density for the three protocols is the same, the performance of the VCO and the MPR protocols is significantly better. As soon as a vehicle finds an RSU, the processed data is forwarded to it. Consequently, traffic density is no longer a conditioning factor. The performance of the VCO and the MPR protocols is similar, but as we can see in Figure 5.8c, the latter shows slightly better values of retrieved data. The same situation presents in Figure 5.8d, but in this case, the difference between the two protocols is more significant. Regarding retrieval or communication delay, the GF protocol is expected to show lower values because of the simple fact that VCO and MPR add at least two more hops of communication to the process.

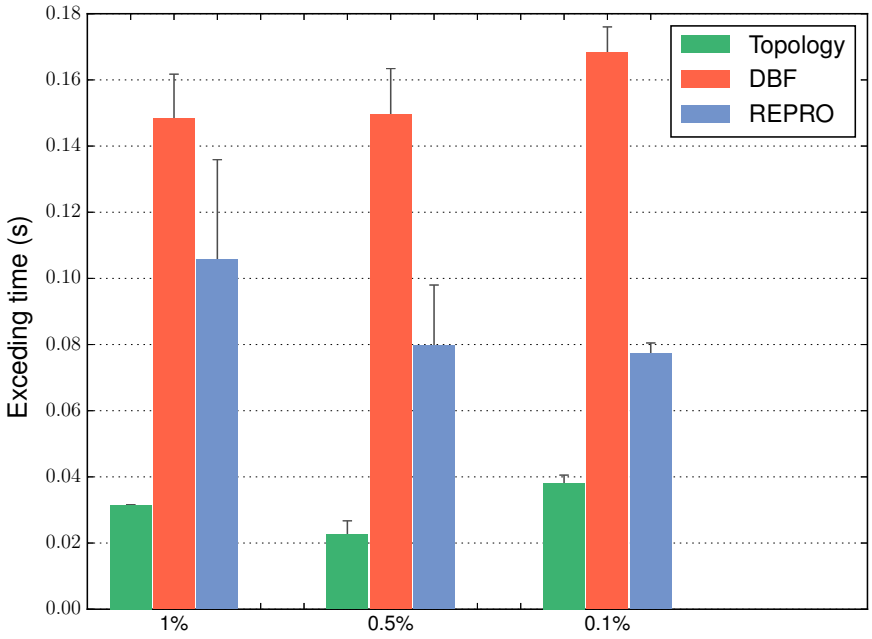
Figure 5.9 shows the performance of the prediction algorithm used in the MPR protocol. The accuracy of the algorithm is shown in Figure 5.9a. Ranges of 10, 50 and 100 meters were used to see the accuracy of the calculated coordinates against the different processing times used in the experiments $(5 - 145s)$. As expected, higher range results in better accuracy. The RSUs in the experiments have a range of approximately $250m$. Hence, a range of $100m$ is enough for a node to reach an RSU. The results show that longer task processing time results in greater difficulty to accurately estimate the position. Using the same processing times, Figure 5.9b shows the routing matches. Whenever an RSU forecasts that the sender pedestrian might not be close enough to retrieve the processed data, the data is forwarded to the RSU that is closest to the pedestrian. The term "Routing Matches" refers to the frequency in which a message was forwarded to the accurate RSU for the processed data to be retrieved. Logically, the matches for the longer processing times are lower, due to the imprecision of more than half of the estimated positions for the same times.

After the analysis of the results obtained from the experiments, the MPR protocol

achieved the best performance regarding retrieved processed data even though its prediction technique is simple. In terms of delay, GF was the one with the best performance. The importance of the delay depends on the application in which the retrieval protocol is used, but considering that the difference between the other protocols is not critical, it is safe to say that the proposed MPR protocol is the more efficient option.

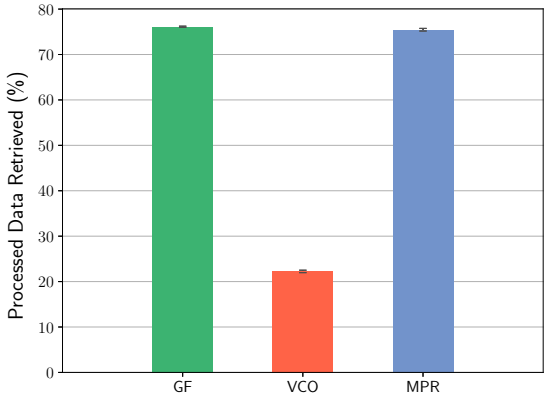


(a) Deadlines met out of all the results sent

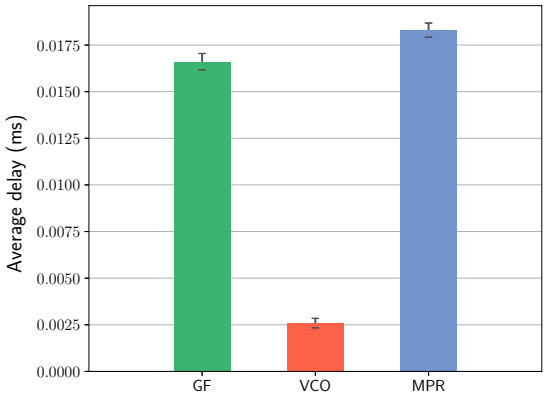


(b) Average exceeding time

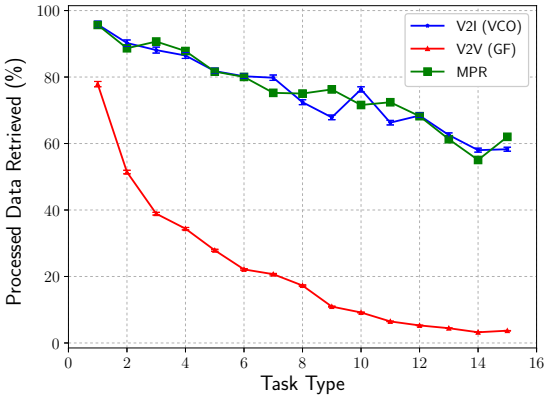
Figure 5.7: Effect of task deadlines in the retrieval of processed data.



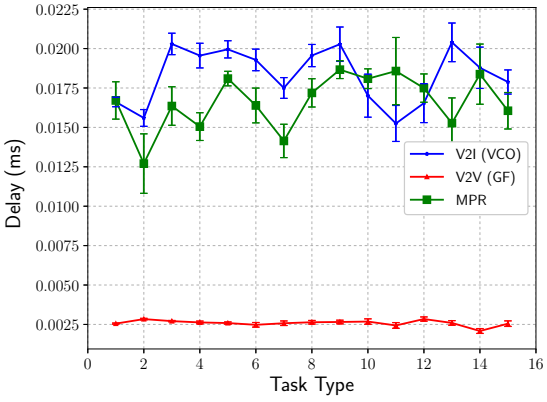
(a) Average processed data retrieved



(b) Average retrieval delay

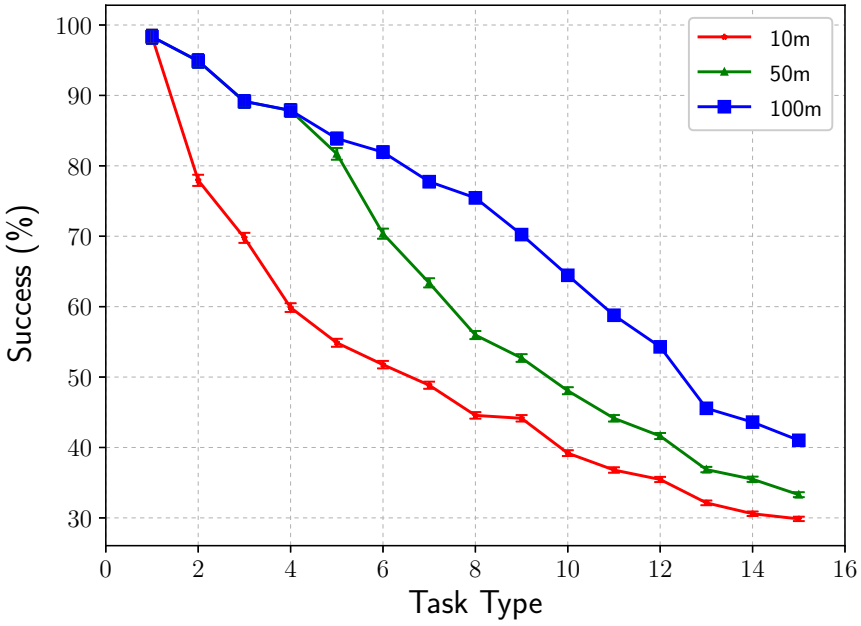


(c) Average processed data for the different processing times

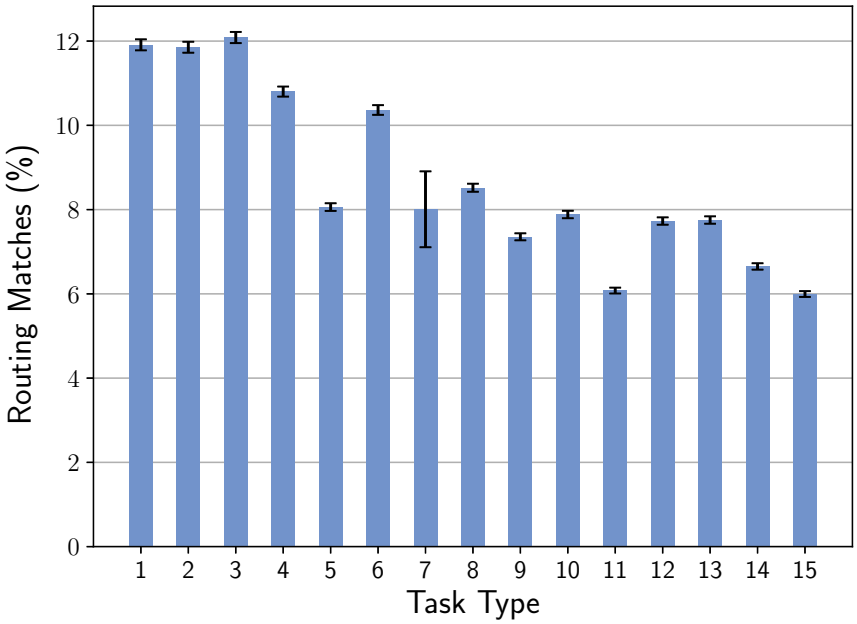


(d) Average retrieval delay for the different processing times

Figure 5.8: Performance evaluation of mobile device-initiated retrieval protocols.



(a) Accuracy performance of the prediction algorithm.



(b) Routing accurate hits (%).

Figure 5.9: Performance of the prediction algorithm.

Chapter 6

Conclusion

In this thesis, an algorithm for a data retrieval protocol is proposed. Vehicle-to-Vehicle and Vehicle-to-Infrastructure communication were used in the design of the protocol and in the experiments, so as to benefit from the underutilized resources of vehicles and help alleviate the load of the servers in the network. The use of the computation offloading method for expanding the capacity of mobile devices is increasing. This technique gives devices with low processing or energy life the opportunity of running compute-intensive applications with a lower consumption of energy by using available resources in a remote server, commonly Cloud servers. Considering these devices and many others that are arising have a range of mobility, it is necessary to solve the challenges that this presents. Using the edge of the network has become a necessary measure to solve network problems, such as bandwidth consumption and work overload in the servers. But since deploying new infrastructure is a costly solution to this, using vehicles, which count with high resources commonly underused is a promising solution. Another thing to consider is location, using vehicles for support allows the computation offloading scheme to execute highly localized content in a more efficient way, but as with the mobile devices, vehicles have a higher range of mobility. Mobility creates the issue of connection intermittency, which for task offloading is critical, not only because it makes the service unavailable, but if the process has already started, this only adds more delay which affects directly the quality of service and experience for the user. The work in this thesis is based on a Mobile Edge Computing scheme using vehicles as support infrastructure for Computation Offloading

6.1 Final Remarks

To give thorough detail, the retrieval of the output processed data in the computation offloading scheme was divided into two parts. The first one, starting from the RSU assigning tasks to vehicles in nearby and retrieving the processed data after the execution. For this part the REPRO protocol was presented, which is a resulting protocol for the retrieval of processed data designed to support the task offloading scheme in VCC, exploiting underutilized resources of vehicles to alleviate the task load of the RSUs in the network. This protocol makes use of a hybrid retrieval technique which involves the V2V communication from a distance-based forwarding protocol and the V2I from a topology protocol which uses the network map to calculate the best path to retrieve the results efficiently. The REPRO protocol proved to retrieve result messages with a high success rate with a low delay in a regular traffic density scenario and also the analysis showed that it is not significantly affected by the change in traffic density.

The second part starts from the pedestrian that holds a mobile device and sends a partitioned application to a remote server, RSU in this case, to process and it is retrieved after that. For this part two protocols and an analysis of two communication models was discussed. VCO, a data retrieval protocol for task offloading in vehicular environments, and MPR, a data retrieval protocol for a realistic offloading scheme in mobile edge computing environments which uses location prediction to estimate the location of the user to achieve a more efficient retrieval, regarding the delay, and retrieval rate. Vehicle-to-Vehicle and Vehicle-to-Infrastructure communication were used in the experiments to compare the performance of both in a Mobile Cloud Computing environment that benefits from the underutilized resources of vehicles and help alleviate the load of the servers in the network. The VCO protocol proved to retrieve results messages efficiently, but the MPR showed an overall enhance regarding both retrieval rate and delay.

6.2 Future Work

The work presented aims to further improve the overall performance of the retrieval of processed data in the computation offloading scheme. As the analysis showed, each protocol has its own advantages and drawbacks in terms of communication delay and data retrieval. Therefore, a good step towards future work will be to develop a dynamic solution which, based on the scenario, selects the optimal protocol to obtain the best performance.

Nevertheless, the improvements needed can be divided into three sections, the mobility models, scheduling, and the traffic models. Even though the accuracy of the prediction algorithm used can be improved, the use of mobility models proved to enhance the performance of the retrieval protocols when using a realistic scenario in MEC-VCC. This means, investing in a more efficient prediction algorithm, that uses mobility models with movement and behavior patterns [61], [68], can definitely improve performance for the system. In the RSU retrieval of this work, where the REPRO protocol is proposed, the results show that the tasks with higher processing time are the ones that are more difficult to retrieve. Thus, using an efficient scheduling policy [57], [35], [81] so the tasks are assigned to the node that will finish processing faster means an improvement in overall performance as well. Finaly, traffic flow models, such as [92, 95, 96], can be related to scheduling, where using a model to know the mobility of the vehicles can lead to a better scheduling performance, therefore a better performance in the computation offloading system.

Bibliography

[1] Gps query optimization in mobile and wireless networks. In *Proceedings of the Sixth IEEE Symposium on Computers and Communications*, ISCC '01, pages 198–, Washington, DC, USA, 2001. IEEE Computer Society.

[2] Semd: Secure and efficient message dissemination with policy enforcement in vanet. *Journal of Computer and System Sciences*, 82(8):1316 – 1328, 2016.

[3] K. Abrougui, A. Boukerche, and R. W. N. Pazzi. Design and evaluation of context-aware and location-based service discovery protocols for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):717–735, Sep. 2011.

[4] Yash Agarwal, Kritika Jain, and Orkun Karabasoglu. Smart vehicle monitoring and assistance using cloud computing in vehicular ad hoc networks. *International Journal of Transportation Science and Technology*, 7(1):60 – 73, 2018.

[5] E. E. Ajaltouni, A. Boukerche, and M. Zhang. An efficient dynamic load balancing scheme for distributed simulations on a grid infrastructure. In *2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, pages 61–68, Oct 2008.

[6] Deepak Ajwani, Adam Hackett, Shoukat Ali, John P. Morrison, and Stephen Kirkland. Co-optimizing application partitioning and network topology for a reconfigurable interconnect. *Journal of Parallel and Distributed Computing*, 96:12 – 26, 2016.

[7] Khadija Akherfi, Micheal Gerndt, and Hamid Harroud. Mobile cloud computing for computation offloading: Issues and challenges. *Applied Computing and Informatics*, 14(1):1 – 16, 2018.

[8] Ebrahim Al-Rashed, Mohammed Al-Rousan, and Naser Al-Ibrahim. Performance evaluation of wide-spread assignment schemes in a vehicular cloud. *Vehicular Communications*, 9(Supplement C):144 – 153, 2017.

[9] T. Antoniou, I. Chatzigiannakis, G. Mylonas, S. Nikoletseas, and A. Boukerche. A new energy efficient and fault-tolerant protocol for data propagation in smart dust networks using varying transmission range. In *37th Annual Simulation Symposium, 2004. Proceedings.*, pages 43–52, April 2004.

[10] A. Ashok, P. Steenkiste, and F. Bai. Adaptive cloud offloading for vehicular applications. In *Proceedings of the IEEE Vehicular Networking Conference*, pages 1–8, 2016.

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

[12] B. Baron, P. Spathis, H. Rivano, M. D. de Amorim, Y. Viniotis, and M. H. Ammar. Centrally controlled mass data offloading using vehicular traffic. *IEEE Trans. on Network and Service Management*, 14(2):401–415, 2017.

[13] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[14] A. Boukerche, Yan Du, Jing Feng, and R. Pazzi. A reliable synchronous transport protocol for wireless image sensor networks. In *2008 IEEE Symposium on Computers and Communications*, pages 1083–1089, July 2008.

[15] A. Boukerche, N. J. McGraw, C. Dzermajko, and Kaiyuan Lu. Grid-filtered region-based data distribution management in large-scale distributed simulation systems. In *38th Annual Simulation Symposium*, pages 259–266, April 2005.

[16] A. Boukerche and C. Tropper. A distributed graph algorithm for the detection of local cycles and knots. *IEEE Transactions on Parallel and Distributed Systems*, 9(8):748–757, Aug 1998.

[17] Azzedine Boukerche. *Handbook of Algorithms for Wireless Networking and Mobile Computing (Chapman & Hall/Crc Computer & Information Science)*. Chapman & Hall/CRC, 2005.

[18] Azzedine Boukerche and Kaouther Abrougui. An efficient leader election protocol for mobile networks. In *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, IWCMC '06, pages 1129–1134, New York, NY, USA, 2006. ACM.

[19] Azzedine Boukerche, Sajal K. Das, Alessandro Fabbri, and Oktay Yildiz. Exploiting model independence for parallel pcs network simulation. In *Proceedings of the Thirteenth Workshop on Parallel and Distributed Simulation*, PADS '99, pages 166–173, Washington, DC, USA, 1999. IEEE Computer Society.

[20] Azzedine Boukerche and Caron Dzermajko. Performance comparison of data distribution management strategies. In *Proceedings of the Fifth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, DS-RT '01, pages 67–, Washington, DC, USA, 2001. IEEE Computer Society.

[21] Azzedine Boukerche, Sungbum Hong, and Tom Jacob. A distributed algorithm for dynamic channel allocation. *Mob. Netw. Appl.*, 7(2):115–126, April 2002.

[22] Azzedine Boukerche, Richard W. N. Pazzi, and Jing Feng. An end-to-end virtual environment streaming technique for thin mobile devices over heterogeneous networks. *Comput. Commun.*, 31(11):2716–2725, July 2008.

[23] Azzedine Boukerche and Yonglin Ren. A security management scheme using a novel computational reputation model for wireless and mobile ad hoc networks. In *Proceedings of the 5th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, PE-WASUN '08, pages 88–95, New York, NY, USA, 2008. ACM.

[24] Azzedine Boukerche, Cristiano Rezende, and Richard W. Pazzi. Improving neighbor localization in vehicular ad hoc networks to avoid overhead from periodic messages. In *Proceedings of the 28th IEEE Conference on Global Telecommunications*, GLOBECOM'09, pages 5723–5728, Piscataway, NJ, USA, 2009. IEEE Press.

[25] G. Calice, A. Mtibaa, R. Beraldi, and H. Alnuweiri. Mobile-to-mobile opportunistic task splitting and offloading. In *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 565–572, Oct 2015.

[26] U. V. Catalyurek, E. G. Boman, K. D. Devine, D. Bozdag, R. Heaphy, and Lee Ann Riesen. Hypergraph-based dynamic load balancing for adaptive scientific computations. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–11, March 2007.

[27] Sobin CC, Vaskar Raychoudhury, Gustavo Marfia, and Ankita Singla. A survey of routing and data dissemination in delay tolerant networks. *Journal of Network and Computer Applications*, 67:128 – 146, 2016.

[28] B. J. Chang, Y. H. Liang, and Y. D. Huang. Efficient emergency forwarding to prevent message broadcasting storm in mobile society via vehicle-to-x communications for 5g lte-v. In *2016 International Computer Symposium (ICS)*, pages 479–484, 2016.

[29] Moumena Chaqfeh, Abderrahmane Lakas, and Imad Jawhar. A survey on data dissemination in vehicular ad hoc networks. *Vehicular Communications*, 1(4):214 – 225, 2014.

[30] M. H. Chen, B. Liang, and M. Dong. Joint offloading decision and resource allocation for multi-user multi-task mobile cloud. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2016.

[31] T. Chen, B. Zhang, X. Hao, and Y. Dai. Task scheduling in grid based on particle swarm optimization. In *2006 Fifth International Symposium on Parallel and Distributed Computing*, pages 238–245, July 2006.

[32] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[33] J. Costa, W. Lobato, A. M. de Souza, D. Rosrio, L. A. Villas, and E. Cerqueira. Centrality-based data dissemination protocol for vehicular ad hoc networks. In *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*, pages 1–4, Oct 2017.

[34] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: Making smartphones last

longer with code offload. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, pages 49–62, New York, NY, USA, 2010. ACM.

[35] Penglin Dai, Kai Liu, Liang Feng, Qingfeng Zhuge, Victor C.S. Lee, and Sang H. Son. Adaptive scheduling for real-time and temporal information services in vehicular networks. *Transportation Research Part C: Emerging Technologies*, 71(Supplement C):313 – 332, 2016.

[36] Y. A. Daraghmi, C. W. Yi, and I. Stojmenovic. Forwarding methods in data dissemination and routing protocols for vehicular ad hoc networks. *IEEE Network*, 27(6):74–79, November 2013.

[37] Mourad Elhadef, Azzedine Boukerche, and Hisham Elkadiki. Performance analysis of a distributed comparison-based self-diagnosis protocol for wireless ad-hoc networks. In *Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '06, pages 165–172, New York, NY, USA, 2006. ACM.

[38] Mohamed Eltoweissy, Stephan Olariu, and Mohamed Younis. *Towards Autonomous Vehicular Clouds*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[39] M. J. Farooq, H. ElSawy, Q. Zhu, and M. S. Alouini. Optimizing mission critical data dissemination in massive iot networks. In *2017 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–6, 2017.

[40] M. A. Hassan, Qi Wei, and Songqing Chen. Elicit: Efficiently identify computation-intensive tasks in mobile applications for offloading. In *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pages 12–22, Aug 2015.

[41] Mohammed A. Hassan, Kshitiz Bhattarai, Qi Wei, and Songqing Chen. Pomac: Properly offloading mobile applications to clouds. In *Proceedings of the 6th USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'14, pages 7–7, Berkeley, CA, USA, 2014. USENIX Association.

[42] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen. Vehicular fog computing: A viewpoint of vehicles as the infrastructures. *IEEE Trans. on Vehicular Technology*, 65(6):3860–3873, 2016.

[43] Nevin Vunka Jungum, Nawaz Mohamudally, and Nimal Nissanke. Partitioning application using graph theory for mobile devices in pervasive computing environments. *Procedia Computer Science*, 94:105 – 112, 2016. The 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops.

[44] George Karypis and Vipin Kumar. Parallel multilevel series k-way partitioning scheme for irregular graphs. *SIAM Review*, 41(2):278–300, 1999.

[45] Rohit Khandekar, Kirsten Hildrum, Sujay Parekh, Deepak Rajan, Joel Wolf, Kun-Lung Wu, Henrique Andrade, and Buğra Gedik. Cola: Optimizing stream processing applications via graph partitioning. In *Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '09, pages 16:1–16:20, New York, NY, USA, 2009. Springer-Verlag New York, Inc.

[46] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, 2012.

[47] K. C. Lee, U. Lee, and M. Gerla. Geo-opportunistic routing for vehicular networks [topics in automotive networking]. *IEEE Communications Magazine*, 48(5):164–170, May 2010.

[48] Z. Li and M. Li. An efficient social based data forwarding mechanism for mobile cloud computing. In *2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks*, pages 365–372, Dec 2013.

[49] Wen li Zhang, Bing Guo, Yan Shen, De-Guang Li, and Jun-Ke Li. An energy-efficient algorithm for multi-site application partitioning in {MCC}. *Sustainable Computing: Informatics and Systems*, pages –, 2018.

[50] B. Liu, D. Jia, J. Wang, K. Lu, and L. Wu. Cloud-assisted safety message dissemination in vanet ;cellular heterogeneous wireless network. *IEEE Systems Journal*, 11(1):128–139, March 2017.

[51] Jieyao Liu, Ejaz Ahmed, Muhammad Shiraz, Abdullah Gani, Rajkumar Buyya, and Ahsan Qureshi. Application partitioning algorithms in mobile cloud comput-

ing: Taxonomy, review and future directions. *Journal of Network and Computer Applications*, 48(Supplement C):99 – 117, 2015.

[52] Juan Liu, Yuyi Mao, Jun Zhang, and Khaled B. Letaief. Delay-optimal computation task scheduling for mobile-edge computing systems, 07 2016.

[53] Y. Liu, H. Wu, Y. Xia, Y. Wang, F. Li, and P. Yang. Optimal online data dissemination for resource constrained mobile opportunistic networks. *IEEE Trans. on Vehicular Technology*, 66(6):5301–5315, 2017.

[54] F. Malandrino, C. Casetti, C. F. Chiasserini, and M. Fiore. Offloading cellular networks through its content download. In *2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 263–271, June 2012.

[55] Francesco Malandrino, Carla-Fabiana Chiasserini, and Scott Kirkpatrick. The impact of vehicular traffic demand on 5g caching architectures: A data-driven study. *Vehicular Communications*, 8(Supplement C):13 – 20, 2017.

[56] Mohamed Nidhal Mejri, Jalel Ben-Othman, and Mohamed Hamdi. Survey on vanet security challenges and possible cryptographic solutions. *Vehicular Communications*, 1(2):53 – 66, 2014.

[57] Sadip Midya, Asmita Roy, Koushik Majumder, and Santanu Phadikar. Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach. *Journal of Network and Computer Applications*, pages –, 2017.

[58] Sadip Midya, Asmita Roy, Koushik Majumder, and Santanu Phadikar. Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach. *Journal of Network and Computer Applications*, 103:58 – 84, 2018.

[59] A. Mondal and S. Mitra. Tdmac: A timestamp defined message authentication code for secure data dissemination in vanet. In *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pages 1–6, Nov 2016.

[60] A. Mondal and S. Mitra. Revocation of misbehaving vehicles during data dissemination among connected vehicles in vanet. In *2017 IEEE Region 10 Symposium (TENSYMP)*, pages 1–7, July 2017.

[61] Klio Monokrousou and Maria Giannopoulou. Interpreting and predicting pedestrian movement in public space through space syntax analysis. *Procedia - Social and Behavioral Sciences*, 223(Supplement C):509 – 514, 2016.

[62] A. M. Mustafa, O. M. Abubakr, O. Ahmadien, A. Ahmedin, and B. Mokhtar. Mobility prediction for efficient resources management in vehicular cloud computing. In *2017 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 53–59, April 2017.

[63] T. Nadeem, P. Shankar, and L. Iftode. A comparative study of data dissemination models for vanets. In *2006 Third Annual International Conference on Mobile and Ubiquitous Systems: Networking Services*, pages 1–10, July 2006.

[64] Jianwei Niu, Wenfang Song, and Mohammed Atiquzzaman. Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications. *Journal of Network and Computer Applications*, 37:334 – 347, 2014.

[65] Jianwei Niu, Shihao Wang, Wei Niu, and Mohammed Atiquzzaman. User-aware partitioning algorithm for mobile cloud computing based on maximum graph cuts. *Computer Networks*, 129:193 – 206, 2017.

[66] Horacio A.B.F. Oliveira, Eduardo F. Nakamura, Antonio A. F. Loureiro, and Azzedine Boukerche. Error analysis of localization systems for sensor networks. In *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*, GIS '05, pages 71–78, New York, NY, USA, 2005. ACM.

[67] Ren Oliveira, Carlos Montez, Azzedine Boukerche, and Michelle S. Wangham. Reliable data dissemination protocol for vanet traffic safety applications. *Ad Hoc Networks*, 63(Supplement C):30 – 44, 2017.

[68] Itzhak Omer and Nir Kaplan. Using space syntax and agent-based approaches for modeling pedestrian volume at the urban scale. *Computers, Environment and Urban Systems*, 64(Supplement C):57 – 67, 2017.

[69] Gabriel Orsini, Dirk Bade, and Winfried Lamersdorf. Context-aware computation offloading for mobile cloud computing: Requirements analysis, survey and design guideline. *Procedia Computer Science*, 56:10 – 17, 2015. The 10th International

Conference on Future Networks and Communications (FNC 2015) / The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015) Affiliated Workshops.

[70] S. Panichpapiboon and W. Pattara-Atikom. A review of information dissemination protocols for vehicular ad hoc networks. *IEEE Communications Surveys and Tutorials*, 14(3):784–798, 2012. cited By 147.

[71] Z. q. Luo, W. k. Ma, A. M. c. So, Y. Ye, and S. Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, May 2010.

[72] Tarek K. Refaat, Burak Kantarci, and Hussein T. Mouftah. Virtual machine migration and management for vehicular clouds. *Vehicular Communications*, 4(Supplement C):47 – 56, 2016.

[73] C. Rezende, A. Boukerche, H. S. Ramos, and A. A. F. Loureiro. A reactive and scalable unicast solution for video streaming over vanets. *IEEE Transactions on Computers*, 64(3):614–626, March 2015.

[74] Andra Richa, Michael Mitzenmacher, and Ramesh Sitaraman. The power of two random choices: A survey of techniques and results. 10 2000.

[75] Dimas Satria, Daihee Park, and Minho Jo. Recovery for overloaded mobile edge computing. *Future Generation Computer Systems*, 70:138 – 147, 2017.

[76] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[77] C. Sommer, S. Joerer, and F. Dressler. On the applicability of two-ray path loss models for vehicular network simulation. In *2012 IEEE Vehicular Networking Conference (VNC)*, pages 64–69, 2012.

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

[79] Victor Soto, Robson E. De Grande, and Azzedine Boukerche. Repro: Time-constrained data retrieval for edge offloading in vehicular clouds. In *Proceedings*

of the 14th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, PE-WASUN '17, pages 47–54, New York, NY, USA, 2017. ACM.

[80] Victor Soto, Robson E. De Grande, and Azzedine Boukerche. Repro: Time-constrained data retrieval for edge offloading in vehicular clouds (accepted). In *Proceedings of the 14th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, 2017.

[81] Nasrin Taherkhani and Samuel Pierre. Prioritizing and scheduling messages for congestion control in vehicular ad hoc networks. *Computer Networks*, 108(Supplement C):15 – 28, 2016.

[82] Linlin Tang, Zuohua Li, Pingfei Ren, Jengshyang Pan, Zheming Lu, Jingyong Su, and Zhenyu Meng. Online and offline based load balance algorithm in cloud computing. *Knowledge-Based Systems*, 138:91 – 104, 2017.

[83] Neville Thomas. Dynamic grid-based multicast group assignment in data distribution management. In *Proceedings of the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, DS-RT '00, pages 47–, Washington, DC, USA, 2000. IEEE Computer Society.

[84] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, pages 60:1–60:10, 2008.

[85] C. Wang, Y. Li, D. Jin, and S. Chen. On the serviceability of mobile vehicular cloudlets in a large-scale urban environment. *IEEE Trans. on Intelligent Transportation Systems*, 17(10):2960–2970, 2016.

[86] Zi Wang, Zhiwei Zhao, Geyong Min, Xinyuan Huang, Qiang Ni, and Rong Wang. User mobility aware task assignment for mobile edge computing. *Future Generation Computer Systems*, pages –, 2018.

[87] H. Wen, L. Yang, and Z. Wang. Pargen: A parallel method for partitioning data stream applications in mobile edge computing. *IEEE Access*, PP(99):1–1, 2017.

[88] Md Whaiduzzaman, Mehdi Sookhak, Abdullah Gani, and Rajkumar Buyya. A survey on vehicular cloud computing. *Journal of Network and Computer Applications*, 40:325 – 344, 2014.

[89] P. Yang, Q. Li, Y. Yan, X. Y. Li, Y. Xiong, B. Wang, and X. Sun. Friend is treasure : Exploring and exploiting mobile social contacts for efficient task offloading. *IEEE Transactions on Vehicular Technology*, 65(7):5485–5496, July 2016.

[90] H. Yao, D. Zeng, H. Huang, S. Guo, A. Barnawi, and I. Stojmenovic. Opportunistic offloading of deadline-constrained bulk cellular traffic in vehicular dtns. *IEEE Trans. on Computers*, 64(12):3515–3527, 2015.

[91] Q. Yuan, J. Li, Z. Liu, and F. Yang. Space and time constrained data offloading in vehicular networks. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 398–405, 2016.

[92] Fan Zhang, Robson E. De Grande, and Azzedine Boukerche. Accuracy analysis of short-term traffic flow prediction models for vehicular clouds. In *Proceedings of the 13th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, PE-WASUN '16, pages 19–26, New York, NY, USA, 2016. ACM.

[93] Feifei Zhang, Jidong Ge, Zhongjin Li, Chuanyi Li, Chifong Wong, Li Kong, Bin Luo, and Victor Chang. A load-aware resource allocation and task scheduling for the emerging cloudlet system. *Future Generation Computer Systems*, 2018.

[94] K. Zhang, Y. Mao, S. Leng, A. Vinel, and Y. Zhang. Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks. In *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 288–294, Sept 2016.

[95] Tao Zhang, Robson E. De Grande, and Azzedine Boukerche. Vehicular cloud: Stochastic analysis of computing resources in a road segment. In *Proceedings of the 12th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, PE-WASUN '15, pages 9–16, New York, NY, USA, 2015. ACM.

[96] Tao Zhang, Robson E. De Grande, and Azzedine Boukerche. Design and analysis of stochastic traffic flow models for vehicular clouds. *Ad Hoc Networks*, 52:39 – 49, 2016. Modeling and Performance Evaluation of Wireless Ad Hoc Networks.

[97] J. Zhao and G. Cao. Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 57(3):1910–1922, 2008. cited By 470.

[98] P. Zhao, L. Feng, P. Yu, W. Li, and X. Qiu. A social-aware resource allocation for 5g device-to-device multicast communication. *IEEE Access*, 5:15717–15730, 2017.