TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2020 115

Óscar Urra Cuairán

Distributed Data Management in Vehicular Networks Using Mobile Agents

Departamento
Informática e Ingeniería de Sistemas

Director/es
Ilarri Artigas, Sergio

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

DISTRIBUTED DATA MANAGEMENT IN VEHICULAR NETWORKS USING MOBILE AGENTS

Autor

Óscar Urra Cuairán

Director/es

Ilarri Artigas, Sergio

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2018

Distributed Data Management in Vehicular Networks Using Mobile Agents

Óscar Urra Cuairán

Tesis Doctoral

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Mayo 2018

A mis padres, Ramón † y Maribel,
a mi hija Rocío,
a mi compañera Inés.

Per aspera ad astra.

Acknowledgements

This thesis is the result of several years of patient work that now come to an end, or at least, to the end of a stage. The expression *several years* has a special meaning here, since in my particular case I performed the preparation of the thesis at the same time that I had an unrelated full-time job outside the university, which made the task certainly harder and longer.

I would first like to thank my supervisor, Sergio Ilarri, who guided me through this journey in the research world, by providing me wise and useful suggestions, revisions and advices. He was always willing to help me and had good answers to my questions, and had always his office and e-mail open for me, at any time. Without his help and supervision, this work would not have been possible, and I am especially grateful for accompanying me in this obstacle course, that was also a marathon, for such a long time without decaying and for being so enthusiastic and interested as the first day.

I am also grateful to Eduardo Mena, who provided me suggestions during the beginning of this thesis, as well as to other people related to the University of Zaragoza with whom I shared meetings or simply exchanged ideas, such as (I cite them in no particular order) Raquel Trillo, Mª Carmen Rodríguez, Ángel Luis Garrido, Carlos Bobed, Roberto Yus, Thierry Delot and others that I may have forgotten (I apologize in such a case).

I must also thank the Instituto Tecnológico de Aragón, my workplace, that funded my first travels to conferences and allowed me to spare that time, even if my job there was not related to the topics of my thesis.

And last but not least, I would like to express my gratitude to my family and beloved ones. A special memory for my father Ramón, who passed away during the preparation of this thesis and could not see me ending it. Many thanks also to my mother Maribel, my brother Javier, and my significant other Inés. All of them supported and encouraged me unconditionally to carry on, and made my life much easier with their deeds, that helped me to dedicate more time to the preparation of this work. Without them, this effort would be meaningless.

Zaragoza, May 2018
Óscar Urra Cuairán

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

3G	Third Generation (mobile telecommunications)
4G	Fourth Generation (mobile telecommunications)
ACL	Agent Communication Language
API	Application Program Interface
CPU	Central Processing Unit
DSRC	Dedicated Short-Range Communications
FIPA	Foundation for Intelligent Physical Agents
GB	Gigabyte
GLONASS	Global Navigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Position System
GPU	Graphics Processing Unit
GSM	Global System for Mobile communications
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronics Engineers
ITS	Intelligent Transportation System
KB	Kilobyte
LBS	Location-Based Services
LIDAR	Light Detection and Ranging
MANET	Mobile Ad hoc Network
MB	Megabyte
NDP	Near-Data Processing
P2P	Peer-to-Peer
PAN	Personal Area Network
PDA	Personal Data Assistant
QoS	Quality of Service
RADAR	Radio Detection and Ranging
RAM	Random Access Memory
RSU	Roadside Unit
SC	Spatial Crowdsourcing

SIM	Subscriber Identity Module
SMS	Short Message Service
SQL	Structured Query Language
SSID	Service Set Identifier
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
UWB	Ultra Wide Band
V2I	Vehicle-to-Infrastructure (communications)
V2V	Vehicle-to-Vehicle (communications)
VANET	Vehicular Ad hoc Network
Wi-Fi	Wireless Fidelity
WAN	Wide Area Network
WAVE	Wireless Access in Vehicular Environment
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network

Chapter 1

Introduction

The work presented in this thesis belongs to the topic of mobile computing, and particularly data management in vehicular networks. In these networks, large amounts of data are constantly generated, but their management (creation, transmission, querying, etc.) is a challenging task due to the high number of limitations and drawbacks typical of this type of scenarios. To overcome these obstacles, we propose a series of novel techniques that make the efficient management of data in vehicular networks possible, and that can lead to the development of new applications and services in this area.

1.1 Context of the Thesis

In this thesis, we explore different possibilities related to the management of data within the scope of a vehicular network, and more specifically regarding the problem of how to effectively and efficiently access and transmit those data. In such a scenario, the vehicles that travel through an area (such as a city, a highway, or a small town) can communicate among them using radio-like devices that allow them to exchange information wirelessly with other nearby vehicles. These vehicles are constantly receiving data from different sources. Key data sources are the sensors that are present in modern vehicles, that provide information such as the speed, the fuel level of the vehicle, the currently engaged gear, the oil temperature, or the geographic position of the vehicle (using a GPS receiver). Other data sources are other vehicles present in the surroundings, as well as roadside units (RSUs) that can transmit data of different nature, such as information of interest for the drivers (e.g., about traffic jams or accidents ahead in the route), or of any other type of data (e.g., data collected to perform cooperative surveillance or environment monitoring) as long as it can be digitized and transferred using a network connection.

All these data that are originated in the vehicular network can be very heterogeneous in their nature: they can have different magnitudes and measurement units (e.g., liters for the amount of fuel, kilometers per hour for the speed, longitude and

latitude in sexagesimal degrees for the geographic position, "present" or "not present" for the occupancy of a seat, etc.) and also exhibit different *variability* (e.g., the geographic position varies constantly, but not the occupancy of seats in a vehicle) as well as *utility* along time (e.g., the presence of a traffic congestion in the route will be more useful if it is received as soon as possible, rather than with a significant delay, when it will likely be outdated).

Due to this variety of data types, as well as the distribution of the data sources and the use of wireless communications, a management system with a high level of flexibility is needed. It should be able to handle not only the current existing data, but also future new types that may appear without the need of developing a completely new system. With this purpose, we have created a number of algorithms and defined an architecture (which consists of both software and hardware components) that, when working together distributively, allows the data present in a vehicular network to be properly managed during all their lifetime.

1.2 Motivation

The high number of vehicles traveling along the streets of the cities all over the world has led to a number of problems, such as an increase in the number of deaths in traffic accidents [Adm16] and pollution due to exhaust gases, which in turn leads to a higher number of diseases [HN06] and decreases the quality of life in big cities. This has caused the implementation of drastic palliative measures such as the limitation of speed and traffic circulation in urban centers when the pollution reaches high levels, with examples in Madrid and London [PPM17, KAA+11]. One possible solution might be the generalization in the use of electric vehicles [HMMY16], which are more efficient and have much lower levels of pollution, but they currently have a number of problems that will limit their implantation for some years [HMS14]: they have a higher economic acquisition cost, they can travel a shorter distance, and they need an extensive network of charging stations which is slowly being built.

Nevertheless, and despite the eventual implantation of electric vehicles, a number of challenges may persist, such as the problem of traffic congestion during rush hours, the scarce availability of parking spaces in certain high-demand areas (which leads to the waste of time of drivers and also has an economic and environmental impact due to unproductive driving), or traffic accidents due to human errors and technical failures.

To alleviate these problems, Intelligent Transportation Systems (ITS) [DD10] have been proposed as a means to increase the efficiency and sustainability of the transportation while increasing its safety and reducing the emissions of pollution gases. This also means that the roads and streets could increase the amount of traffic that can be accommodated without causing congestion, the travel times could be predicted more accurately and reduced, and the number of traffic accidents decreased. To try to accomplish all this, these systems make use of computing and communication technologies to coordinate the different elements of a transportation system (such as vehicles, roads, traffic signs, etc.) and make these elements *intelligent*, so

that they can be programmed to be capable of performing their tasks in a flexible and autonomous way, adapting their operation to the changes that may occur in their operating environment or conditions.

Such is their importance, that the European Community adopted on July 7th, 2010, a new legal framework, the Directive 2010/40/EU [Eur10], to accelerate the deployment of these innovative transport technologies. The objective of this directive is to coordinate the implementation of ITS in Europe, allowing the interoperability, compatibility and continuity of different solutions across the European Union. Another sign of the importance of this issue is that, previously to that directive, on August 5th, 2008, the Commission adopted the Decision 2008/671/EC [Eur08] to reserve the 5.9 GHz band for safety-related ITS applications, which require a spectrum for short-range, low-latency communications.

Regarding the vehicles that belong to these systems, in the last years the automotive industry has enhanced them with the latest advances in communication and information technologies, to the extent that modern cars could be easily defined as *computers on wheels*. It even exists a term, *intelligent vehicle*, to name a car with not only the capability to improve the driving experience and the security of their occupants thanks to computers, but also the capability to communicate with other similar cars. This idea of a *connected vehicle* allows to expand traditional techniques to provide comfort and safety to a wider extent.

The vehicles that communicate among them in this way can then build a vehicular ad hoc network (or VANET) [OW09], that can be used for the development of many interesting applications. The use of wireless ad hoc communications supports the quick exchange of useful information among nearby vehicles, for example, the exchange of useful information among the vehicles about accidents, traffic congestions ahead and possible detours to avoid them, hazardous roads and weather conditions, the location of certain facilities (e.g., gas stations, hotels), the activation of an emergency brake by a nearby vehicle which may not be visible to the driver, etc. Another application that would take advantage of the wide area covered by the vehicles when they travel is to use them as moving sensor platforms. In such an application, the vehicles would carry different types of sensors, that would obtain data from the vehicles' surroundings as they move.

However, and despite the potential utility that vehicular networks could have, there exist a number of difficulties that have limited their growth and popularity: the information exchange can be performed only by vehicles within a relatively short range, which in addition is limited by the presence of obstacles that can block the wireless signals (see Figure 1.1), and by the short time interval during which the devices are in range, due to the vehicles' movement especially when they are traveling in opposite directions at high speed, for example in a highway. Another difficulty is how to send the data to the vehicles that might be interested in them, taking into account that the possible receivers may be located in different and unknown places along the area covered by the VANET. Moreover, some vehicles can be interested in obtaining certain information, but they may not know how or from where such information can be retrieved.

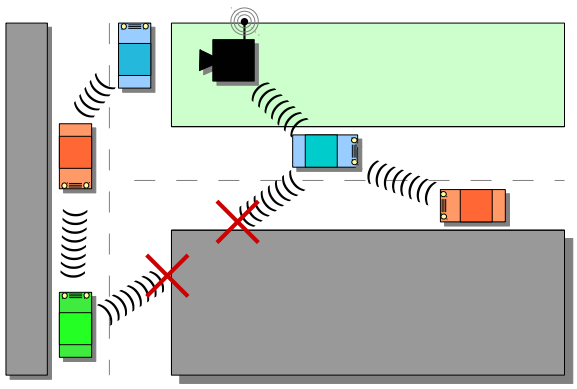


Figure 1.1: Ad hoc communications in VANETs and potential obstacles

So, the main motivation of this work is the possibility to exploit data in a VANET as a first step to provide data services for drivers and passengers (in order to increase their safety and convenience) as well as for third parties (e.g., to perform environment monitoring by using conventional vehicles). For that purpose, we aim at the creation of a system that allows:

- To transfer data in a VANET in an efficient and fast way, using short-range network connections that are free to use, i.e., without implying an economic cost and without needing a previously-existing infrastructure.

- To route data to any destination (a vehicle or place) in the network in an easy and transparent way, with independence of where that destination is located, and if it is moving or in a fixed place.

- To query the data stored by the vehicles within any spatial area, to retrieve information according to the interests or needs of the users of the system.

- To support an easy update of the data management strategies applied and their efficient use in a distributed environment. For this purpose, we will study the use of mobile agents to solve some existing data management challenges in the context of vehicular networks, since it is a promising but quite unexplored technology for this type of scenario.

The creation of this system could have, as an additional benefit, the possibility of developing new tools and services on top of it, and in this way it would *give value* to such an amount of data that at the present time remains unexploited and hidden in the vehicles that run through our cities. Moreover, the development of that system will also imply efforts in other related areas, such as the definition of an evaluation methodology and appropriate simulation strategies.

1.3 Overview of the Work

The main contribution of this work is a system that allows the management of the data present in a vehicular network based on the use of mobile agents to perform that task. The data are scattered among the vehicles that belong to the network, and some portions of those data can be transferred from one vehicle to another using their short-range wireless communication devices. These exchanges of data are made in a peer-to-peer (or P2P) way, so their scope is very limited and it is necessary to perform further actions to transport the data to other places that are out of the range of those devices.

One way to achieve this is by using the technology of mobile agents, which consists of software programs with the ability to moving themselves (both their code and data) from the computer/device where they are being executed to another one after being transferred using a network link, resuming its execution at the same point once they are in the new computer/device. Thus, a mobile agent can benefit from the vehicles' short-range communication capabilities to move from one to another in its vicinity and, by repeating this process many times, eventually reach distant areas (see Figure 1.2).

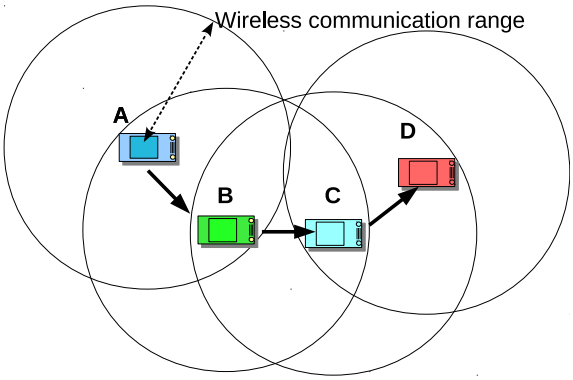


Figure 1.2: Reaching a target area by jumping among vehicles

Another objective is to allow the users to query the data that are stored by the vehicles to extract useful information and obtain the required result within a reasonable time. These queries can refer to any parameter read by the sensors aboard the vehicles, at any time period, and in any geographic place where the vehicles of the network can travel.

Thus, we propose a mechanism to use mobile agents' features to address two key challenges. Firstly, to retrieve data from specific spatial areas by using the vehicles as intermediate communication nodes: a mobile agent will need to take decisions autonomously and intelligently regarding the choice of a vehicle within its communication range where it should jump to (or, alternatively, to conclude that, for the moment, it should remain in its current vehicle), in order to reach its final spatial

destination as soon as possible. Secondly, to solve queries defined by the users by processing the data locally at the vehicles (instead of sending all the data to a central server, which is problematic in a VANET); in this way, it is possible to discard data that are irrelevant to the query and keep only the relevant data, aggregating them to form a complete query answer.

Furthermore, motivated by the lack of simulation software capable of simulating, at the same time, the movements of vehicles in a road network, the actions performed by mobile agents, and the communications involved, we have also developed a simulator, called MAVSIM. This software allows the simulation of VANETs in a realistic way, using maps from real cities and roads, considering the limitations of wireless communications (such as their short range and the potential blocking of communication signals by buildings), and also supporting the simulation of mobile agents that can transfer themselves among the simulated vehicles. These agents can follow the algorithms developed by the user who wants to use the simulator to evaluate data management strategies using mobile agents. The simulator has been programmed in a modular way so that it can be easily used for the simulation of any scenario needed, and it can also be enhanced with new functions if necessary.

Finally, we have performed extensive tests and simulations to evaluate the different approaches developed to use mobile agents for data management in VANETS, under many circumstances and with different purposes. The results show the interest and feasibility of using mobile agents in such a dynamic and uncertain scenario for locating, processing and transporting data to/from different places in the VANET. Due to the difficulty of evaluating the different approaches that use mobile agents moving in a VANET in the real world, we have used the MAVSIM simulator to test those approaches in different scenarios and conditions. For example, a typical simulation scenario consists of a real city where a mobile agent must perform some task using the tested algorithm, which involves its movement to a certain area placed a few kilometers from the starting point of the agent. The simulation can be repeated as many times as necessary, varying the initial conditions (such as the distance to the target area, the number of vehicles moving in the VANET, the range of communications, etc.) and, once each simulation ends, it provides different data as a result, that are analyzed to evaluate the performance of the approach being tested.

In the following, we describe the main contributions of our work with more detail. Firstly, we introduce the problem of data management in vehicular networks. Secondly, we provide some explanations about mobile agents and the benefits of using them in vehicular networks. Thirdly, we describe the simulator that was developed to help in the evaluation of our proposal. Finally, we explain the experiments that were performed to evaluate our approach.

1.3.1 Data Management in Vehicular Networks

Our data management approach is capable of overcoming a number of problems caused by the limitations of vehicular networks. Its main features are:

1. *It can exploit the data that are scattered among the vehicles.* The vehicles in the

VANET are constantly obtaining data from their sensors and from other vehicles as they move, and these data are stored locally in their onboard computers, along with information about the geographic position and the time when they were obtained. The users of the system can search a subset of those data satisfying a number of conditions previously defined, that can refer to spatio-temporal constraints and/or to the values of the data themselves, and can be specified in the form of *queries*. Then, the data that meet the conditions are located and returned, no matter the number of vehicles involved.

2. *It uses P2P connections for communications.* The proposed data management approach does not need the existence of a previously-installed communication infrastructure, using instead Wi-Fi type wireless communication devices. A drawback of these devices is that they have a short range, which allows them to establish connections only with other similar devices within a range of a few hundred meters, but on the other hand they also have important advantages that make them ideal for this scenario. Among these advantages we can highlight that they are inexpensive, standard and widely adopted, and there exist many options to choose from (e.g., IEEE 802.11, UWB, WiMax, Bluetooth, Zigbee, etc.), so it is highly likely that any two random vehicles will be able to communicate with each other by using some of these technologies. Another important advantage is that the vehicles can initiate or end communications at any time with any other peer in range, without needing to register previously to, or needing the approval or intervention of, any other control entity.

3. *It operates in a decentralized and distributed way.* The data are obtained, stored and processed locally by the vehicles, instead of being transferred or submitted to a centralized location or entity to perform these tasks which, on the other hand, would be problematic using P2P short-range communications, and would also require a lot of computing power if the amount of data to process is high. Instead, the data are stored and processed using the computer aboard every vehicle, which has benefits such as the distribution of the computing power, the enhancement of the privacy (the data owner is who controls their use), and the fact that the existence of central servers is not necessary (since all the vehicles act as equal peers).

4. *It supports the transfer of data from any vehicle to anywhere in the VANET.* In a vehicular network with P2P short-range connections whose nodes (i.e., the vehicles) are constantly moving, it can be difficult to transfer data from one place to another distant one. However, we can achieve this by using algorithms that use intermediate vehicles as relays that store temporarily the transferred data until they can be retransmitted to another vehicle. So, the data are *moved* along the VANET both wirelessly (when one vehicle transfers them to another) and physically (when the vehicle transporting the data moves to another geographic area). Our approach is able to determine the best option dynamically at any time, so that the data will eventually arrive to the destination.

As a summary, the proposed approach takes advantage of the features of vehicular networks (e.g., their high mobility, which allows the vehicles to cover wide areas and potentially obtain large amounts of data from the environment) and tries to overcome their difficulties (e.g., the instability of the network connections). The data management techniques developed are functional, flexible and robust enough to allow the development of new applications.

1.3.2 Mobile Agents in Vehicular Networks

In order to make it possible that the data in the VANET can be queried and transported using the vehicles as network/processing nodes, we have decided to explore the use of *mobile agent* technology, whose concept has existed since some years. A mobile agent is a program (code and data) that, while it is being executed in a computer, can suspend its execution at any time and transfer itself to another computer using a network connection, then resuming its execution in the new computer.

The reason for choosing mobile agents is that they have a number of interesting properties that make them the ideal election to deal with such a changing and unpredictable scenario as a vehicular network. The main features of mobile agents are that they can potentially be intelligent (as long as they are programmed to embed an intelligent behavior), autonomous (they are not tied to a single execution environment and can be programmed to take decisions autonomously), communicative (they can exchange data with other agents or with the computer where they are currently being executed), and mobile (they can move to other execution environments as long as a network connection exists).

The use of mobile agents in our approach has important benefits, since they allow to implement, in a suitable way, the following aspects of the proposed data management approach for vehicular networks:

- The mobile agents can be executed in any computer aboard vehicles and interact with the sensors to retrieve the data read by them, only requiring the installation of a lightweight software framework, called a *mobile agent platform*. This platform provides, among others, a mobility service needed by the mobile agents to move from one execution environment to another, as well as a communication service that allows them to interact with each other.

- By means of the mobile agent platform, they can transfer themselves from one vehicle to another using the wireless communications, despite their limitations (short range, brief availability of links due to the movements of vehicles, etc.). The action of transferring the mobile agent from one execution environment to another is efficient and does not take more than a few seconds at most, so the mobile agents can move freely among the vehicles in a VANET.

- Due to their intelligence and mobility features, they can travel to any place of the VANET by hopping from one car to another until they reach their destination. Since the vehicles are constantly moving, every time an agent reaches a new one, it evaluates the properties of other vehicles in range (such as their position,

speed, heading, etc.) constantly, in order to decide which one will be the next to hop to in order to reach the destination in a shorter time. It is also possible that the agent decides to stay in the same vehicle, if hopping to another does not bring any benefit.

- By using this way of moving from one location of the VANET to any other, mobile agents can be used to transport the data of interest for the users of the system. In other words, they can provide the necessary intelligence for the data to be transferred anywhere in the VANET using the always-changing network connections available.

- Additionally, mobile agents can be used not only to transport data through the vehicular network, but also to fetch them directly from the vehicles present in a delimited area, by processing the data stored in their computer devices in a decentralized way. In this case, the mobile agent will transfer itself from one vehicle to another within the specified area, and at every interesting vehicle, it will process or query the locally-stored data to obtain the required data. The process will be repeated until the mobile agent finds the data being searched.

- In order to increase the efficiency of these processes, some enhancements are possible. One is to use many simultaneous copies (or *clones*) of the mobile agent to search for the data and, in this way, increase the probability of finding them and/or enlarge the amount of data recovered, taking into account that some data will probably be duplicated and the process will need more resources. Another improvement is helping the mobile agent to find or reach the searched data by asking the vehicles' drivers to collaborate with the agents by carrying them to the area where the data are located, using a spatial crowdsourcing scheme to exchange physical transportation with some type of compensation.

An example of the usage of mobile agents in a vehicular network can be seen in Figure 1.3, where the agent hops from one vehicle to another until it reaches a target area, dealing with unexpected situations such as vehicles that change their route in directions that are not favorable for the agent.

Given all these features of mobile agents, as well as the way they can be applied to build our approach, we believe that using this technology is an appropriate choice. We hope that it will help to contribute to increase the current knowledge of the use of mobile agents for data management in vehicular networks (and mobile networks in general), and also in other scenarios.

1.3.3 MAVSIM: A Simulator of VANETs to Evaluate Data Management Approaches Using Mobile Agents

The proposed data management approach has been extensively tested and analyzed using simulation software, since performing the necessary tests in a real VANET would have been extremely complex and expensive, if not impossible.

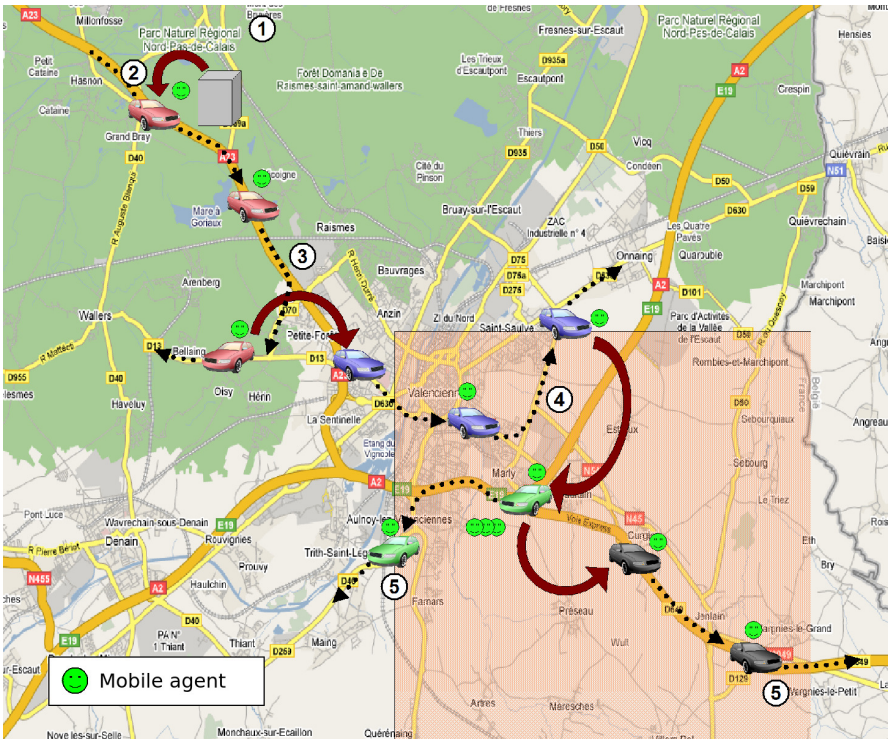


Figure 1.3: Mobile agents hopping in a VANET in the city of Valenciennes (France)

Due to the novelty of this approach, there is no simulation software available with the capability to simulate a VANET and mobile agents at the same time, which is needed to evaluate our data management solution properly. In fact, there are many simulation tools for networks, and also for transportation and traffic, but they are not useful for the simulation of environments where our data management approach can be directly tested. The main reason is that they cannot simulate mobile agents, which are necessary for the implementation of our proposal, and they cannot simulate either that the transmitted data may have an influence on the behavior or movements of the vehicles. For example, the notification of a traffic jam in an area makes the users of the application to take a detour in order to avoid the traffic congestion (altering the original vehicle's route). As another example, existing tools would not support testing spatial crowdsourcing techniques, that compensate drivers for altering their routes to travel physically to a certain location of interest.

Due to these reasons, and motivated by the lack of simulators with the ability to simulate programmable mobile agents in a flexible way, we developed an entirely-new simulation software, called MAVSIM (Mobile Agent Vanet SIMulator), that allows us to test the algorithms used in our data management proposal, as well as any other similar algorithm, in a realistic way and using real city maps. Among the most

interesting features of the simulator, we could cite:

- Simulation of traffic using roads and streets from maps of real cities.
- Simulation of both moving and static wireless communication devices.
- Simulation of constraining factors that are typical of mobile networks, such as: limited communication range, signal blocking by buildings, latency and transmission errors, etc.
- Simulation of a mobile agent platform that allows the programming and execution of arbitrarily-complex mobile agents.
- Support for mobile agents to obtain information and/or properties from any simulated device where they are executed (e.g., the position of a vehicle, its speed and heading, etc.) and also to move freely to any other device using the simulated communication devices, with the only limitation of the mobile communication constraints.
- Simulation of fixed roadside units that allow the mobile agents to move quickly among distant places using a wired link that inter-connects all these units.
- Generation of trajectories using different mobility models and support to use them along with trajectories from real traffic traces or from third-party traffic simulators.
- Simulation of public transportation lines involving vehicles, stops, and fixed and cyclic routes.
- Execution in batch mode for the simulation of large or complex scenarios in high-capacity servers.
- Recording of entire simulations for their later analysis using a graphical replaying tool.
- Modular development, to ease its extension.

Summing up, the simulation tool developed allows the definition and testing of different data management approaches for a VANET. The mobile agents programmed can be simulated in different scenarios in order to test their behavior under different conditions and with the typical limitations and difficulties of mobile and wireless communications, and the results can be analyzed to enhance the algorithms or detect adverse conditions. In Chapter 5, we explain the features of MAVSIM in more detail.

1.3.4 Experimental Evaluation

The data management approach that we have developed involves a number of variables that define its behavior. In addition, the scenario where it is used (i.e., a VANET) may become quite complex due to its unpredictable nature. That is, the high number

of vehicles moving at the same time towards unknown destinations, and the instability of wireless communications, can make the task of our mobile-agent based approach really challenging. Therefore, in order to validate the feasibility of our proposal, it is essential to test it in different scenarios under different conditions and varying the values of the parameters that define its behavior.

We evaluated the different features of our approach by performing several experiments, where the data obtained as a result were analyzed to draw conclusions. These conclusions were used later to enhance and refine the tested approach or to perform further experiments if the results obtained initially were unclear or yielded new questions. Among the most relevant experiments performed, we can cite the following:

- Measurement of the average transmission time of mobile agents when they move wirelessly from one mobile device to another.
- Testing of different *hop strategies* used by mobile agents to approach an area by moving (hopping) among the vehicles in a city.
- Determination of the best hop strategy by taking into account the total time needed, the number of movements performed by the mobile agent, etc.
- Comparison of the performance of mobile-agent based algorithms according to the features of the street layout of different cities.
- Measurement of the reliability of the algorithms and the influence of communication errors, the occurrence of unexpected events, the existing uncertainty regarding the position of a target vehicle, etc.
- Analysis of the influence of the presence of obstacles (such as buildings), that block the wireless signal propagation, on the performance and reliability of the different approaches.
- Testing of the use of multiple instances of a mobile agent (*clones*) and assessment of the advantages and drawbacks of their use in terms of query processing time, performance, bandwidth usage, reliability, etc.
- Testing of the use of spatial crowdsourcing techniques and their advantages and drawbacks in terms of query processing time, compensation costs, reliability, etc., comparing them with other solutions that do not use such techniques.

Among these experiments, the first one was performed using real mobile devices and, for the rest of them, the MAVSIM simulator was used, completing more than 2500 hours of simulations. In Chapter 5, we explain the methodology followed for the experimental evaluation, and in Chapter 6 we detail the experiments performed.

1.4 Structure of the Thesis

This thesis is composed of nine chapters, including this one. In Chapter 2, we first review the technological context of this work, where we explain some topics related to mobile computing and mobile devices, peer-to-peer and vehicular networks, mobile agents and mobile agent platforms, and some existing data management approaches that have been proposed for VANETs.

In Chapter 3, we describe in detail our proposed approach to the problem of distributed processing of queries in a vehicular network. Different algorithms are presented and fully explained, depicting each phase into which the query solving process with mobile agents is divided.

In Chapter 4, we discuss several techniques that can be used by the mobile agent to reach its destination by moving (or *hopping*) from one vehicle to another. These methods use the geographic position of both the mobile agent and its destination to estimate the best way of reaching the latter, and can be enhanced if more detailed geo-spatial information or collaborating vehicles are available.

In Chapter 5, we describe the evaluation methodology used in the experiments performed along this thesis. We also describe MAVSIM, the simulator that we have developed to represent scenarios with mobile agents and vehicular networks. Our software has been extensively used to test and analyze the algorithms and methods designed in this thesis.

In Chapter 6, we evaluate our proposed approach by performing a wide variety of experiments where we measure a series of metrics, significant to quantify the performance and scalability of the proposed techniques.

In Chapter 7, we present a use case example, where a driver uses the proposed approach to find available parking places in a mixed urban/interurban scenario. Existing proposals to obtain information about available parking spaces are push-based and cannot be applied in the scenario described in that chapter. Besides, the use of mobile agents to search available on-street parking spaces is also new.

In Chapter 8, we review previous existing works related to ours, and we provide comprehensive comparisons among them.

Finally, in Chapter 9, we present our conclusions and main contributions, and we finish the document by outlining some topics for future research.

Chapter 2

Technological Context

In this chapter, we introduce some technologies upon which our data management approach has been built. Firstly, we describe the particularities of mobile computing and, especially, the topics related to wireless communications as well as the main advantages and drawbacks of such technology. Secondly, we focus on peer-to-peer mobile networks, which have a great importance in our proposed approach. Thirdly, we present some aspects of Intelligent Transportation Systems. Fourthly, we turn our attention to vehicular networks, which are the basis on which our data management approach has been built. Fifthly, we focus on software agents, and especially on mobile agents, which play a crucial role in the design and implementation of our system. Finally, we discuss some existing data query approaches for vehicular networks, that have some limitations that inspired the development of our approach.

2.1 Mobile Computing

In the last years, the demand of mobile applications has grown considerably [AFJ+06] thanks to the miniaturization of devices (which are more and more powerful and cheaper) and the deployment of global wireless communication networks. It is now possible to access an increasing number of Internet-based services and applications, anywhere and anytime, in a fast and easy way. However, the development of more complex and efficient services can be a challenging issue in such a mobile environment due to the existence of a high number of users, a variety of devices (in terms of features and computational capabilities), and mobile communication limitations.

2.1.1 Mobile Devices

Mobile devices are small computers that have many of the properties of conventional computers: they can be programmed, their users can execute different applications and software by means of a user interface, they can store data, and they can exchange data using communication devices with other computers connected to a network.

However, mobile devices have a number of special features that make them different from conventional computers: the most important one is, precisely, their mobility or portability, that allows their usage at any place and at any moment thanks to their small size and light weight. Another feature is their wireless connectivity, that makes it possible to access a high variety of services anywhere, as long as a wireless network is available. Since they are portable, they must be powered by batteries with a limited lifetime, so different energy-saving strategies must be applied in order to extend their autonomy (e.g., shutting down some components —such as the screen or the communication interfaces— when they are not used for a certain period of time).

Due to the energy and size constraints, their features in terms of processor performance, memory and storage size, are usually more limited than those of their fixed counterpart, although, thanks to the technological evolution [Fli09], today's mobile devices are more powerful than the conventional computers of a few years ago. Another reason for this is their economic cost, since building a mobile device with the same performance as a fixed device (but maintaining its small size and weight) is very expensive and commercially limited.

There exist different types of mobile devices that have several form factors and features (as shown in Figure 2.1), that are suitable for different uses. The most wide-spread ones are the following:

Figure 2.1: Different types of mobile devices

- *Smartphones*, whose main initial purpose was making and receiving voice phone calls, but that evolved into platforms capable of executing a high variety of applications. These devices are small-sized, they have user interfaces based on tactile screens, and they can communicate using cellular telephony technology (such as 3G/4G [DPS13]), as well as others such as Wi-Fi [OR03] or Bluetooth [Mor02]. They also have a geographic positioning system (such as the GPS [PS96]) that allows their users to know their current position with a high level of accuracy. Moreover, it is usual that these devices include a number of built-in sensors not related to processing or communication tasks, such as magnetic field sensors (to act as a compass), accelerometers (to read the movements of the device), microphones, video/photographic cameras, or even barometers for the atmospheric

pressure and other types of sensors. Their features in terms of CPU power and memory size are modest, and the most popular operating system they use (according to a survey from Gartner Group [Gar18]) are Android [Has09] and iOS [Sad12].

- *Tablets*, whose main feature is their large tactile screens, that allow their users to execute interactive applications more comfortably than in smartphones. Tablets usually do not have mobile phone-based communication (although they also can use them by installing a SIM card, if that feature is available in the device); instead, they use Wi-Fi as a main communication technology, and usually also include the same types of sensors as mobile phones. Their CPU and memory features are similar to those of smartphones, and they also have a GPS system. They use the same operating systems as smartphones, with minor modifications to take advantage of the big-sized screen.

- *Laptops*, that are the portable version of personal computers (i.e., desktop computers). Their CPU and memory characteristics are better than in mobile phones or tablets, but not as high as in conventional computers. They usually use Wi-Fi as their main communication technology, but they also can use wired high-speed connections (e.g., gigabit Ethernet [Eth16]) for their use as fixed computers. Their screens are not usually of the tactile type, and to interact with them it is necessary to use their built-in keyboard and mouse. Laptops usually do not have integrated GPS or 3G/4G communications, but these and others technologies can be easily added to the device thank to its hardware extensibility using, for example, USB ports [Axe15]. Their most widely-used operating systems are the same as in fixed/conventional computers, such as Microsoft Windows [Mic], Apple macOS [Appa], and GNU/Linux [Theb].

- Others. There exist other types of mobile devices, that we are not considering for this thesis due to different reasons. For example, some devices belong to the category of *wearable computers* [Bar15] such as *smartwatches* or *smartglasses*, that have promising features but their hardware is too limited and they are not very widespread yet. As another example, the personal digital assistants (PDAs [JT03]) were the predecessors of the current smartphones and had their golden age in the first decade of the 2000s, but their use declined in favor of smartphones and tablets and are no longer in the market.

Despite their lower performance, their relatively low economic cost makes them very affordable and, as a consequence, their spread has grown dramatically in the last years, being a technology that has become practically ubiquitous. According to some statistics [Sta17a], the number of mobile phone users worldwide in 2017 is about 4.77 billion and is estimated to increase to 5.07 billion in 2019. Similarly, the number of smartphone users worldwide is about 2.32 billion in 2017, and is expected to increase to 2.71 billion in 2019 [Sta17b]. By the year 2023, 95% of all mobile data traffic will come from smartphones [Eri17], and in 2021 there will be more people

with smartphones (5.5 billion) than people with access to clean drinking water (5.3 billion) [Cis17].

2.1.2 Mobile Communications

Given the mobile nature of mobile devices, the use of wireless radio communications is a logical consequence, since using fixed wired network connections would have little sense, except for certain periods of time when the mobile device can stay in the same place for a relatively long time (e.g., when their batteries are being recharged, etc.). For this reason, there is a wide range of communication technologies that mobile devices can use, although all of them have some drawbacks.

One type of mobile communication technologies are those based on cellular telephony, such as GPRS [LT03], 3G/4G [DPS13] or the future 5G [HH15], that belong to the category of Wide Area Networks (WAN) since they cover large geographic areas. In these technologies, the devices (i.e., mobile phones) establish a link with one of the base stations located in their vicinity to make phone calls or send/receive data. These base stations are connected to each other with high-speed wired connections and, in this way, they can transfer data or make phone calls from one user to any other in a distant place or access the Internet (see Figure 2.2). The base stations have an antenna and are usually located at high places, in order to communicate with the user terminals with the least amount of possible obstacles.

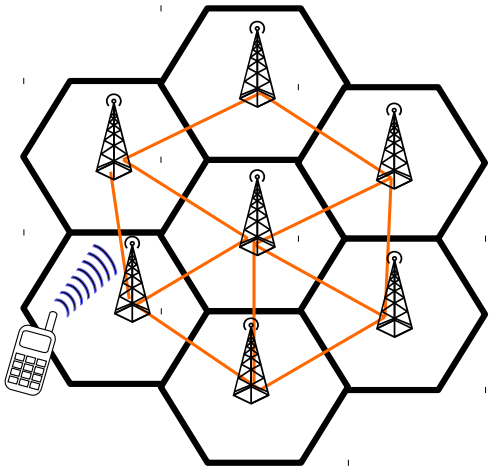


Figure 2.2: Example of a cellular network

The antennas cover their surrounding area and have a range of about 0.8 to 8 km, although there can be certain places where the radio signal is strongly attenuated (e.g., underground places or tunnels) and therefore the communication between the terminal and the antenna becomes impossible. Due to the limited range of a single antenna, it is necessary to place many of them at different places, each one covering

its surrounding area, which is called a *cell*. In each cell, the antenna covering it uses different ranges of frequencies, in such a way that two adjacent cells do not use the same frequencies in order to avoid interferences and collisions. When a user terminal moves from one cell to another, it may establish a connection simultaneously with each cell's antennas so that if a communication or phone call is in course, it will not get interrupted when the terminal leaves one cell and loses the connection with the base station that belonged to the previous cell (this is called a "soft handoff").

Cellular telephony networks usually offer a good coverage to their users, especially in urban areas, which allows to use them almost anywhere and at any time. The drawbacks are that they require the deployment of a costly infrastructure to operate (e.g., the installation of a large amount of base stations), and their bandwidth and latency to transfer data are modest, although they are improving as technology advances. Another drawback is that, due to their high installation costs, their users are required to have a contract with a mobile telephony operator and pay a fee in exchange of their services.

Another type of wireless communication is based on the IEEE 802.11 [IEE16] standards (in the last years especially the 802.11g, 802.11n and 802.11ac specifications), also known as Wireless-Fidelity or Wi-Fi, that belong to the category of Wireless Local Area Network (WLAN), since their typical use takes place at home, universities, office buildings, etc. These standards are the most widespread today and they are present as the communication interface by default in mobile phones, tablets, laptop computers, electronic books, surveillance cameras, wearable computers, and many other devices and *gadgets*.

This type of wireless network has higher data transmission rates than cellular telephony-based networks (as shown in Table 2.1), but they are constrained by a limited communication range, of about 200 to 300 meters with no obstacles. In their usual usage scenario, the individual devices connect to each other through one or more intermediate devices called *access points*, that act in a way similar to the base stations of cellular networks, extending the coverage of the radio signal and allowing different devices to exchange data. However, unlike in cellular networks, the users of Wi-Fi technology do not have to pay a fee to a telecommunication operator and can install and create their own wireless networks, without needing any further authorization, and at an affordable cost. Moreover, since the introduction of Wi-Fi Direct [CMGSS13], it is possible to establish direct device-to-device communications without needing the use of an access point. However, the communications performed in this way are limited to the range of the involved devices, and it is not possible to extend it wider.

We must also mention another variant of these standards, the 802.11p or Wireless Access in Vehicular Environment (WAVE) [IEE10], that was created for its specific use in vehicular networks. It is designed to operate in scenarios where network connections must be established as soon as possible, as it happens between high-speed moving vehicles. To accomplish this, it operates in a dedicated frequency range and some of its synchronization and negotiation protocols are simplified so that they

take less time to complete, which allows more efficient communications of the types *vehicle-to-vehicle (V2V)* [HPY+14] and *vehicle-to-infrastructure (V2I)* [UJ16]. This technology is currently not present in consumer mobile devices, and only some car-related manufacturers offer solutions based on this standard [Ara, Red].

Finally, we will also refer to mobile communications based on the Bluetooth standard [BS], that belong to the category of wireless Personal Area Networks (PAN). This technology provides low bandwidth communications with a very short range (usually about 10 meters), and consequently, its energy consumption is also very low, which is a good feature to extend the batteries' autonomy in mobile devices. This technology does not need the use of intermediate access points and operates using direct connections from one device to another. Before establishing a connection, the devices must *pair* to each other by exchanging a key and, after that, they can continue with the data exchange. It is present in many mobile devices (mobile phones, tablets, laptops) and its primary use is for the direct transference of small files from one device to another, or for the wireless connection of these devices with other small accessories such as keyboards, mice, headsets, speakers, etc.

A summary of the different features of some of the most widespread wireless communication technologies are shown in Table 2.1.

Technology	**Category**	**Max. bandwidth** Down/Up (Mbit/s)	**Max. range** (meters)	**Spectrum type**
GPRS	WAN	0.0856/0.0428	Global	Licensed
3G	WAN	14.4/5.76	Global	Licensed
4G	WAN	150/75	Global	Licensed
802.11g (Wi-Fi)	WLAN	54/54	250	Unlicensed
802.11n (Wi-Fi)	WLAN	600/600	300	Unlicensed
802.11ac (Wi-Fi)	WLAN	1300/1300	500	Unlicensed
802.11p (WAVE)	WLAN	54/54	1000	Unlicensed
Bluetooth	PAN	24/24	10	Unlicensed

Table 2.1: Summary of wireless communication technologies

2.1.3 Mobile Applications

Mobile environments have a number of limitations (e.g., the limited bandwidth and connectivity) and unstable conditions (e.g., the presence or absence of communications, the changing position of the user) that make the development of mobile applications a process more complex than with conventional applications.

An important aspect that we must highlight is the heterogeneity and limited capabilities of mobile devices. Thus, there exist different types of devices (smartphones, tablets or laptops), which run applications in varied environments and operating systems. These devices have different processing, storage, and user interface capabilities,

and due to this heterogeneity in hardware configurations, architectures, and operating systems, the development of software and applications may become a difficult task: one program or application developed specifically for a certain device will likely not work in another different device, unless it is properly adapted, which can be a costly process. Thus, to overcome this issue, there exist some approaches. One option is to develop different versions of the applications, and maintain one version for each type of device. This is the best option in terms of performance, since each version can be optimized for the specific hardware where it will execute, but maintaining a high number of different versions can have an excessive cost. As an alternative, applications can be developed using some programming framework, such as Apache Cordova [Thea] or React Native [Facb], that perform automatically the required modifications so that the application can execute in any supported device. In this way, the underlying platform is abstracted and the programmer only needs to code one single version of its application. The drawback is that the software developed in this way has less performance than if it had been developed natively, since the generated code will probably not be optimized for any specific hardware where it will execute; instead, it will use generic code in order to be functional in as many devices as possible. Finally, there also exist hybrid applications that, as an intermediate alternative, use both native components provided by the device's operating system as well as code generated in some standard interpreted language (such as JavaScript or HTML5), that balance the best features of both options.

Despite these difficulties, the use of mobile applications has increased in the last years, and the two biggest mobile application stores (Google Play and Apple App Store) contained, respectively, 2.8 and 2.2 millions of applications as of March 2017 [Sta18]. These figures have been reached thanks to a number of factors, such as: the lightweight and small size of the mobile devices, that allows carrying them everywhere; the increase of computer power and storage in mobile devices, that allow the creation of more complex applications; the wide adoption of tactile interfaces that eases the use of applications by their users; the immediacy in access to information, thanks to the advances in mobile communications that have increased their speed and are present almost anywhere; and the relatively low cost of mobile devices, that has led to their massive adoption by non-technical users, and thus the creation of a software industry around them.

These features have made possible the creation of entirely new types of applications that take advantage of features not present in the traditional *fixed* computers, such as the location of the user, or the status of some of the different types of sensors that mobile devices may have.

One type of these applications are *context-aware* applications [YS00, FYN04, MV11, IHTLdCRH15], where the *context* of the users are taken into account for providing them different services or information. The context can be given by the status of many parameters from the user himself/herself, from his/her surroundings or his/her environment, etc. For example, some parameters that can be used to determine the context of a user are the current time and day, the geographic position, the temperature, the noise level, the traveling speed (if the user is aboard a vehicle),

the heading, the battery level of his/her mobile phone, the presence or absence of certain Wi-Fi network, etc. Many of these parameters can be obtained by sensors integrated in mobile devices (e.g., mobile phones). Alternatively, some sensors can also be standalone devices that work autonomously and can be queried by other devices using, for example, wireless communications using Wi-Fi or Bluetooth. An example of these applications is Automate [Lla], where the user can program his/her mobile device to perform some actions when one or more conditions are met. For example, if it is Sunday morning and the device is recharging its batteries, put the ring volume to the minimum and send an SMS message with the text "do not disturb" to the phone number from any incoming call. Another example is IFTTT (If This Then That) [Ova14, IFT], that is a web service to perform automatic actions according to the user's actions in other on-line services (e.g., if I save a picture in my Dropbox folder, then publish it in my Instagram account). As another example, we can cite *virtual assistants*, like Siri [Appb] or Google Assistant [Goo] that take voice commands or requests from their users and provide them an answer by performing some actions or giving useful information. The user's request can be not entirely precise, and the application must deduce the missing information according to the context. For example, the user could say "I want information about the weather", and the assistant will have to infer that the information requested by the user is about the weather in his/her current location for the next hours or days.

When the context refers specifically to the geographic location of a user, we can talk about *Location-Based Services* (LBS) [Küp05, DHdLD10, IMS+11]. The location of the device, that can be known using the mobile device's built-in GPS receiver, or the triangulation of Wi-Fi stations or cell towers positions, is used to provide a service or relevant information according to both the position of the owner as well as possibly of other users or services that are located in the vicinity. One obvious example is navigation software, that allows its users to find the shortest path from their current location to any other place, and guides them through the computed path, according to the updated position of their users, and that have the ability to computing alternate paths if, for example, its users make a mistake and take a wrong direction. As another example, some social networking applications [Faca] allow their users to know the updated position of their *friends* (as long as they agree to share their positions with a certain precision), or display a notification if some of them are nearby. Similarly, the aforementioned virtual assistants can use their position to give a location-dependent answer to some types of requests related to spatial locations. For example, if the user says "I would like to eat sushi", the application will return a list of Japanese restaurants near the current position of the user.

As a summary, we can conclude that mobile applications have grown thanks to the great utility they have to their users, that can obtain different information and services in a fast an easy way. This immediacy is possible thanks to a number of factors, such as the mobile communication features in terms of bandwidth and coverage, and the use of sensors integrated in mobile devices, that help to determine more easily the user context and, in this way, develop smarter applications.

2.2 Mobile P2P Networks

P2P networks have a number of advantages over traditional centralized solutions for their use as a means to exchange data or other resources, and the same principles can also be used to establish network communications among many devices. In this section, we explain the principles of P2P networks and we explain how the can be applied to mobile networks.

2.2.1 Generic P2P Networks

A peer-to-peer (or P2P) network [Ora01] is the connection of many computers or devices by means of networks connections in such a way that, when it comes to the exchange of data or other resources, all the members of this network have the same category and they do not follow a hierarchical structure.

For example, in the traditional client/server paradigm, one computer acts as a server, while many other computers or devices are the clients, that request some resource to the server, using a network channel to reach it, as shown in Figure 2.3. In this case, the server has a higher hierarchical level: the different clients depend on its presence to access or exchange data, and the server could *decide* if such data are served or not to certain clients.

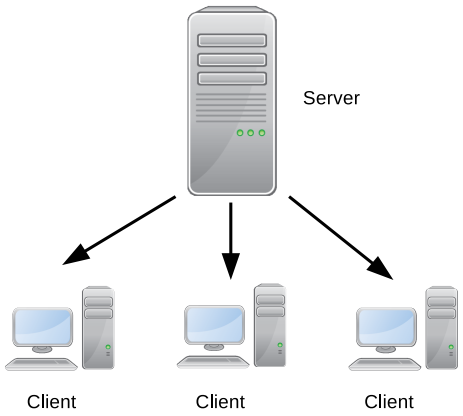


Figure 2.3: A client/server data exchange approach

On the other hand, in a P2P network, there is no hierarchical structure and all its participants have the same category. In these networks, every member, or *peer*, can have at any moment the role of a client, a server, or both, and send or receive data or other resources indistinctly, as shown in Figure 2.4. In this case, the absence of a hierarchy means that one member does not depend on a single one to access or exchange data but, instead, it can obtain those data directly from any other member.

This type of decentralized organization has a number of advantages [HAY+05] that makes it an adequate option for building distributed systems: the bandwidth and

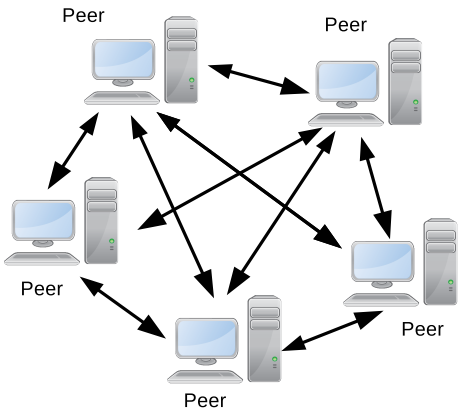


Figure 2.4: A P2P data exchange approach

other computation resources are used more efficiently, since they are not concentrated in a single node, as it occurs with the client-server approach; the reliability is higher, since if one or more peers fall or get disconnected, the remaining peers can keep exchanging data or resources; and the scalability of the whole system is better, since if it were necessary to increase its resources, it would only be necessary to add new peers to participate in the network. However, P2P networks also have a number of problems such as those related to security and trust (since there does not exist a central authority), the discovery of other nodes when there does not exist a centralized list of them, or the difficulty to find specific data and route them through the nodes, which requires complex algorithms that are not necessary in simpler client-server approaches.

There exist different types of P2P networks, according to their degree of decentralization [PBV05]:

- *Centralized.* There exists a central node with an index of the data resources of the network. When a user searches some resource, the search is performed in the central node using that index, that also contains information about the nodes that have the resource. Then, the client user can obtain the resource directly from any of those nodes.

- *Purely decentralized.* There does not exist any central node, and every peer maintains an index of its own shared resources. When a user searches a resource, the search is broadcasted to the other peers, that search into their own indexes and resend again the search if it is necessary.

- *Hybrid.* In these networks, a relatively small number of nodes become *supernodes*, and they maintain the index of the shared resources, synchronizing them periodically in case some of them fail. When a user searches a resource, the search is performed in any of those supernodes, and the resource can be obtained from any node.

Due to the aforementioned advantages, P2P networks are currently used in several applications. One of the most popular applications is the exchange of files, where users can share files (e.g., music, videos, images) with others, and in the same way they can also download them. Some examples of these applications are BitTorrent [Bit] and eMule [eMu]. Another example of P2P use is for streaming multimedia content, such as Peer5 [Pee] and Play2Live [Gam18], where the bandwidth usage is shared among their users. The I2P project [I2P] is an anonymous overlay network (a network within a network), that is intended to protect communications from surveillance and monitoring by third parties such as Internet Server Providers (ISPs). I2P is intended to be used by many people who care about their privacy: activists, oppressed people, journalists and whistleblowers, as well as the average person. Another example of P2P application corresponds to the existence of web search engines that perform their task in a decentralized way, such as FAROO [FAR] or MINERVA [BMT+05]. As a last example of P2P applications, Bitcoin [Nak08] and other alternatives such as Ethereum [Woo14] or Peercoin [KN12] are P2P-based and decentralized digital cryptocurrencies.

2.2.2 Mobile Ad hoc Networks

When it comes to the way of connecting several mobile devices among them to exchange data, some difficulties may appear. If all the users are using 3G/4G based communications, they must pay a fee to their mobile phone operators, and they also must be under the coverage of one or more cellphone towers, which can be problematic in certain places such as underground areas, tunnels, inside buildings built massively with concrete and/or metallic materials [MDS+14], desert or rural areas with a very low density of cell towers or, on the contrary, areas that are suddenly crowded with hundreds or thousands of people who can saturate the maximum service capacity (for example, in sport events or concerts). If, on the other hand, the users are using their Wi-Fi devices instead, it is still necessary to be under the coverage of some access point or wireless router, that must be installed and managed by a third party, not to mention other security issues that may arise if they are using an unencrypted channel or if the owner of the infrastructure is not trustful [KOB+08].

Additionally, in the case of mobile networks, there are some other issues that make the usage of communications even more difficult. For example, while in wired connections the established links can be quite stable (i.e., once they are started they will not be interrupted unless one of the peers stops it), this may not occur in a mobile network, where the wireless communications can end unexpectedly if one or more peers are moving and get out of the communication range of the wireless router or access point.

One way to solve several of these problems is by the use of Mobile Ad Hoc Networks, or MANETs [CCL03, OHY13, CG14]. In this type of wireless networks, their nodes can move across a certain geographic area and, in order to exchange data, they create ad hoc links. These connections are established directly between two or more nodes (in a peer-to-peer way) that must be within each other's communication range.

Thus, they do not need any additional infrastructure such as access points or cell towers, and can freely establish a link at any moment to any other nearby device.

One issue to consider in this type of networks is that the data can only be directly transmitted from one node to another if both of them are within their communication range, or at a distance of one hop. However, it is also possible to send data to other nodes whose location is at more than one hop of distance by using other nodes as intermediate relays (through multi-hop routing). These nodes, once they receive a packet of data, immediately forward it to the next node and, by repeating this process the required number of times, the data can reach their destination node. Routing data in MANETs is a problem that has attracted the attention of researches and many routing algorithms and protocols have been proposed and evaluated [CM99, TNCS02, GSB02, DPH05, RN15, BMW17].

In order to take advantage of the properties of MANETs, and to ease the development of applications that use this type of networks, some frameworks have appeared, such as Peer2Me [WBS07] and Proem [Kor02]. As another example, the platform iTrust [LMMSC14] is a P2P retrieval system over Wi-Fi direct, and there also exist proposals for streaming audio/video [IBHD16], and content/file sharing [SAQM17] in MANETs.

One special type of MANET are the Vehicular Ad hoc Networks, or VANETs, which is the focus of this thesis. In these networks, the nodes are the vehicles that travel along the roads or streets of a city or a wider area. The special properties that characterize VANETs are that their nodes (the vehicles) usually move at a high speed, they follow constrained paths (i.e., the roads or streets) that they usually cannot leave, and therefore they cannot change abruptly the direction they are following. The communications in a VANET can be interrupted more frequently than in a generic MANET, due to the high mobility of its nodes and the changing conditions of the road. For example, the number of vehicles to which a link could be established would increase when approaching a city, and decrease when moving away. The communication links can be established not only among its vehicles, but also with other types of nodes, such as roadside units (e.g. traffic lights, or surveillance cameras) that are located along the roads at fixed places. Another characteristic is that, in urban environments, the wireless communication signal can also be constrained to the street layout, since it likely cannot propagate through buildings or other obstacles that separate the streets.

Due to these special features, some of the issues present in a generic MANET require another approach in order to tackle them [SKRM11, RA11, SK14]. For example, for the problem of routing data, a number of algorithms similar (but not equal) to those of MANETs have also been proposed [LW07, FRSS13, VCML13].

2.3 Intelligent Transportation Systems

Since the motor vehicles became popular thanks to their serial manufacturing in the 1920s, by Henry Ford, the innovations applied to them have led to vehicles more and more efficient in terms of reliability, efficiency, security and, more lately, respect for

the environment [TC14]. However, despite the progressive introduction of electronic and computer technologies in the vehicles, their operation and interaction with other elements of the driving environment (such as traffic signals, traffic lights, other vehicles and drivers, etc.) are still performed by humans.

Only very recently, autonomous vehicles [FK15, Lit17], that are able to drive without the intervention of humans, have started to take center stage, since the computer power needed to analyze in real time a such a changing environment has not been possible in vehicles until recent years.

However, the autonomous vehicle is only one of the many achievements in the process of automatizing all the tasks related to driving, with the objective of making this process unattended for humans, as well as more efficient in terms of travel time, comfort, security and energy usage. This goal has not been fully achieved yet, and a number of intermediate steps are being taken, evolving towards the aforementioned final goal, and whose increasing innovations belong to the field of Intelligent Transportation Systems (ITS) [DD10, ZWW+11, KGM14, AFF16, SP16].

2.3.1 Intelligent Transportation Technologies

Intelligent transportation systems use different information and communication technologies, so that their different actors (i.e., the drivers, the vehicles, the roads and their signals, etc.), can exchange information among them regarding their status. Once the received information is processed, it may lead to perform some action that, as a result, will improve some aspect of the driving process (e.g., its security, comfort, or energy usage). Some of these technologies are the following:

- *Computational technologies.* Thanks to the advances in the miniaturization of electronics and their lower costs, computers are being introduced in all the scopes. In the case of ITS, they are present as on-board units in the vehicles and also in the roadside infrastructure, among other things to process data from a high number of sensors and to take decisions accordingly. Moreover, the application of artificial intelligence techniques to ITS will have a big impact in the next years [DLBB+15, GBD15, LDK+15].

- *Wireless communications*, to allow the data exchange between vehicles, or between vehicles and roadside units (e.g. traffic lights, or surveillance cameras). For short-range communications, protocols such as 802.11p [IEE10] or the Dedicated Short Range Communications standard (DSRC) [JTM+06] have been proposed, and their use is oriented to communicate with nearby vehicles and the infrastructure (e.g., the toll stations in highways). On the other hand, for long-range communications the proposed protocols are WiMax [AGM07] and also those based in cellular telephony, such as 3G/4G [DPS13]. Their intended use is for vehicle safety and information, and to provide entertainment to the passengers.

- *Floating car data*, that is a method to know the speed of the traffic flow in a road or street by using different procedures. One of these methods is by

using the cellular signals from the mobile phones of the drivers or passengers traveling in the vehicles. A similar method uses the Bluetooth [BS] identification broadcasted, for example, by hands-free devices used by drivers. These methods have the advantage of being passive and not needing the installation of extra devices in the vehicles. On the other hand, there are other systems that require the active collaboration of the drivers, for example, by carrying a GPS [PS96] transponder in the vehicle, or by installing a tracking application in their smartphones [VVV+12].

- *Sensors.* These devices are necessary to obtain information from the road or the vehicle's condition, and therefore they can be installed in any of those places. The information that they acquire is transmitted to a computer in order to make proper decisions. There exist many types of sensors (to measure different parameters) [STYW04, TZQS09], such as: Inductive loop detectors, that are placed under the pavement and are activated by the presence of big amounts of ferromagnetic materials, such as vehicles; pressure pads, that are activated by the weight of vehicles or pedestrians; radars, that use the Doppler effect of electromagnetic waves to measure the speed of vehicles; video vehicle detectors, that use video cameras and artificial vision techniques to read the vehicles' plates.

The joint operation of all of these elements make Intelligent Transportation Systems a reality. These technologies have been operating for some years, and researchers are developing several more [CYS15, HYY+15, GIZCC15] that will bring new possibilities to ITS in the future.

2.3.2 Intelligent Transportation Applications

There exist a number of applications for Intelligent Transportation Systems that are currently being used, both by drivers and by traffic authorities. These applications use the technological elements described before, and they can benefit from the exchange of data between the traveling vehicles, or between the vehicles and the roadside infrastructure. Some of the most popular and important applications are the following:

- The *accounting of the traffic* that traverse a certain street or road, using different devices such as magnetic sensors or inductive-loop detectors. Some of these devices, when used in conjunction with video cameras and artificial vision techniques can even discriminate the type of vehicles (cars, trucks, buses, etc.) that are moving through the road, which can be interesting for the management of traffic by local authorities.

- The *automatic regulation of traffic lights* in a city, adjusting the times of every color depending on the traffic conditions, which can be obtained by using sensors or devices such as the ones mentioned above. Thanks to this information, the regulation can be optimized to allow a maximum traffic flow speed with the minimal waiting time for the drivers and minimum probability of traffic jams.

- The *notification of interesting information to drivers*, by means of variable-message signs located in frequently-traveled roads. These electronic banners can offer information of interest that can be obtained both automatically (for example, related to the weather or the density of traffic) or manually by the traffic operators of the system (e.g., the presence of works or accidents in some point of the road).

- The *automatic allocation of bidirectional lanes* in motorways to be used in one direction or the other, according to the traffic density in certain hours and the need to increase the maximum capacity of the path from the city center to the suburbs or vice-versa.

- *Electronic toll collection*, in high-ways and other toll roads such as tunnels or bridges. The vehicles carry a transponder that is activated by an antenna located on a toll lane. The antenna identifies the transponder, which is associated to the user's bank account, and the toll fee is deducted automatically every time the vehicle traverses the toll lane without the need to stop. Some examples of this service are the E-ZPass [E-Z] in the United States and the VIA-T [Aso] in Spain.

- *Emergency vehicle notification systems*, that calls the emergency services when an accident occurs. The call can be activated automatically by sensors aboard the vehicle, or manually by any of its occupants. When the emergency service is contacted, a voice communication is established with an operator while, at the same time, a minimal amount of relevant data (e.g., the vehicle position) is also sent. In the European Union this service is known as *eCall*, and it was made mandatory in all new cars sold in that region after March 31st 2018 [Eur15, The14].

- *Applications based on VANETs*, that use data exchanged by vehicles in a P2P way for different types of applications, such as those related to security, entertainment, communication, social aspects, etc. These applications and several more are explained in Section 2.4.

The development of Intelligent Transportation Systems has innumerable benefits, such as time savings for drivers and passengers, better emergency response times and services, the reduction of crashes and fatalities, decreasing the probability of traffic congestions, more energy efficiency, environmental benefits, etc. Moreover, its development has become a serious topic and the academia, the industry and the political authorities, are devoting a lot of effort and money in its promotion, by researching new ways and systems for improving the ITS, launching new products that implement those improvements, and by elaborating laws to enforce them.

2.4 Vehicular Networks

A *vehicular ad hoc network* (*VANET*) [BEH04, HL08, OW09, KAE+11] is a highly mobile network whose nodes are vehicles traveling along roads or highways. They can establish connections with other nearby vehicles and, in this way, they can exchange different types of information. This makes the development of new applications interesting for drivers (as well as for their passengers) possible, such as applications related to security, monitoring, entertainment, or data sharing. It also enables the participation of VANETs and their vehicles in Intelligent Transportation Systems (see Section 2.3) and, in this way, a more efficient, secure and environmentally-friendly transportation can be achieved.

2.4.1 Features of a VANET

One of the main features of a VANET is that its nodes are vehicles that are constantly moving while they perform the typical activity of a network node (i.e., sending and receiving data to/from other nodes by using a communication channel). The vehicles are equipped with short-range wireless communication devices (such as WAVE [USAM09], Wi-Fi [IEE16] or UWB [OHI05]) and can establish connections with other nearby vehicles (V2V) or with road-side infrastructure (V2I) in a peer-to-peer way.

However, such communications usually have a short duration (a few seconds), due to the potentially-high speed of the vehicles and the short range of these communication devices, that in practice is approximately around 100 to 250 meters. Moreover, if two vehicles are approaching at high speed in opposite directions (e.g., in a highway) the time window available for communication can be reduced considerably.

Another problem, that comes from the short-range communications, is that it is not possible for a vehicle to directly send data to another one that is located farther than such range, and therefore it is necessary to use multi-hop algorithms, that can be quite complex. The reason is that the network topology is constantly changing due to the vehicles' movements, and therefore, in this scenario, locating the destination vehicle and routing the data through the network can be challenging.

Since a VANET is a type of MANET (see Section 2.2.2), many of its routing and multi-hop algorithms could be considered for those purposes. However, this may not be always efficient or even possible, since VANETs have a number of singular features that makes them different from generic MANETs:

- *High mobility.* In a VANET, the vehicles are constantly moving (usually at high speeds) when they travel through roads or streets.

- *Unstable connectivity.* Due to the high mobility of the vehicles and the short-range wireless communications, once a link between two or more vehicles is established, it will likely last only for a brief lapse of time. When one of the participating vehicles moves out of the communication range, the link will break and any communication operation in progress will fail.

- *Variable density of vehicles.* The ad-hoc connections need the presence of other vehicles in the surroundings. The higher the number of vehicles present, the more likely an ad-hoc connection can be established. Therefore, with higher traffic densities these connections will occur more frequently than with lower densities. However, the density of vehicles can change depending on many factors, such as the type of area (e.g., rural or urban), the time of the day (e.g., peak or off-peak periods), or other events that may affect traffic (e.g., concerts or sport matches).

- *Restricted mobility.* The vehicles can only move through defined paths (such as urban roads and highways), whose layout is known and does not change very frequently along time. Besides, they have additional restrictions, due to the existing speed limit regulations and other traffic rules (e.g., the number of lanes, or whether a road is bidirectional or one-way).

- *No energy constraints.* Since the vehicle's engine can provide electricity as it works, the communication and computer devices can obtain power without depending on batteries with a limited lifetime.

- *Better computing capabilities.* Thanks to the energy supplied by the vehicle, and to its size and load capacity, the devices carried in a vehicle do not need to be too much lighter or smaller, as it occurs with mobile devices such as smartphones. Therefore, the vehicles can use conventional electronic components, which can be cheaper and more powerful than those used in mobile devices.

Some of these features, that are not present in generic MANETs, should be considered in the development of applications or other types of algorithms for VANETs. For example, the restricted mobility of the vehicles can help to predict their future positions, and this could be exploited by routing algorithms, such as [ZC08].

2.4.2 Vehicles in VANETs

Besides the communication equipment, vehicles that participate in a VANET have an additional number of devices that enable them not only to send and receive data, but also to acquire, store and process them to a certain extent. Some of these devices are the following:

- A *computer* or a device with similar capabilities. It processes all the data received from the other devices present in the vehicle, and it also executes the algorithms or programs that, according to those data, can take actions if it is necessary (e.g., warn the driver about a road event or send certain data to another vehicle). This computer can also have a storage system (e.g., a hard disk) to save all those data. In this thesis, we abstract ourselves from the specific technology used in the vehicles to store data (e.g., relational or NOSQL databases, plain files, etc.).

- *Communication devices.* They allow the vehicle to establish communications and exchange data with other entities (i.e., other vehicles, roadside units, or people) using wireless signals, either based on a centralized infrastructure (e.g., a mobile telephony network) or a decentralized solution using direct ad hoc P2P communications. Both options have a number of advantages and drawbacks and are discussed in Section 2.4.3.

- A *geographic positioning system.* This device allows the vehicle to obtain its geographical position with a high level of accuracy by using a network of satellites. The most used is the Global Positioning System (GPS) [PS96], but there are other systems such as GLONASS [Rev12] or GALILEO [BDG+00, NLS+15]. All of them provide an accuracy of a few meters (typically around five meters), or even less, if several of these systems are used combined at the same time. There also exist other complementary methods to enhance the location precision, such as the *differential GPS* [ME06] or the *assisted GPS* [VD09].

- *Digital sensors.* These devices measure a physical phenomenon, and those measures are converted to a digital format, so they can be processed by computers. Regarding vehicles, we can distinguish two types of sensors. On the one hand, embedded sensors that measure parameters related to the operation of the vehicle itself and its mechanical components (e.g., the engine's temperature, the level of fuel, the pressure of tires), which can be very numerous (more than 100 in luxury cars [Fle13]). On the other hand, there may also be additional sensors that measure parameters that are external to the vehicle operation. For example, emerging autonomous vehicles [BMS00, FK15] have ultrasonic [PUV95], LIDAR [LAB+11] and even RADAR [CDW98] sensors to obtain precise information about nearby obstacles in their surroundings.

- *User interface.* This element allows the human driver and the vehicle to communicate with each other by using an interface that, ideally, does not distract the driver from paying attention to the road [Cel01]. Therefore, as an input method (from the human user to the vehicle computer), the typical interfaces are based on tactile screens, simple buttons near the steering wheel, or voice commands. On the other side, the communication methods from the vehicle's computer to the human driver can use different sound signals or color lights, large screens where easy-to-read texts or icons are displayed, or synthetic voice locutions. A typical example of the latest are navigation systems, that guide the driver with voice instructions about the next turn that he/she must take.

The exchange of data among vehicles participating in a VANET can enable the development of several interesting applications for drivers (and even passengers). The most important element is a computing device that, while it processes the data and executes applications, also acts as a coordinator of all the rest of devices in the vehicle, as shown in Figure 2.5.

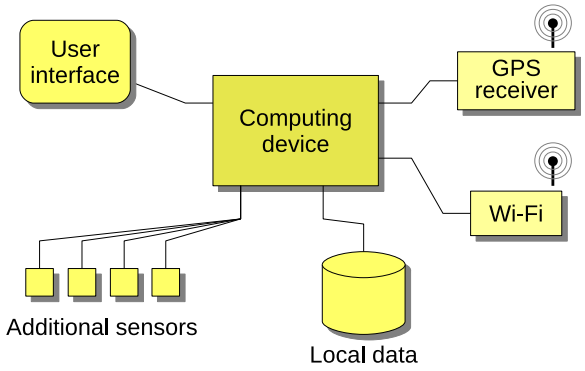


Figure 2.5: Different elements of a vehicle in a VANET

2.4.3 Communications in VANETs

Communications in a VANET are performed using direct ad-hoc connections from vehicle to vehicle (or from vehicle to roadside units), using short-range wireless communication devices. Using this type of network has a number of advantages, such as [IDTL15, ZJ18]:

1. The users do not need to pay for the use of these networks, since they operate in an unlicensed frequency spectrum.

2. There is no need of a dedicated centralized support infrastructure (expensive to deploy and maintain), such as it occurs with mobile telephony networks or WLANs (e.g., enterprise networks), that also need an infrastructure installed by a third party, with access points, routers, authentication services, etc. Instead, the users can establish direct connections from one device to another at any moment, provided that they are within their communication range.

3. It allows a very quick and direct (i.e., without intermediate proxies or routers) exchange of information between two vehicles that are within range of each other, which may be critical for safety applications for vehicular networks.

4. Many application scenarios do not need to communicate with a specific target vehicle but with all the vehicles within a certain area, and therefore broadcast (the typical communication mechanism in VANETs) is a suitable choice.

5. They are naturally distributed and scalable, due to their decentralized structure. The processing power and the bandwidth are divided among all the nodes, instead of being centralized in a single node. In a centralized solution, the network could only grow to a certain extent, since the resources of that single node would be finite.

6. It favors privacy, since there are no central entities that may know the position of all the vehicles in the network over time. Instead, every vehicle can only know

the position of its neighbors, and only while they are within its communication range. The only concern could be that, when the nodes (i.e., vehicles) forward data received from other nodes, they could inspect their content unless those data are encrypted.

Other communication schemes can also be considered, based on a fixed infrastructure or mobile telephony networks (e.g., 3G/4G[DPS13]). Thus, even if it may be unrealistic to assume the availability of a generalized wide-area fixed infrastructure in the next years, mobile telephony networks already offer new perspectives for the development of applications to assist drivers. Anyway, such solutions, based on a centralization of the data and decision processes, still suffer from issues such as poor scalability or low reaction time available when dealing with some events like an emergency braking. So, although it is important not to rely on a fixed network infrastructure, which can be difficult and expensive to deploy at a large scale and with global availability, some roads could also offer some static relaying devices which provide Internet access to nearby vehicles by using a fixed network (enabling V2I communications [UJ16]). According to [CMSM+18], combining V2V communications and communications based on the use of a cellular network may be important to reduce the communication costs and to improve the timeliness of ITS data.

2.4.4 Applications of VANETs

According to ABI Research, more than 500 million connected cars will have shipped by 2022 [All17]. Therefore, VANETs open up a wide range of opportunities to develop interesting systems for drivers. Although safety applications are usually emphasized, there are also interesting applications related to comfort, entertainment, and travel efficiency [IDTL15, ECVL15]. For example, traffic information systems [SB11], congestion assistance [SvEK+10], vehicular platoons [JLW+16] (where a group of vehicles follow and replicate the movements of a leader vehicle, by queuing after it at a short distance, thus reducing the road congestion), post-collision assistance in the case of accidents [FGM+14], location-based message boards [WER+05], vehicular social networks [MDNB14], content distribution/sharing [MCCF14], drive-thru Internet access [CLZ+14] (opportunistic content-delivery from Wi-Fi access points), file sharing [SDKM14], transmission of multimedia data [FCM+14] (e.g., to provide live videos of traffic jams or emergency situations, to enable intervehicle video conversations, video-on-demand), monitoring and surveillance [UIDM09], advertising [LLC13], virtual flea markets [LLPG10a], or even games played by occupants of different vehicles [PRF10]. Several of these applications would benefit from a platform that enables the exchange of information about vehicles; as an example, the VESPA project [DI11, DI13] is a system for vehicles to share information: they can process and disseminate any type of event (e.g., information about available parking spaces, accidents, an emergency braking, information relative to the coordination of vehicles in emergency situations, etc.) to potentially interested vehicles and they can evaluate the relevance of the event data received in order to determine, for instance, whether the driver should be warned or not.

Another interesting application for VANETs is their use in mobile sensor networks. In a classic sensor network [ASSC02, YMG08], there exist many sensing devices located at fixed places to measure some parameter (e.g., the temperature or polluting gases in a city), and the acquired data are sent to a monitoring center (e.g., a research facility or a city council) for their analysis or exploitation. However, these types of sensing networks have some drawbacks. Firstly, they can obtain data only from the place where the sensors were deployed in the first place; if it is necessary to obtain data from a different place, they would have to be re-deployed. Secondly, they can only measure a limited type of parameters, according to the type of sensors installed. Finally, they can have high costs of maintenance if some sensors have a malfunction or they run out of battery, and reaching remote or inaccessible places may be necessary in order to replace them.

To overcome these problems, the vehicles in a VANET can be used as moving sensors [LMZ+06, HBZ+06, UIDM09], since they can move to any place to measure the designated parameter, and they can transport any other sensor type if it is necessary to measure some new parameter. Moreover, this sensing activity can be performed in either an explicit or an opportunistic way, depending on whether conventional vehicles are used and whether the drivers are willing to deviate or not from their original routes. Thus, performing the sensing activity in any of these ways can enable *collaborating sensing* [IWD14] for the monitoring of city areas and others. Thanks to the VANET, the acquired data could be immediately sent to the monitoring center by forwarding them through the vehicles using a multi-hop protocol in a quick way and without any additional economic cost.

2.5 Agent Technology

An agent [MDW99] could be defined as an entity whose goal is to perform some action on behalf of somebody. When these agents belong to the context of information technologies, they are called *software agents*, since these entities are usually programs that follow a certain behavior on behalf of a user in an autonomous way. As they were defined by Wooldridge [Woo09], a software agent is an "encapsulated computer system, situated in some environment, and capable of flexible autonomous action in that environment in order to meet its design objectives". Agents have a number of interesting features [Nwa96], that makes them a suitable election for the development of distributed systems.

In the rest of this section, we first discuss the properties of agents. Then, we focus on the specific case of mobile agents, which is a key technology in this thesis. Finally, we describe some existing agent platforms.

2.5.1 Software Agents

Software agents, which we will refer to in the rest of the thesis simply as *agents*, are programs that execute in a certain environment, and they act on behalf of another

program or user to accomplish a specific goal. They have a number of special features that makes them different from other types of software [FG96]. Some popular properties are described in the following:

- *Autonomy.* Once their goal is set, they can perform the necessary actions to achieve them without needing any further intervention from the user or program that launched them. Instead, the agents take their own decisions according to their status, their environment and the way they were programmed.

- *Communication.* Agents can communicate with other agents or entities (for example, with programs or users) in order to obtain information, or coordinate their actions. The communication can be performed by using an *Agent Communication Language* (ACL) [GK94] if they communicate with other agents, an application interface (to communicate with other programs, such as databases), or a user interface (e.g., a GUI or text interface) if they communicate with a human user.

- *Cooperation.* They can cooperate with other agents in order to achieve their goals. This cooperation can be very simple (e.g., a client/server interaction) or very complex if they require to follow a specific protocol or negotiate with other agents according to a number of rules or constraints.

- *Reactivity.* They can receive stimuli from their environment and react according to them. These stimuli can be, for example, actions performed by other agents, changes in the execution environment, an input from the user interface, or even a combination of several of these.

- *Proactivity.* They are goal-oriented and they keep executing and performing the required actions until the goal is reached.

- *Rationality.* The actions performed by the agents must lead them towards their goal. Therefore, the decisions must be taken following a logic that does not harm the achievement of the goal.

- *Intelligence.* The agents should be able to learn from past situations (i.e., from their environment or from other agents) and, in this way, change their behavior to improve their performance. For this reason, the term *intelligent agents* is used sometimes to refer to software agents.

- *Persistence.* This property represents the ability to keep executing over long periods, avoiding situations such as their termination or blocking when some undesired condition occurs (e.g., the scarcity of some resource, the absence of a network connection, or the unavailability of some specific computer or device).

- *Mobility.* Some kinds of agents can move to other execution environments (through a network connection) and execute in other remote sites in order to achieve their goals. Agents with this property are called *mobile agents* (see

section 2.5.2). From the point of view of data management in distributed environments, this is a very relevant property.

These are the most common properties for a program to be considered an agent, although it is not necessary to have all of them (e.g., the mobility), with the exception of the autonomy [Jen01]. In a *multiagent system* [VdHW08, Woo09], a number of agents execute simultaneously in order to solve a common problem that, otherwise, could not be solved. A summary of all of these agent's properties is shown in Table 2.2, that also specifies if that property denotes a *weak* or a *strong* notion of the concept of agent [WJ95]. The properties whose type of notion is weak are the ones that have been considered as more essential (i.e., more inherent to agents), according to what most scientists have agreed.

Property	**Type of notion of agent**	**Explanation**
Autonomy	Weak notion	They can act without needing intervention
Communication	Weak notion	They can exchange information with other agents
Cooperation	Strong notion	They can coordinate their actions with other agents
Reactivity	Weak notion	They can respond to changes in the environment
Proactivity	Weak notion	They have a goal-directed behavior
Rationality	Strong notion	They perform logical actions to achieve their goals
Intelligence	Strong notion	They can adapt to the environment and learn from it
Persistence	Weak notion	They can keep executing over long periods of time
Mobility	Strong notion	They can move to other execution environments

Table 2.2: Summary of properties of software agents

These systems can be formed by a single execution environment or they may be distributed along several places linked by a network connection. Either way, in a multiagent system the cooperation and sociability of its agents are the most important properties, since they are needed to coordinate their actions and perform the tasks needed to accomplish their goal more efficiently.

Software agents need a suitable environment to execute, that is provided by a middleware known as the *agent platform* [RD00]. This software must be executed in all the computers/devices where the agents are hosted, and makes it possible to grant them many of their aforementioned features, such as their autonomy, reactivity, communication abilities, persistence, or mobility. Thus, the agent platform provides agents with a number of services that the agents can directly use and do not need to implement; for example, sending a message to another agent or group of agents (that can be located in the same computer/device or in a remote one), obtaining information about the status of the current platform (e.g., its load, memory occupancy, presence or not of other agents), obtaining a list of other execution environments that may also be executing agents, saving the current agent's state to the platform's local storage in case it is shut down, or (in the case of mobile agents) perform a movement from

one execution environment to another. More details about agent platforms and an overview of some popular platforms can be found in Section 2.5.3.

2.5.2 Mobile Agents

A mobile agent is a special type of software agent that runs on an execution environment (traditionally called *place*) and can autonomously travel from *place* to *place* (within the same computer or between different computers) and resume its execution at the target [MDW99]. Thus, mobile agents are not bound to the computer/device where they are initially created and they can move freely among computers/devices. To be able to use mobile agents, it is necessary that the agent platform provides the *mobility* feature, that takes care of the process of sending both the code and data of the agent from one place to another.

Thanks to the mobility capability of mobile agents, it is easy to build complex distributed applications that are at the same time flexible [LO99, UITLM12]. Thus, a mobile agent can carry a required task wherever it is needed. If the task executed by an agent must be changed in the future, a new version of the agent (a new agent implementation) can be delivered. Thus, there is no need to keep specialized software installed on the computers/devices composing the distributed system: only the generic mobile agent platform software is needed to be available on the nodes and an agent implementing the required behavior can move there at any time.

Mobile agents can be designed and programmed to provide interesting benefits (e.g., autonomy, flexibility, and effective usage of the network) that make them very attractive for distributed computing. Particularly, and motivated by the increasing popularity of mobile devices, mobile agents have been found useful for the development of applications in mobile environments [SSPE04, UI13, UITLM15, UI17a]. A mobile environment has a number of special properties, such as the need to rely on wireless communications due to the mobility of the mobile devices, which creates a scenario completely different from that of a traditional distributed environment with fixed networks. Such an environment has a number of advantages (e.g., the processing is not tied to a fixed location) but also some drawbacks, such as the limited computational power of mobile devices and the communication constraints imposed by the use of wireless communications (that usually either offer a low bandwidth, a high latency, and intermittent/unreliable connectivity, or they are expensive or not available everywhere).

The autonomy, intelligence, and movement capabilities of mobile agents render them a powerful and flexible tool to build distributed systems, especially in mobile environments [UIM09, UITLM09]. For example, a mobile agent could be programmed to visit certain devices in a topologically-complex network whose nodes are mobile devices (for example, a vehicular network [UIDM10]) and, once it reaches devices storing relevant data, to process those local data in order to collect interesting information, and return it to the origin. So, mobile agents can move the processing to the data source instead of bringing all the data to the node that will perform the processing (thus reducing the amount of data communicated and benefiting from local interac-

tions as much as possible). Other interesting advantages of mobile agents include the following [LO99]:

- *Ability to support disconnected operations.* An agent can live outside its *home device*, which can be turned off while the agent is performing its tasks elsewhere, to exploit the most suitable resources available (e.g., using powerful fixed computers instead of the limited resources of a mobile device, when appropriate).
- *Minimize the use of network connections.* Instead of the traditional client/server approach, that requires a connection open and alive while the request is being performed, an equivalent request processing would only require the connection to be active during the movements of the mobile agents.
- *Resilience to frequent and/or long network outages.* If the network is not available, they can perform other tasks while they monitor the network status and, when it becomes available again, retry their transmission to reach another place.
- *They can encapsulate protocols.* Since they can move to any other execution environment, they can carry with them the necessary algorithms to achieve their goals, and therefore, the installation of additional software in the computers or devices visited by the mobile agent is not necessary.

For all of these reasons, mobile agents are a serious option to consider when designing a distributed system that needs to be flexible, fault-tolerant, and efficient in terms of network bandwidth usage. These features can be applied to fixed environments (e.g., an enterprise network), mobile environments (e.g., a float of taxis using cellular communications), or a mixture of both types of environment.

For example, mobile agents can play an important role in recent related technologies, such as *cyberforaging* [BFS+02, SKK12], where mobile devices with low computing resources can offload resource-demanding tasks to other nearby computers with higher resources to execute those tasks and return their results. In hardware architectures, there is also emphasis on performing *Near-data processing (NDP)* [GAK15, BG16], which implies placing the processing near the data, instead of transmitting the data to the processor; this is, to some extent, similar to the idea of mobile agents carrying the processing to remote data sources to filter the data locally and so save remote communications. Other related technologies that can take advantage of mobile agents properties are the so-called *edge computing* [BWFS14, Sat17] and *fog computing* [BMZA12, VRM14], where data generated by a number of devices are processed before sending them to the cloud, in order to discriminate which of them should be stored nearby (for efficiency reasons) and which ones could be sent to the cloud. As another example, mobile agents have also been used to support the development of location-based services, such as in the case of the system SHERLOCK [YMII14]. In this system, the users request information and they receive up-to-date answers in heterogeneous and dynamic contexts. Ontologies and semantic techniques are used to share knowledge among devices, which enables the system to guide the user in the selection of the service that best fits his/her needs in the given context. The system

uses mobile agent technology to carry out the processing tasks wherever necessary in the dynamic underlying networks at any time.

In Figure 2.6, an example that summarizes the basics of a mobile-agent based architecture is shown. There are two computers that can communicate by means of a network connection. *Computer 1* executes a mobile agent platform with two *places* or execution environments, *Place 1* and *Place 2*. Similarly, in *Computer 2* there are two other places called *Place 3* and *Place 4*. An agent executing in *Place 1* (*Agent a*) establishes a communication with another agent (*Agent c*) that is executing in a different place inside the same platform (*intra-platform* communication), and also communicates with *Agent d*, that is placed in a different computer (*Computer 2*) using the network. Another agent (*Agent b*) is mobile and decides to change its execution environment, so it moves (by means of a network connection) to *Place 4*, which is hosted in a different computer (*Computer 2*), and then establishes a connection to a local database (which might be not accessible from a remote computer) to query data. Similarly, *Agent e* is also mobile and performs a migration to another place inside the same computer (*Place 3*).

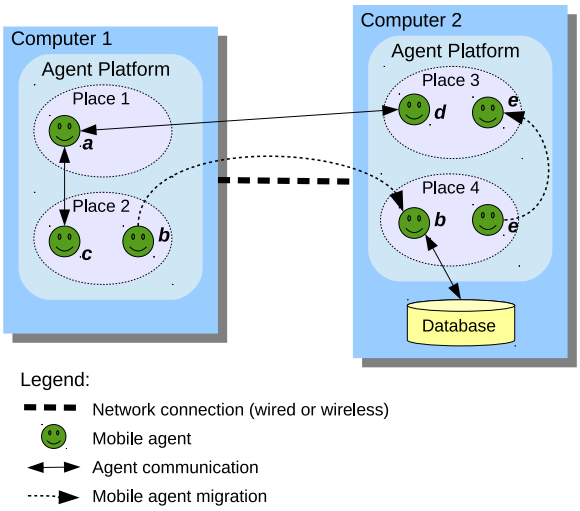


Figure 2.6: Different elements in a mobile agent architecture

2.5.3 Agent Platforms

Software agents must execute in an environment that provides a number of services they need to perform their tasks, with independence of the underlying operating system or hardware. The use of bytecode-based languages like Java eases the implementation of agent platforms and its portability between different hardware architectures. Every device can execute one or more instances of these environments (usually known as *places*, *containers*, or *contexts*, depending on the specific mobile

agent platform considered), where several agents can be running simultaneously.

There are many agent platforms available, which differ in several aspects (such as their general architecture, communication style, etc.) and behave differently in terms of performance, reliability, security, or scalability [TLIM07, BH17]. It even exists a standard promoted by the Foundation for Intelligent Physical Agents (FIPA) [ON98, FIP], which aims to the development of specifications of generic agent technologies that maximize interoperability within and across agent-based applications. However, all of them are very similar concerning the basic services offered to the agents executing on them:

- *Communication service.* One of the strongest points of agents is their communication capability, so the existence of this service is very important. When an agent starts a communication, it needs to determine the message to be transmitted and the destination where it must be sent (i.e., the target agent/s). The message must contain the information to transmit and must be intelligible for both the sender and the receiver. To achieve such mutual comprehension, there exist different agent communication languages (ACLs), that make the communication between different agents possible [KSN00, CLF01, LFP99]. The destination of the message can be a single agent or multiple agents, which can be located in the same execution environment as the sender or in a remote location. The goal of the communication service is to provide agents with a common mechanism to build and deliver the messages to their destination, independently of where the target agents are located. Some platforms also allow the communication between agents by using *remote calls* [ITLM06, TLIM07].

- *Mobility service.* This service allows agents to move to other execution environments. We can distinguish two types of mobility [BDN01, CLZ00]: weak mobility and strong mobility. With *weak mobility* the agents do not resume their execution from the instruction following the migration action, and instead they are always restarted from a given point. On the other hand, with *strong mobility* the agents resume their execution in the next instruction they were when the movement started. With independence of the type of mobility, the process involves three steps. First, the agent determines the destination place and invokes the mobility service in order to be transferred to another place. Then, its code and data are transferred across a network connection to the destination, where another running platform receives them. Once the transmission has finished without errors, the copy of the agent in the origin is destroyed and a new one is created in the destination from the code and data that compose the agent. This process is fail-proof: if there is any problem with the trip of the agent, the agent that attempted to travel will get the control back and decide what to do next.

- *Object tracking.* The object tracking service keeps a record of the locations of all the objects present in a multiagent system [RP01], such as the agents themselves or the execution places available in the distributed system. Whenever a new

object is created, destroyed, or moved, the service must be aware of such an action and update its location. It is very important for the programmer that this service provides a true location transparency, so that once a reference to an object is obtained, it will be kept up-to-date by the system as long as necessary (the programmer will not need to refresh/update the reference).

- *Directory service.* Agents can be programmed to offer different services to other agents or software components. Conversely, agents may also need to use a number of services, offered by other agents, to achieve their goals. The directory service allows the agents to register a description of the services they provide, as well as a common way to query, locate, and access the services included in the registry.

Thanks to all these services, agents can live in a generic distributed environment and perform their tasks effectively. As examples of existing agent platforms (all of which include the mobility feature and can also be executed in mobile devices), we can name: JADE [BPR01, Tel], that has become one of the most popular FIPA-compliant agent platforms and needs a plugin called LEAP to allow its execution in mobile devices; SPRINGS [ITLM06, UIM08], that was developed in the University of Zaragoza, is very robust and scalable, and has a location-aware implementation called GeoSPRINGS [IRTL18]; and VoyagerEdge [Rec], developed by a commercial company (Recursion Software), which includes a very complete API to develop agent-based applications in an easy way.

2.6 Mobile Query Processing in Vehicular Networks

In this section, we describe some possible approaches to the problem of query processing in vehicular networks, along with their limitations. We explain two types of approaches: *pull-based* approaches and *push-based* approaches.

2.6.1 Pull-Based Approaches: Query Dissemination

The classical approach to process queries in such a distributed context, used in traditional Peer-to-Peer (P2P) systems, consists of diffusing the queries to different data sources either directly or using multi-hop relaying techniques such as the one proposed in [BM05]. With this pull-based model (on-demand model, query-to-data model, or reactive model), the query transmitted represents an explicit request of data relevant to such a query. Therefore, as opposed to other solutions that, even in the absence of queries, disseminate data based on their popularity or expected interest for the vehicles, a pull-based approach can potentially retrieve any specific data available in the vehicular network by diffusing queries to retrieve the required remote data. For this solution to work, each target node must be able to understand and process the different types of queries. With this capability, each node can compute a partial query result based on its local data and then deliver it to the destination node. However,

since no fixed data server or any kind of infrastructure is available in vehicular ad hoc networks, new techniques to access data are needed.

In a vehicular network, the mobility of nodes makes the management of an indexing structure, used in traditional P2P systems to decide how to route queries, impossible. An alternative could be to try to disseminate queries towards neighboring vehicles through the vehicular network until these queries reach the target vehicles. However, this implies that the vehicles must be able to understand, route, and process those queries. An additional challenging problem is that of routing the results back to the query originator [IDTL15]. In general, it is not possible to guarantee that the query results computed on these neighbors can be delivered to the node that initiated the query. Indeed, once the query is processed, it may be difficult to know where the query originator is currently located, if it is a moving vehicle. Furthermore, since the vehicles keep moving, it is not even possible to ensure that there is at that moment a communication path to the originator node. Even if the *difficulty to route queries and results* could be overcome, the specific solutions should be embedded in every participating vehicle, and would be quite inflexible and difficult to change or adapt dynamically.

An example of this process is shown in Figure 2.7. There exist several vehicles in a VANET that have different data stored locally in their computing devices, and one vehicle issues a query (e.g., about the location of gas stations in the city). The query is disseminated by broadcasting it to nearby vehicles, that may also forward it to other farther vehicles. When any of these vehicles receives the query, it tries to solve it by processing the data that it stores locally and, if it finds an answer, it transfers it through the VANET so that it reaches the vehicle which issued the query in the first place. Since the query can be received by several vehicles, they may return different answers at different moments, that they will send to the query originator. As commented before, an important difficulty arises due to the movements of the vehicles: while the query is disseminated and processed, the originator vehicle may keep moving, and therefore its location when the answer is obtained will be different from the one it had when the query was initially issued. Thus, the answers will need to find the query originator in its new location, and the answers will probably need to be transmitted using a multi-hop protocol until they reach their destinations. On the other hand, the query originator vehicle may receive asynchronously (i.e., at different moments) several answers from any vehicle in its vicinity, and therefore it is possible that some of those answers are duplicated, or that they arrive when enough answers have been already obtained.

2.6.2 Push-Based Approaches: Data Dissemination

A *push model* (data-to-query model or proactive model) is a common approach for query processing in such highly-dynamic ad hoc networks. With this approach, each vehicle receives data from its neighbors and decides whether they are relevant enough (e.g., based on spatio-temporal criteria, the interests of the driver, etc.) to be stored in a local database, data cache, or knowledge base. Then, the data may be used locally

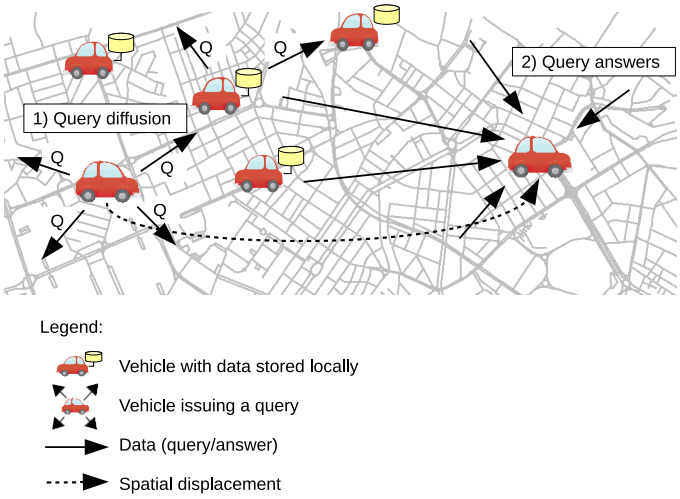


Figure 2.7: Example of query dissemination in a VANET

by a query processor in charge of retrieving data relevant to the driver. The query processing performed with this approach is opportunistic, as data become available only through encounters with other vehicles that transmit them. Some queries may be running locally in a continuous way (predefined implicit queries or continuous queries), such as in the case of queries asking about emergency events that may disturb the driving experience (e.g., accidents), which are relevant all the time (even if the driver does not explicitly ask that information). Other explicit queries can be submitted by the driver at specific moments, according to his/her information needs (e.g., queries asking about available parking spaces). Some examples of push-based approaches can be found in [DCI10, CDI11, XOW04, AYC18].

The major difficulty here is to disseminate data in the vehicular network so that vehicles receive the relevant information efficiently (timely and without unneeded overheads such as duplicate packets or irrelevant data). Nevertheless, with such a push model, only data about events that are potentially interesting for a large set of vehicles are diffused among the vehicles, because of bandwidth consumption reasons. Moreover, the dissemination of certain information is usually restricted to a spatio-temporal area where it is estimated to be relevant. So, a push-based approach also presents some challenges, such as the *difficulty to disseminate the events efficiently*, and it is also quite *inflexible* due to the limited spatio-temporal scope supported and the need to disseminate only data which are expected to be relevant to many vehicles. Due to these limitations, in this thesis we focus on a pull-based approach. Specifically, we use mobile agent technology to proactively disseminate queries in the vehicular network, as needed.

An example is shown in Figure 2.8. There exist several vehicles in a VANET that are disseminating data to their neighbors, and one car receives those data oppor-

tunistically when there is an encounter with any of those disseminating vehicles. The data, or a portion of them in which the driver can be interested (e.g., data regarding certain topics, or that belong to an area surrounding the current position) are stored locally in the computing device. Later, the driver can issue a query (e.g., a list of gas stations in the city) but, instead of disseminating it through the VANET, the data stored locally in the vehicle are processed to solve the query. Thus, the query solving process is faster, but on the other hand the amount of local data regarding the topic of the query may be insufficient, and therefore the query answer may return a small amount of results (or even no results at all).

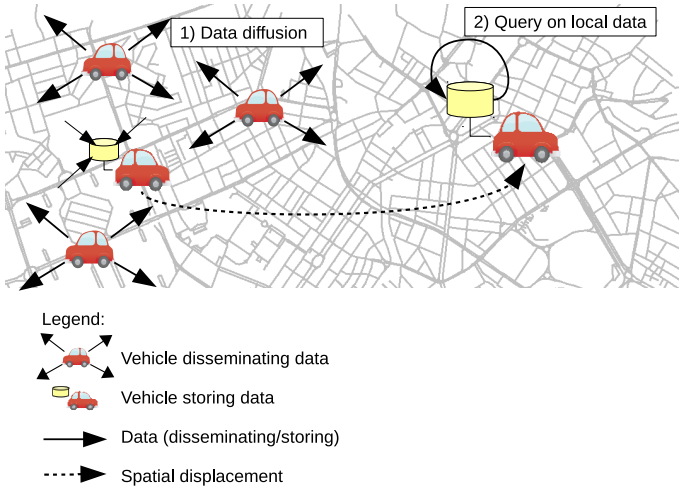


Figure 2.8: Example of data dissemination in a VANET

2.7 Summary of the Chapter

In this chapter, we explained the technological context related to this thesis, which includes the features of mobile computing, the basics of mobile P2P networks, intelligent transportations systems and VANETs, mobile agents, and the processing of queries in vehicular networks.

Thus, we started describing the particularities and limitations of mobile computing, such as the reduced computation capabilities of mobile devices, the necessity of batteries as a source of energy, with a capacity that is limited to a few hours at most, and the need of using wireless communications, which in turn have also limitations in terms of range and bandwidth. However, mobile devices have the key advantage of, precisely, their mobility, so they can be carried to any place and can transfer data without needing any wire, and are can be operated as any moment, since they usually can be started without a long setup time.

Concerning mobile P2P networks, they were created to directly connect two or

more devices without the need of a central common point, as it happens with conventional wireless communications that need the presence of hubs or access points. This has the advantage of enabling the communication among devices in a decentralized and independent form, but it has also some drawbacks, such as the ability to establish communications only with other directly-connected devices, that makes using relatively-complex algorithms to send data to farther devices necessary.

Intelligent Transportation Systems involve a number of components, such as devices with computing capabilities aboard vehicles and wireless devices that enable the communications among the vehicles, and also roadside units, to exchange data in order to ease and make driving more efficient for people and at the same time increase their safety. Closely related to these systems are the vehicular networks, in which the different vehicles traveling along the roads and streets of an area establish network connections among them to exchange data.

Regarding mobile agents, they are software entities that have the ability to moving from one execution environment to another after being transferred by means of a network connection. They are highly flexible and independent, which makes them a very adequate solution for performing some complex task in mobile and distributed scenarios.

Finally, we concluded the chapter by describing the problem of querying data present in a vehicular network, which can be a very challenging task due to, among other factors, the constant movement of the vehicles, the spread of data, and the difficulty of locating and obtaining them.

Chapter 3

Query Processing Approach

In this chapter, we describe the problem considered regarding data management in vehicular networks. Then, we present our proposed approach to tackle the problem, which uses mobile agents, and we discuss our main motivations to use them. After that, we detail how our proposal works by means of a number of algorithms that are extensively explained. These explanations are later expanded with the addition of Petri Nets, that help to understand in a graphical way the behavior of the proposed approach.

3.1 Overall Goal

Our final goal is the development of a system to process queries distributively in a vehicular ad hoc network [UITL17a], where a variety of queries can be considered. We will use the term *mobile query* to refer to any query that, in order to be solved, implies collecting data that are not stored in a single and well-known source but in distributed nodes that can be mobile. Some important difficulties to process such a mobile query are the following: the number of relevant data sources may be high or unknown, the data sources may not be accessible directly through a static network connection, and the locations of the data sources (and their connectivity) can change over time.

Mobile queries can be classified based on different parameters. For example, according to their purpose we could identify *range queries* [WCY06] (which retrieve objects within a certain range/region), *nearest-neighbor queries* [TPS02] (which retrieve a specified number of objects of a certain class which are the closest to a certain object or location), etc. In general, all these queries are *location-dependent queries* (i.e., queries whose answer depend on the locations of certain objects), which are studied extensively in [IMI10] (although not in the context of vehicular networks). In this thesis we focus on range queries, as nearest-neighbor queries could be processed based on range queries, as suggested in [IMI06].

In range queries and in other types of queries, such as in *constrained nearest-*

neighbor queries [FSAA01], the scope of interest of the query is constrained to a certain area, that we call the *area of interest*. Besides, in order to bound the amount of data to process and speed up the query processing, the search of data relevant to the query is usually constrained to data sources located within a certain geographic area, that we call the *relevant area*, which is at least as large as the area of interest. Thus, it is assumed that only the vehicles within the relevant area may store relevant data (i.e., data about the area of interest), even though it is possible that other vehicles outside that area could also hold data relevant to the query. The area of interest could be defined in different ways:

- As a *fixed area*, defined by static absolute coordinates, such as the vertices of a predefined rectangle or the center and radius of a static circle. As an example, consider retrieving the list of affordable restaurants at no more than five kilometers from the work place, or the gas stations located within the municipal boundaries.

- As a *moving area of a fixed size*, whose location is given by the location of a certain vehicle. In this case, the queries are usually called *moving queries* [GL04, IMI06], as the locations of their associated geographic areas change with the current locations of the objects referenced by such queries. As an example, a driver may want to know available parking spaces in a radius of five kilometers around his/her location. It may also be possible to submit queries relative to the location of another vehicle, such as public buses and their proximity to their next stop. In other words, the *reference vehicle* for the query may be the query originator vehicle or a different one. In some cases, there may be some extra information about the trajectory of the reference vehicle, in which case it is possible to estimate its location (and so the area of interest) at a specific future time instant.

- As a *moving area with dynamic shape*. In this case, both the location of the area of interest and its boundaries/geometry may change along time. As an example, we may want to monitor the number of vehicles within a densely traffic area or the concentration of atmospheric gases in the vicinity of a storm. Depending on the specific situation, in some cases the trajectory of the moving area may be known in advance or be predictable to a certain extent.

Obviously, the most challenging cases occur when the area of interest is moving and its shape changes along time. In these cases, a vehicle may become relevant or irrelevant to the query not only due to its own movement but also due to changes in the area of interest. When the area of interest is not fixed, there should be some mechanism to keep track of the area of interest efficiently during the query processing. In this thesis, we consider the case of a fixed area, as our focus is on the development of strategies to keep track of data within a geographic area rather than on defining mechanisms for the dynamic definition of the interest area.

3.2 Motivation for the Use of Mobile Agents

Mobile agents can be very useful in wireless environments [SSPE04], but the application of this technology in the specific case of vehicular networks has not been extensively explored yet. Therefore, in this thesis, we explore the possibility of using mobile agents for processing queries in a VANET. The queries could be used as a building block to develop applications that provide the drivers or other interested parties with useful information, such as applications that retrieve information about available parking spaces, applications that monitor environment factors (e.g., the amount of CO_2, pollution, or the level of noise in an area), or in general applications that retrieve data captured in a certain geographic area (e.g., photos of an area). However, processing queries in vehicular networks involves a number of difficulties, such as the problem of routing the query to a certain area using only short-range wireless communications, searching the data that will be relevant to solve the query distributively among the vehicles present in the network, and finally returning the query results to the query originator once the query has been processed.

The use of mobile agents in vehicular networks can offer key advantages, thanks to their adaptability and mobility features:

- They can bring a processing task wherever it is needed, and the algorithm implemented by the agent can be changed at any time by simply deploying an updated version of the agent's code. This flexibility is extremely interesting in a vehicular network. A mobile agent-based application for a vehicular network can be updated by just releasing new versions of the involved agents, without the need to upgrade the software system of all the vehicles.

- They can move to wherever the data are located, in order to process and collect only the relevant data (filtering out data which may be unnecessary). For example, if we want to obtain some information from vehicles located within a certain geographic area, a mobile agent could move there and process the data locally. Once the most interesting data are obtained, they will be carried along with the mobile agent, keeping the size of the relevant data collected smaller, and making it easier to transmit them in a scenario where communications could be constrained.

- They can also be very useful for data dissemination, since they can adapt their behavior to the changing environment of a vehicular network, improving in this way the data dissemination. For example, simple flooding dissemination protocols will not be efficient when the traffic density is low and the number of vehicles is not enough to route the data towards the destination (besides other problems such as implosion, overlap, and resource blindness [HKB99]). In such situations, other protocols can be used, such as carry-and-forward [ZC08], where the data are stored in the vehicle while waiting for the moment when they can be transferred to other vehicles. Given the variety of protocols that can be developed, mobile agent technology could be used as a basis for building such protocols and making them as flexible and complex as needed. In this

way, an agent can carry data and decide where and when to move, whether it should wait in the current vehicle before jumping to another one, whether it could be beneficial to clone itself, etc. Using this approach, the *intelligence* of performing an appropriate routing is attached to the data themselves (via the mobile agents), and different dissemination protocols can be embedded to dynamically adapt the route followed to the conditions of the network at any moment.

One advantage of using mobile agents is that, instead of sending only data, they can carry along with them the logics or the algorithms needed to transfer and process those data. Since the vehicles are constantly moving, it is not possible to use routing tables to reach the destination. Therefore, it is necessary to evaluate continuously the appropriate next hop according to the traffic conditions and other factors. Once the needed data are reached, it may be processed differently according to their nature (e.g., pollution gases concentrations, gas prices, etc.). The ability of mobile agents to carry such algorithms (that can additionally be changed or enhanced at any moment) make them a very flexible and functional option.

We believe that this is an interesting application for VANETs, that makes it possible to extract valuable information using the vehicles as moving sensing platforms, which is more efficient and dynamic than the traditional solution based on measurement stations deployed at fixed locations. Besides, mobile agents are very adequate since their flexibility and autonomy provide advantages in a fast-changing scenario such as a VANET.

3.3 Proposed Approach Based on Mobile Agents

Thanks to their mobility and autonomy, mobile agents can bring a processing task wherever it is needed, and they can adapt dynamically to the current conditions, which makes mobile agents a valuable means to process distributed data in a VANET. Their *mobility* allows them to hop from one vehicle to another carrying a query and/or its results, and their *intelligence* and *autonomy* (when properly designed and programmed) help them to reach suitable vehicles for data processing.

Regarding our approach, we recall that the vehicles traveling along the roads or streets of a certain area are constantly reading data from their surroundings (e.g., using their sensors or obtaining them from other vehicles) and those data are stored locally in each vehicle's on-board computer, since transferring them to a central location would be costly or impractical. Then, those data, that are distributed and scattered among the moving vehicles of the area, could be queried in order to obtain information. The whole query processing would consist of the following four steps:

1. First, the user creates a query about some information of his/her interest related to an area (e.g., the pollution levels in the city center), which we call the *target area* or *interest area*, and immediately a mobile agent starts its execution with the defined query parameters.

2. In the second step, the mobile agent travels towards the interest area using the vehicles in the VANET to *hop* from one vehicle to another, until it reaches the destination. Every time the mobile agent arrives at a vehicle, the next hop may not occur immediately, since the number of potential vehicles to hop to might be insufficient. In such cases, the mobile agent will stay in the same vehicle waiting for another one that is more suitable, or even use it as a *taxi* that will physically carry the mobile agent closer to its destination area.

3. Once in the target area, the agent hops among the vehicles within the area and, upon its arrival at each of them, processes the locally-stored data in that vehicle to try to find an answer (total or partial) to the query initially created.

4. Finally, in the last step, the mobile agent returns the result of the query to its originator using the same procedure of hopping among vehicles.

In this way, a great amount of data that are scattered, unindexed, and may not be relevant to the user's interests, can be exploited to obtain useful information thanks to the ability of the mobile agents to locate the relevant data in run time and obtain them, while minimizing the wireless communications.

In the following, we explain in detail this process by means of a number of algorithms that are followed to accomplish this task. Algorithm 1 summarizes the process performed by a mobile agent to process a query. In the algorithms that follow, we assume for simplicity that *strong mobility* is supported by the mobile agent platform (i.e., the instruction following a *hopTo*/*moveTo* operation is executed at the target computer, instead of resuming the execution from a predefined point). Although this is not usually the case (most platforms are implemented by using standard Java, which does not support capturing/restoring a given execution stack trace), it is easy to achieve the same effects with *weak mobility* by performing some syntactic translations in the code [BN02]. After describing the four steps, in Section 3.3, we summarize the process by indicating the algorithm executed by the mobile device embedded on a vehicle when a user is going to submit a query, and we highlight the key aspects of the proposal.

Step 1: Query Definition

In the first step, the query itself must be defined by setting some basic parameters that are the input to Algorithm 1:

- The *dataNeed* parameter indicates what the user is asking about (e.g., hotels, petrol stations, parking slots, etc.).

- The *deadline* indicates a maximum time interval by which the user expects to get back the results. For simplicity, we will assume in our description that it is expressed as an absolute deadline (e.g., "the results must be ready by 15:30"). However, as this would require the synchronization of the internal clocks of all the computers involved, it is actually converted to a relative deadline (e.g.,

Algorithm 1 queryProcessingAgent(dataNeed, relevantArea, deadline, amountOfDataToCollect, minPercentData)

Require: *dataNeed* is an expression of the information needs, *deadline* is the time instant by which an answer to the query must be obtained, *relevantArea* is the area that contains the vehicles (data sources) of interest, *amountOfDataToCollect* represents the total amount of data that should ideally be collected, and *minPercentData* is the minimum acceptable percentage of data that must be retrieved.

Ensure: The query is solved before the deadline, or otherwise the query processing terminates silently.

1: travelTo(relevantArea, deadline); {See Algorithm 2.}
2: processQuery(dataNeed, relevantArea, deadline, amountOfDataToCollect, minPercentData); {See Algorithm 4.}
3: returnToQueryOriginator(deadline); {See Algorithm 5.}

"the results must be ready in 10 minutes") by using the techniques proposed in [IMI08]. It should also be noticed that if the deadline is exceeded the query processing finishes silently; the reason is that communicating the failure to the query originator could be difficult and the absence of answer by the deadline can be interpreted as a failure anyway.

- The *relevantArea* defines where the vehicles that can provide *relevant data* for the query are located (e.g., in the city center, within five kilometers around the current location of the vehicle, near specific GPS coordinates, etc.). As explained in Section 3.1, it can be larger than the *area of interest*. For simplicity, we assume that the area of interest is fixed and the corresponding relevant area is also fixed (i.e., they do not move or change their shape).

- The *amountOfDataToCollect* specifies the amount of results that the user would like to retrieve.

  Although, for simplicity, we consider this as a parameter of the query, the idea is to determine it automatically (without user intervention), if possible. For example, if a driver is searching for information about available parking spaces, the number of parkings retrieved could be dynamically adapted, automatically, depending on the level of parking competition at that moment (e.g., if there are many vehicles searching for parking, the query processing could try to find a higher number of parking spaces, to maximize the probability that at least one of them will be available when the driver gets there).

  In other cases there may be no limit on the amount of interesting data that can be retrieved, and so the *amountOfDataToCollect* can be set to ∞.

- Finally, *minPercentData* represents the minimum percentage of data that must be retrieved to consider that the query has been solved. The use of this last parameter is motivated by the fact that in a highly-dynamic environment (such as a vehicular network) it could be really difficult to guarantee that all the interesting data have been retrieved. Indeed, as indicated in [BCOS08], in P2P

environments the query results are usually assumed to be incomplete. So, this parameter can be used to set a minimum quality for an answer to be acceptable. If *amountOfDataToCollect* is ∞, then *minPercentData* is set to 0 and the query processing will retrieve as much relevant data as possible within the allocated time limit.

The definition of the query parameters can be performed by a person by using an appropriate graphical user interface (see Figure 3.1). With this interface, the user selects the target area by drawing a rectangle on a map, and sets values for the different query parameters described above. Default values are also considered to minimize the effort required by the user to submit a query. When the query definition is completed, a mobile agent will be launched, that will be in charge of the query processing by executing Algorithm 1.

Figure 3.1: Graphical user interface to launch a query

Step 2: Relevant Area Tracking

Once the query is defined, the mobile agent must travel towards the relevant area, hopping from one vehicle to another by using the short-range wireless devices aboard the vehicles (see Algorithm 2). The mobile agent can identify the nearby vehicles by using a service of the underlying middleware (the agent execution platform), which in turn can obtain this information by listening to the mobile network identifier that wireless devices broadcast constantly to announce their presence (e.g., in Wi-Fi devices, this information is the service set identifier or SSID).

The mobile agent hops to other vehicles (call *hopAmongVehiclesToReach*, which invokes Algorithm 3, shown later) only if the agent estimates another vehicle as a more promising transportation mode towards the relevant area (if no other vehicle is estimated as promising, then the agent stays in the current vehicle). This assessment can be more or less difficult depending on the knowledge that the agent has about the

Algorithm 2 travelTo(destination, deadline)

Require: *destination* is a target area defined by the geographic coordinates of its boundary, and *deadline* indicates the maximum time interval allowed for the agent to arrive in the destination.

Ensure: The mobile agent reaches the *destination* by hopping from one vehicle to another, or finishes its execution if the specified *deadline* is reached.

```
1: while (¬ inside(destination)) & (currentTime() < deadline) do
2:    hopAmongVehiclesToReach(destination); {See Algorithm 3.}
3:    sleep(MILLISECONDS_BETWEEN_POSSIBLE_TRIPS); {Sleep a while (e.g., 5 s).}
4: end while
5: if currentTime() ≥ deadline then
6:    end(); {The mobile agent execution ends.}
7: end if
```

surrounding area and about the trajectories of the vehicles. For example, if a digital road map is available to the agent and the destination/trajectory of the potential target vehicle is known in advance (e.g., as in the case of a bus route), then the mobile agent will be able to determine with a high accuracy if the vehicle will reach the relevant area or not. Otherwise, it could estimate the probability that the vehicle reaches the relevant area. In this case, when the agent is traveling aboard a vehicle and another vehicle is within communication range, it is possible to consider a number of *hop strategies* that the agent can follow to decide if is better to move to the other vehicle or stay in the current one. Thus, the agent can travel from one place to another by using two complementary mechanisms: by hopping among vehicles (transportation using wireless communications) and by staying in a moving vehicle (transportation via locomotion, using the cars "as taxis").

The process of hopping from one vehicle to another is used multiple times by the mobile agent during the traveling process, and so it has been written as a separate algorithm (Algorithm 3). The specific hop strategy must be encoded within the *isBetter(...)* function shown in that algorithm (line 2).

Algorithm 3 hopAmongVehiclesToReach(destination)

Require: *destination* is the target geographic area, defined by the geographic coordinates of its boundaries.

Ensure: The mobile agent hops to another vehicle if it is estimated to follow a more promising path towards the *destination*.

```
1: for all newVehicle such that withinCommunicationRange(currentVehicle, newVehicle)
   do
2:    if isBetter(newVehicle, currentVehicle, destination) then
3:       hopTo(newVehicle);
4:       break; {Exit the loop.}
5:    end if
6: end for
```

It should be noted that, without loss of generality, in the previous description we

have assumed that only direct communications between vehicles are possible. However, the possible existence of fixed relay devices with Internet connection along the roads (such as the roadside units mentioned in Section 2.3) would enormously facilitate this step of the query processing, as the mobile agent could hop to one of these devices and move between them by using a fixed network connection. This would enable the ability to perform long jumps.

Step 3: Data Collection

When the mobile agent reaches the area of interest, it starts gathering information from the vehicles in the area, hopping from one to another in search of new data (see Algorithm 4). Depending on the nature of the data requested by the user, the mobile agent may need to visit the vehicles located at certain specific places within the relevant area, for example, if some environmental parameter must be monitored. In such a case, the agent considers that spatial area as divided into a certain number of spatial cells (of the same size), and we assume that the agent needs to visit at least $minPercentData$ cells, and obtain the required data there, in order to finish its task. Notice that, by increasing or decreasing the number of cells, we could achieve a more fine-grained or coarse-grained monitoring. Alternatively, instead of dividing the relevant area in cells, we could require the mobile agent to visit a number of vehicles searching the requested data, without needing these vehicles to be located at specific places, as long as they are within the relevant area. For example, the first data collection approach could be used for environment monitoring by using sensors available in the vehicles, for example to measure the level of CO_2 or the amount of noise in an area; in this case, the goal would be to cover the spatial area to perform the required measurements by using the required sensors aboard vehicles. The second data collection approach could be the most suitable one for querying local databases in a number of vehicles, for example to collect information about available parking spaces, received by the vehicles from their neighbor vehicles; in this case, the goal is to visit several vehicles within the area, to query their local databases.

In each vehicle, the mobile agent processes the data stored locally, filtering out the irrelevant data. The relevant data are integrated into the mobile agent's *knowledge base* and carried to another vehicle, where a local processing starts again to integrate new data into the local knowledge base. This process continues while the query remains unsolved/incomplete, and as long as a time limit established for data collection (based on the *deadline* set for the whole query processing) has not been exceeded. Additionally, if during the process the mobile agent leaves the area due to the continuous movement of the vehicles, the agent will need to temporarily interrupt the data collecting process, and it will try to return to the area again by hopping from one vehicle to another, by using the same techniques applied in the previous step. In the following, we explain some aspects of the data collection step in more detail.

Algorithm 4 processQuery(dataNeed, relevantArea, deadline, amountOfDataToCollect, minPercentData)

Require: The mobile agent has reached the *relevantArea*. The input parameters are defined as in Algorithm 1.

Ensure: Either the mobile agent tries to return the query solution to its origin, or it dies if it has not been able to solve the query.

```
1: dataCollectionStartingTime ← currentTime();
2: numClones ← 0;
3: collectedData ← 0;
4: localData ← 0;
5: repeat
6:    deadlineForDataCollection ← deadline - travelToOriginDelayEstimation() - SECURITY_MARGIN;
7:    localData ← collectLocalData(dataNeed);
8:    collectedData ← collectedData ∪ localData;
9:    if (size(collectedData) < amountOfDataToCollect) & (currentTime() < deadlineForDataCollection) then
10:      if inside(currentVehicle, relevantArea) then
11:         if withinCommunicationRange(currentVehicle, newVehicle) & needToHopToAnotherVehicle() & inside(newVehicle, relevantArea) then
12:            hopTo(newVehicle); {The mobile agent will continue collecting data in another vehicle.}
13:         end if
14:      else
15:         travelTo(relevantArea, deadlineForDataCollection); {The mobile agent has left the relevant area and must return there.}
16:      end if
17:      if thisIsNotAClone() & badDataCollectionRate(dataCollectionStartingTime, deadlineForDataCollection) & (numClones < MAX_CLONES) then
18:         numClones ← numClones + 1;
19:         cloneAgent();
20:      end if
21:   end if
22: until (size(collectedData) ≥ amountOfDataToCollect) | (currentTime() ≥ deadlineForDataCollection)
23: if (answeredRatio(dataNeed) < minPercentData) then
24:    die();
25: else
26:    returnToQueryOriginator(deadline);
27: end if
```

Hopping from Vehicle to Vehicle for Data Collection: Deadline

As there is a *deadline* established for the whole query processing, the *deadline for data collection* can be obtained by subtracting an estimation of the time that the agent may need to return to the query originator (carrying with it the query results) plus a *security margin* to diminish the effects of a possible too-optimistic estimation

(see line 6 in Algorithm 4). As the query originator may be moving, the estimation of the time needed to reach the query originator should be reevaluated periodically by calling $travelToOriginDelayEstimation()$ and taking into account statistics about the trips performed by the agent along its life cycle and the distance traversed by the agent when performing those trips (i.e., the effective travel speed of the agent). After setting the deadline for data collection, the agent tries to solve the query by collecting data stored on the local vehicle. In case not enough data have been collected, then the agent evaluates whether it is convenient to hop to another vehicle to continue the data collection or not; to do so, it calls $needToHopToAnotherVehicle()$, which returns *false* if the current vehicle can still be used for data collection (e.g., if the query requires collecting measures of environmental data by using sensors available at the vehicle) and *true* otherwise (i.e., the current vehicle cannot provide more data relevant to the query).

The data collection task ends when the mobile agent has collected the requested amount of data, or when the deadline expires. The amount of data collected may not have reached the minimum ratio established by the parameter $minPercentData$, which is computed by calling $answeredRatio()$ (see line 23 in Algorithm 4). If this ratio does not reach the minimum established, then the monitoring task is considered to have failed and, as a consequence, the mobile agent interrupts immediately its execution. If, on the other hand, the ratio equals or exceeds the minimum, the mobile agent will start the process of returning the results to the query originator.

Use of Clones

In order to increase the reliability or performance of the query processing and the amount of data collected, the mobile agent could also create copies of itself (*clones*). We indicate in the following a strategy that can be applied when the data collection does not progress with enough speed. For simplicity, only the original agent is allowed to create clones of itself (i.e., a clone cannot create another clone). Moreover, a maximum number of clones is considered (MAX_CLONES) in order not to overload the network with many clones of the same agent; a value of 0 implies that clones are not used. Finally, it should be noticed that a clone performing exactly the same actions than the original agent would be of little use. Therefore, in order to desynchronize the behavior of clones, once a clone is created it hops immediately to another vehicle (selected randomly from those within the communication range), if possible, and starts its execution in the new vehicle. Clones act independently of each other, as trying to coordinate the different clones to collaborate among themselves in an ad hoc network would be really challenging.

To decide whether a clone should be created, the algorithm makes use of the function $badDataCollectionRate()$, that simply returns a boolean that indicates if the percentage of data collected so far, considering the minimum amount of data that must be collected, is smaller than the percentage of time spent. In case it is (i.e., if *true* is returned), then it is convenient to clone the agent to try to increase the data collection rate; otherwise, it might not be possible to collect the minimum amount of data required by the deadline. A poor data collection rate could mean that only

a small fraction of vehicles store relevant information or that it is difficult for the agent to jump from one vehicle to another (e.g., due to a weak network connectivity or sparse traffic density). Thus, the use of clones could maximize the probability of obtaining an answer.

Another alternative strategy for the use of clones would be that the mobile agent creates copies of itself already in the first step of the process (all at once), when the query is launched. These clones hop then randomly among the nearby vehicles for a certain interval of time. Afterwards, each one follows the remaining query processing steps. The purpose of this behavior is to make the multiple copies of the mobile agent to spread from the initial point in different directions so that they can reach the target area following different paths, in this way maximizing the probability to increase the data collection rate. When compared to the previous cloning strategy, this one has the advantage that if the mobile agent finds it difficult to reach the target area following a particular route (e.g., due to low traffic density), the other copies of the agent might follow other alternative routes that could reach the destination more quickly. In Section 6.2.6, both cloning strategy are evaluated and compared by means of experiments.

Step 4: Return of Results

When the data gathering ends, the agent travels back to the vehicle from which the query was launched, along with the results of the query (see Algorithm 5). Given that the query originator might be a moving vehicle, a problem arises for the agent to reach it. If a direct connection with the query originator could be established (e.g., by using 4G), which is not expected to be the usual case, then the solution would be easy since the agent could move there immediately regardless the location of the query originator.

Otherwise, unless the agent has some knowledge about the expected route of the query originator, it can be difficult to locate it. A potential solution is to diffuse its expected trajectory along with the query, both embedded within the mobile agent. Another possibility is that the query originator disseminates periodically information about its current location, but it may be difficult to guarantee that these updates will reach the intended agent. Although choosing the best strategy for routing the results to the query originator could be considered an open problem [IDTL15], in our prototype we assume for simplicity that the agent has a suitable estimation of the trajectory of the query originator; in Section 6.2.1, we evaluate the impact of potential errors in this estimation. In any case, the use of mobile agents will help to track the query originator; for example, a mobile agent can more flexibly deal with situations where a communication route to the query originator is temporarily unavailable.

In Algorithm 5, the estimation of the location of the query originator is abstracted in the function *estimateQueryOriginatorLocation()*. The function *travelTo(...)*, described in Algorithm 2, is used by the mobile agent to travel to its destination by hopping from one vehicle to another.

Algorithm 5 returnToQueryOriginator(deadline)

```
1: sucessfulReturn ← false;
2: repeat
3:    queryOriginatorLocation ← estimateQueryOriginatorLocation();
4:    areaOfTheQueryOriginatorLocation ← createSquare(queryOriginatorLocation,
      COMMUNICATION_RANGE); {A squared area centered in the estimated location
      for the query originator and with side twice the communication range (e.g., 200
      meters) is considered.}
5:    travelTo(areaOfTheQueryOriginatorLocation, deadline); {See Algorithm 2.}
6:    if withinCommunicationRange(currentVehicle, queryOriginator) then
7:       hopTo(queryOriginator);
8:       sucessfulReturn ← true;
9:    end if
10: until sucessfulReturn | (currentTime() > deadline)
```

Summary of the Process

Algorithm 6 shows the program that runs on the mobile device aboard a vehicle when a query is going to be launched. As described along this section, the query is first defined by specifying the different query parameters. Then, a mobile agent is created to perform the query processing.

In case the query is solved and the mobile agent reaches the query originator, it will return its result to the user. However, if multiple copies of the mobile agent (*clones*) were created, then it is possible that other different results (carried by the different clones of the mobile agent) will reach the user after the first one arrives. In this case, two approaches could be considered. The most simple one is to discard any data arriving after the first valid result is received. The other option is to aggregate the results when several answers (from different clones) are obtained. Finally, we could also provide the user with new integrated results as they are available. In Algorithm 6 we illustrate this latter option. So, the program waits for the arrival of one or more agents (clones) with their solutions. All the solutions received are accepted, integrated, and shown to the user as they arrive. So, the last solution shown to the user is the most complete one.

The key points to highlight regarding the suggested query processing strategy are the following:

1. It is a novel and flexible approach that uses mobile agents for query processing in vehicular networks.

2. It tackles pull-based query processing (more challenging than push-based query processing and more generic in terms of the queries that can be processed, as queries can be transmitted to retrieve any required remote data).

3. Mobile agents autonomously make hopping decisions based on the information they have about their surroundings, to reach the target geographic areas.

Algorithm 6 launchAndWaitQueryAgent()

Require: The user must provide the parameters of the query (some parameters, such as *amountOfDataToCollect* may be determined automatically for some queries, as explained in Section 3.3).

Ensure: If some result is obtained before the *deadline* specified by the user, it is shown to the user. Otherwise, a *NO_SOLUTION* is notified. In case some solution is obtained, it may be incrementally enhanced (if clones are used) until the *deadline* is exceeded or the user cancels the query.

```
1: setParameters(dataNeed, relevantArea, deadline, amountOfDataToCollect, minPercentData);
2: agent ← create a mobile agent to execute the algorithm queryProcessingAgent(dataNeed, relevantArea, deadline, amountOfDataToCollect, minPercentData)); {See Algorithm 1.}
3: solution ← ∅;
4: totalNumberOfAgentResultsToReceive ← ∞; {Default initial value.}
5: numberOfResultsReceived ← 0;
6: while (currentTime() ≤ deadline) & (¬ isQueryCanceled()) & (numberOfResultsReceived < totalNumberOfAgentResultsToReceive) do
7:    agentArriving ← waitForAgentArrival(deadline); {Maximum waiting limited by the deadline. If this time is exceeded, the waiting is interrupted and agentArriving will be equal to null.}
8:    if (agentArriving ≠ null) then
9:       partialSolution ← getAgentSolution(agentArriving);
10:      solution ← solution ∪ partialSolution;
11:      numberOfResultsReceived ← numberOfResultsReceived + 1;
12:      notifyUser(solution);
13:      if agentArriving.isOriginalAgent() then
14:         totalNumberOfAgentResultsToReceive = agentArriving.numClonesCreated();
15:      end if
16:   end if
17: end while
18: if (solution = ∅) then
19:    notifyUser(NO_SOLUTION);
20: end if
```

4. The agents continuously re-evaluate the situation in order to adapt their behavior, which leads to a good reliability.

5. Clones of agents can be used to further increase the reliability or performance of the query processing and the amount of data collected in a given time frame.

The current proposal is focused towards the development of applications and services that provide the users with useful information (drivers or other people that need to retrieve data from an area). These correspond to the so-called *General Information Services* according to the classification presented in [WTM09]. Safety information services and motion control applications usually have very strict latency requirements, and so they are out of the scope of this work, where mobile agents perform their tasks in a pure vehicular ad hoc network using only short-range ad hoc communications; we

believe that specialized and prioritized data traffic schemes would be needed for these kinds of services, but it might also be possible to analyze their potential support by combining mobile agent technology with other wide-area communication mechanisms (e.g., 3G/4G) and prioritized messaging, when available.

3.4 Modeling with Petri Nets

The proposed data management approach involves a number of elements whose actions can occur asynchronously and at the same time. For example, the movement of the vehicles, the execution of the mobile agents in their on-board computers and their hops from one vehicle to another, etc. Additionally, many of these actions are independent (i.e., they have no influence on each other's behavior), such as the movement of any two distant vehicles, but others are indeed dependent (i.e., one action cannot occur unless some condition is met), such as the hop of a mobile agent from one vehicle to another, since it will not be possible unless there are two vehicles within the range of their wireless communication devices. Moreover, these conditions are not fixed and stable but, on the contrary, are constantly changing and evolving along time.

We have used Petri nets [Mur89] for modeling the different phases of the proposed data management approach and clarifying their behavior [UI17b], since their graphic representation complements the previously presented algorithms and helps to improve their understanding.

In the following diagrams, the execution flow of the mobile agent is represented by a *mark* in the Petri net, which traverses many states or situations (e.g., waiting for a vehicle to get in range, reaching or not the interest area, etc.) that are represented by *places*. The mark can stay in a place until certain condition is met, and then the corresponding *transition* is activated and the execution flow continues as the mark travels to the following state. In order to ease its legibility, we present one separated Petri net for every phase or step into which the whole process can be divided (see Section 3.3). To obtain the whole Petri net, it would only be necessary to join the places with common names from the different diagrams into a single one.

Traveling to the Interest Area

Figure 3.2(a) shows a Petri net that models this stage of the process. The mobile agent is initially created in a given vehicle, as represented by the initial mark in the place "In_Vehicle". Whereas the agent has not succeeded in its attempt to reach the interest area (condition "IA_not_Reached"), it evaluates if there is another vehicle within the communication range that could be a better candidate to transport it to the area. If a better candidate is found (condition "Better_Found"), the agent jumps there, and otherwise it stays in the same vehicle. The process continues until the agent reaches the target area ("IA_Reached"). Notice that there is a transition ("Vehicle_approaching") injecting marks into the place "Vehicles_in_Range" and another transition removing marks from that place ("Vehicle_leaving"), representing the fact

that, at any time, new vehicles can start being within (and out of, respectively) the communication range of the vehicle currently transporting the agent. It should be noted that the agent could apply a variety of strategies to decide if a vehicle is a better candidate or not; for example, a simple greedy approach could select as a better candidate any vehicle which is closer to the target area than the current vehicle.

Monitoring the Interest Area

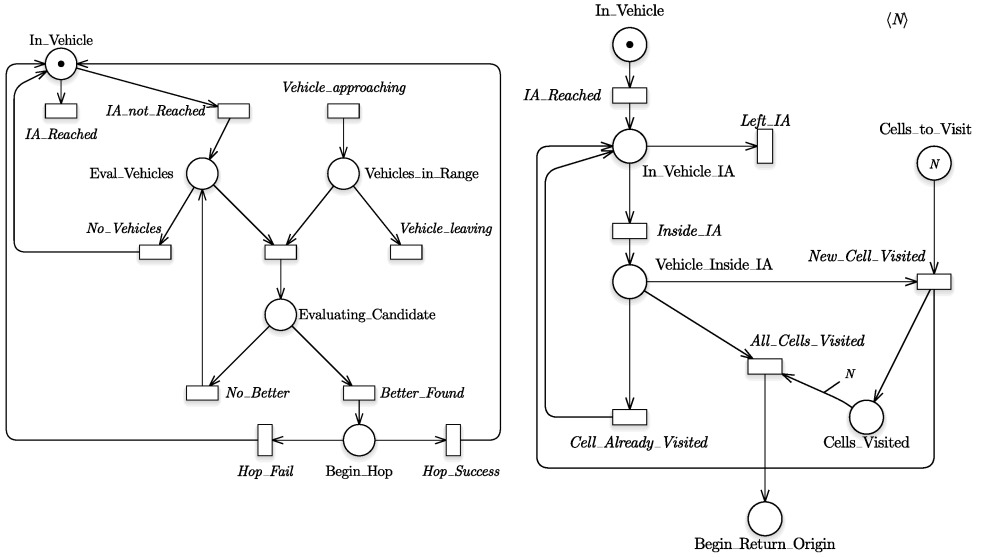
As explained in Section 3.3, there are two possible approaches for the data collection step: visit a number of spatial cells or visit several cars. Without loss of generality, to model the Petri net of the data collection step, here we assume that the agent needs to visit at least N cells to finish its task (alternatively, we could require visiting a minimum number of cars, which would lead to minor modifications to the corresponding Petri net). Figure 3.2(b) shows a Petri net that models the process of visiting the required spatial cells within the area. Each time that there is an opportunity to visit a cell that has not been previously visited, the agent tries to visit it and, if it succeeds, a mark is injected into the place "Cells_Visited"; if not (transition "Another_Cell_not_Reached"), it will try to reach the cell by traveling to other vehicles if necessary. Once the cell has been visited (transition "Cell_Already_Visited"), the agent will try to visit a different cell (the mark representing the agent returns to "In_Vehicle_IA" and the agent will consider a different cell). When there are no marks left in the place "Cells_to_Visit", which means that there are N marks in the place "Cells_Visited", the transition "All_Cells_Visited" is fired and the agent finishes this stage of the process (a mark representing the agent is put in the place "Begin_Return_Origin") to start the last stage (step 4).

Returning to the Target Area, if needed

Notice that, as shown in Figure 3.2(b), the mobile agent can leave the target area if the vehicle that carries it leaves the area (transition "Left_IA"). Figure 3.3 shows the process followed by the mobile agent when the car that carries it leaves the interest area.

Returning to the Origin

Figure 3.4 shows the process of the agent returning to the originator device. Whereas the origin is not reached (transition "Origin_not_Reached"), the agent tries to find alternative vehicles to travel there. Once the origin is found by the agent (transition "Origin_Reached"), the process finishes. As can be observed in the figure, the model is quite similar to the one used to represent the trip of the agent to the target area (shown in Figure 3.2(a)).



(a) Mobile agent traveling to the target area

(b) Mobile agent visiting the cells within the area

Figure 3.2: Petri net of a mobile agent traveling to a target area and visiting the cells inside it

3.5 Summary of the Chapter

In this chapter, we explained the data management approach proposed for the processing of queries in a VANET. It is based on the use of mobile agents, since they have a number of remarkable features (such as their autonomy, intelligence, and mobility) that make them an ideal choice for such a changing scenario as a VANET.

Firstly, the query is defined by the user, and a mobile agent travels to a designated target area, by hopping from one vehicle to another, using short-range wireless communications, or by being physically carried if there are not suitable vehicles to hop to. When the mobile agent reaches the interest area, it looks for the data needed to solve the query: it visits the vehicles that are within that area (by moving to them using the wireless connection), processing in situ the local data stored in those vehicles and collecting the pieces of relevant information. Once the agent finds all the necessary data to solve the query, or when a predefined timeout is reached, it returns to the query originator with the results. For a better understanding, we explained the process followed by the mobile agents by means of algorithms, and also by their graphical representation using Petri nets.

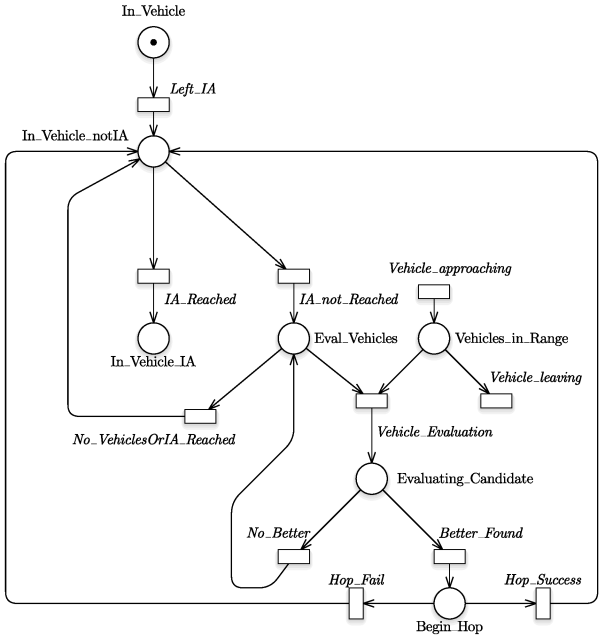


Figure 3.3: Petri net of a mobile agent returning to a target area after leaving it

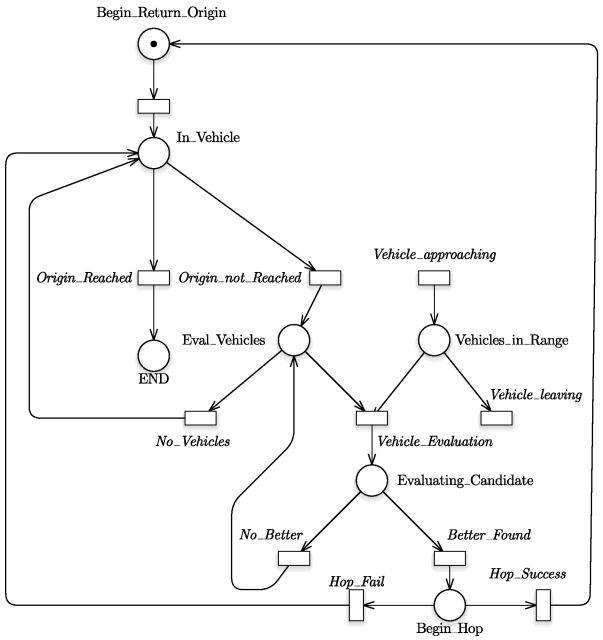


Figure 3.4: Petri net of a mobile agent returning to the origin

Chapter 4

Traveling to a Target Area

One of the particularities of VANETs is the communication among the different vehicles, that is achieved by using wireless devices with a relatively short range (a few hundred meters, at most, in the best conditions). Therefore, data can only be directly transmitted to other vehicles that are in the vicinity, within the range of the communication signal. On the other hand, if the data must be transferred to a location farther than the range of the wireless devices, then a multi-hop approach can be used. To do so, the data are relayed (that is, received and resent to other nodes) by a variable number of vehicles that can communicate with each other (i.e., they are within the wireless communication range) and are located between the source and the destination of the data.

The difficulty with these kinds of approaches is that the intermediate nodes (the *relays*) of a VANET are its forming vehicles, that are constantly moving and changing their positions, which leads to the rapid creation and destruction of network links. In fact, these links could last only a few seconds if the connecting nodes are two vehicles approaching each other at high speeds in opposite directions. Due to the high dynamism of this scenario, the use of static routing tables, that are used in other more traditional networks, is totally discarded and another approach must be taken.

In this thesis, we advocate providing intelligence to the data to be transferred, by attaching them to mobile agents. So, instead of being passively forwarded by the intermediate nodes/vehicles, the data become *smart* and have the ability to actively decide the next step to follow after they have been transferred to one of the intermediate vehicles. This process of being transferred from one vehicle to another will be repeated (possibly, many times) until the data eventually reach their destination, which can be a predefined geographic location or a certain vehicle.

Thus, in order to carry the data where they are needed, one of the main tasks of the agent is to decide the next action to take once it arrives to a new vehicle after being transferred wirelessly from a previous one. These actions are basically two: to stay in the vehicle or to move (or *hop*) to another one. Obviously, if there are no other vehicles nearby, the only option will be to stay in the same vehicle and wait

until some alternative vehicles appear within range. On the contrary, if there are other vehicles where the mobile agent could potentially move, the question is how to choose the best candidate among them to try to reach the destination earlier. The mobile agent will evaluate the current situation by taking into account a number of factors (such as its current location, the position of the other vehicles, the distance to the destination, etc.) and decide to which of them it should travel, after estimating it as the most promising one.

We call *hop strategy* to this evaluation, since there can be many ways to estimate the suitability of a vehicle to lead the agent towards its destination, and therefore different approaches to decide the next hop to make. The election of an adequate hop strategy is very important, since it may have a number of implications for the performance and behavior of the agent, as well as for the whole data management process.

Another type of strategy that a mobile agent can follow is not based on selecting the most suitable vehicle to hop to, but instead on asking for help directly to the drivers to ease the travel of the agent to its destination. In these strategies, some drivers can collaborate with the mobile agent to provide it with the spatial mobility that it may need to reach its destination sooner. Thus, the term spatial crowdsourcing is used to designate the collaboration of users who share their ability to physically moving by driving their cars to certain designated locations, in order to achieve a global goal by helping the mobile agent. This collaboration, however, will surely have a cost for the drivers in terms of time, fuel, etc., so in order to motivate their participation, they can receive some compensation in exchange for their tasks. This compensation can come in many forms (e.g., discount points for gas stations or other shops, money, cryptocurrency, etc.) according to the amount of effort invested by the collaborator to help the mobile agent.

In this chapter, we explain several strategies that can be used by a mobile agent. In Section 4.1, we detail how different hop strategies work. Then, in Section 4.2, the spatial crowdsourcing approach is explained, as well as its advantages and drawbacks when compared to the other hop strategies.

4.1 Hop Strategies

In an ad hoc network, whose nodes establish temporary links to communicate with each other, it is necessary to use a multi relay method to transfer data to other locations that are more distant than the directly-connected nodes. Thus, the data are resent from one node to the next many times until the destination is reached.

In the case of a VANET, the ad hoc connections are established among the vehicles within the communication range of their wireless devices, which are constantly changing due to the vehicles' movements and the presence (possibly intermittent) of obstacles that block the propagation of the communication signals, such as buildings, other vehicles, tunnels, etc. In this scenario, when a vehicle sends data using a multi relay protocol, their final destination can be another vehicle or a geographic location. In the latter case, the data can be routed by the intermediate nodes (i.e., the vehi-

cles), since they know their current geographic position thanks to their geographic positioning system, and they can transfer the data towards the position where the destination is located.

In our approach, this task of routing the data through the VANET nodes is performed by mobile agents, that carry with them the data to transfer. By encoding the routing task in mobile agents, important advantages are obtained. There can be many different suitable routing algorithms, depending on the nature of the data to transfer. For example, it is not the same sending a warning about an accident at a certain location in the road to the potential interested drivers, as sending an informational message about gas prices in the city. In the first case, the data would have a high priority and they should be routed to the vehicles traveling towards the accident point in the same lane. In the second case, the data would not be critical at all and they could be sent to any vehicle in the area near the gas stations. The implementation of all the possible (present and future) routing algorithms in every node (i.e., vehicle) would be inefficient in terms of computing resources, as well as prone to errors. By using mobile agents, instead, only a generic execution platform is necessary, since the mobile agents can bring with them the necessary algorithms.

Another reason, closely related to the previous one, is flexibility. If the routing algorithms were updated, or new algorithms introduced, it would be problematic to update them in all the nodes (i.e., vehicles) in a short time, due to the limited network connectivity. However, by using mobile agents, instead of updating all the nodes, it would only be necessary to deploy the agents with the new version of the algorithms, and they would be used from the first moment.

Regarding the use of mobile agents in VANETs, the way they travel to a destination located at a certain geographic location is by using the wireless ad hoc connections established among the vehicles, to transfer themselves from one vehicle to another that is directly connected. Every time the mobile agent completes successfully one of these movements (that we call *hops*), one of the first tasks they execute upon arriving to the new vehicle is to decide which will be the next vehicle where it will hop to. To take this decision, the mobile agent will need to consider the available information from their surroundings, such as the position of its current vehicle and the others, their speed and heading, the remaining distance to the destination, etc. These and other data can be evaluated by the agent in different ways to decide to which vehicle hop, following what we call different *hop strategies*, and in this way, get closer to its destination. Actually, it is also possible that, as a result of the evaluation, the agent decides to stay in its current vehicle if the other options are worst.

The election of the hop strategy is a very important issue, since it can have a severe impact on the behavior of the mobile agent, as well as on the performance of the whole data management process. Depending on the hop strategy, it can have implications in the following aspects:

- The time needed by the mobile agent to reach its destination. If the election of the next vehicle to hop is not a good choice, then it will unnecessarily travel a longer distance and take more time.

- The bandwidth used by the agent when it moves (along with the data) from one vehicle to another. In a scenario such a VANET, where network connections are constrained by many factors, an excessive usage of communications might be problematic. An ideal strategy would require a minimum number of hops in order to use as little bandwidth as possible.

- The reliability to accomplish its task. The hop strategy should perform logical actions and not act in unexpected ways that may hinder the efficiency of the process or even prevent its successful completion.

The hop strategies that can be programmed in the mobile agent can be as simple or as complicated as required, but they always should take into account these implications. However, it may not be an easy task, since the ideal goal of programming the *perfect* hop strategy is subject to a number of difficulties and limitations:

- The mobile agent may have a limited amount of information of its surrounding environment to make a decision, and its efficiency will likely depend on that. For example, the hop strategy can be very different depending on whether there is a digital map of the area available to the mobile agent or not.

- Even if the mobile agent has good sources of information to help it to decide the best candidate to hop to, this information will comprise only the status or situation of nearby places or actors. That is, only local information will be available, since obtaining information from more distant elements would need a global view that it is difficult to achieve in the environment of a VANET. Therefore, a hop strategy cannot guarantee a globally optimal sequence of agent hops, since the hop strategies can be, at most, optimal locally.

- The high number of vehicles moving in an area, and the fact that their trajectories are not previously known, introduce a factor of randomness that makes establishing a priori the performance of the hop strategy for a given initial situation nearly impossible. So, hop strategies are non-deterministic and in certain situations their performance could be suboptimal.

As a conclusion, it is necessary to evaluate and analyze carefully any hop strategy, in order to know in which circumstances it behaves better, and at which cost.

In the following, we explain in detail different hop strategies that can be used by a mobile agent when it evaluates the convenience to hop to another vehicle in order to reach its destination. The global schema works by following these steps:

1. The mobile agent receives an updated list of vehicles that can be reached directly from the current vehicle using the ad hoc connection. For every vehicle, the agent receives at least its identification, current geographic position, speed and heading. These data are supposed to be exchanged automatically among the vehicles by their execution platforms whenever they are within their wireless communication range. It is also possible that this list is empty, if there are not vehicles nearby.

2. The mobile agent assigns a score to every candidate vehicle from the list, according to the hop strategy followed. A score is also assigned to the vehicle where the agent is traveling on at the moment.

3. If the score of any of the vehicles is better than the score of the current vehicle, then the mobile agent will try to hop to the one with the best score and, if the hop is completed successfully, once it arrives to the new vehicle it will start again this process at the Step 1.

4. If the score of all the other vehicles is worse than the score of the current vehicle, or there are no other vehicles nearby, then the mobile agent will stay in the same vehicle and, after waiting a short lapse (e.g., a few seconds), it will start again this process at the Step 1.

Whatever the hop strategy is, the mobile agent will always follow this general process for evaluating the suitability of a vehicle to carry it nearer the destination. Some minor variations can, however, be introduced to avoid some undesired glitches. For example, in order to avoid loops where the mobile agent hops once and again to the same vehicle or place, additional constraints can be added to the evaluation process. One variation could be not considering as a candidate the last vehicle visited by the mobile agent. Another one could be discarding the chosen candidate if its position is extremely close to where the agent was recently, etc.

In the following, we will describe in detail some hop strategies that have been used in this thesis, although many more could be considered.

Euclidean Distance Strategy

The *Euclidean distance* strategy (designated with the acronym *EUC*, for short) is a fairly simple strategy. The mobile agent hops to another vehicle within the wireless coverage area if that vehicle is closest to the target area than the current one. To compute the distance, the classic Euclidean distance is used.

An explanation can be seen in Figure 4.1. The mobile agent is traveling in vehicle A towards the destination D, and A is considering the vehicles B and C to hop. The distance *d1* from B to D is computed, as well as the distance *d2* from C to D. Since $d1 < d2$, the agent will hop to vehicle B, which seems more promising to reach sooner the destination, since it is closer to it.

Frontal Angle Strategy

In this strategy (abbreviated as *ANG*), the angle of direction of the target vehicle regarding the destination is considered. The agent hops to the target vehicle if this angle is less than 90° and the current vehicle where the agent is traveling does not satisfy this condition.

An explanation can be seen in Figure 4.2. The mobile agent travels in vehicle A towards the destination marked in the map as D, and it is considering the possibility to jump to vehicles B or C. Vehicle A itself is discarded (and that is the reason why

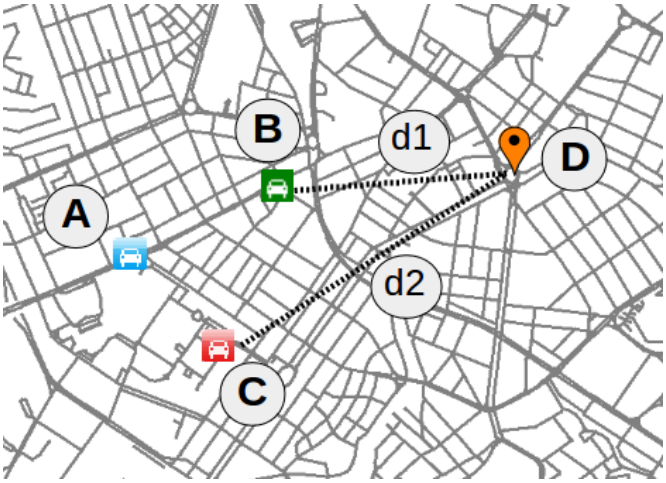


Figure 4.1: The Euclidean Distance hop strategy

the agent is considering the other vehicles) because it is traveling following a direction (the dotted line) in which the destination D does not lies within the frontal angle of 90° that forms the vehicle's direction and the destination.

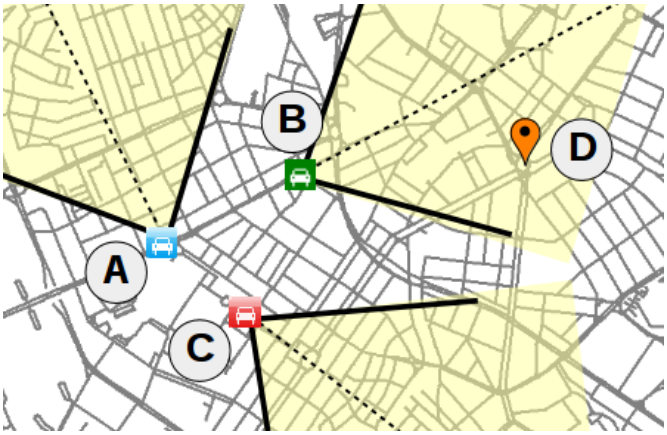


Figure 4.2: The Frontal Angle hop strategy

The same happens with vehicle C, that travels in another direction in which D does not lies within the front angle. However, vehicle B, although it does not follow a direction going directly to D, forms an angle smaller than 90° with the destination, that lies within its frontal angle, so it is the chosen candidate where the mobile agent will hop, since it seems more promising to reach the destination (or at least travel close enough) than the other vehicles.

Encounter Probability Strategy

This strategy (with acronym *EP*) is measured using the Encounter Probability as defined in [DCI10], and it estimates the probability that a vehicle will meet an *event* on a road (e.g., an accident) using the position of both the vehicle and the place of the event, as well as their direction and velocity. In our application scenario, the *destination* does the role of a static *event* as designated in the original calculation.

The formula for the encounter probability is the following:

$$EP = \frac{1}{\alpha \times \Delta d + \beta \times \Delta t + \gamma \times \Delta g + \zeta \times c + 1} \tag{4.1}$$

where every term has the following meaning:

- Δd is the minimal geographical distance between the vehicle and the event over time.
- Δt is the difference between the current time and the time when the vehicle will be closest to the event.
- Δg is the difference between the event's generation time and the moment when the vehicle will be closest to it, or the expected age of the event.
- c is a collinearity coefficient that denotes the angle between the direction vectors of the vehicle and the event.
- α, β, γ and ζ are penalty coefficients with values ≥ 0. They are used to balance the relative importance of the $\Delta d, \Delta t, \Delta g$, and c values.

Some of these terms will take simple values if the destination of the mobile agent is a fixed and static location, as in this case. For example, Δg will be a constant (e.g., zero), since the *event* (i.e., the destination) does not have a generation time and its age when the vehicle will be closest to it is irrelevant. In the same way, the collinearity coefficient c will take the value of 0 since the direction from where the destination is reached is not important. So, the basic factors for this hop strategy are the first two ones.

In Figure 4.3, we can see an example of the geometrical representation of Δd and Δt and the computation of the encounter probability for the hop strategy. The mobile agent in the vehicle A travels towards its destination in D, and it must decide whether it hops to vehicle B or C (for the sake of clarity, we assume that vehicle A has a worse EP value). So, the encounter probability for all the vehicles is calculated using Equation 4.1. In the figure, vehicle B's Δt_B represents the time interval between the current time and the time where the vehicle will be nearest the destination point, and Δd_B the minimum geographical distance between the vehicle and the destination. Regarding vehicle C, Δt_C and Δd_C have the same meanings, respectively. Graphically, the size of Δt_C and Δd_C are greater than the size of Δt_B and Δd_B. Therefore, after applying Equation 4.1, the result will be that vehicle B has a higher EP than vehicle C, and thus the mobile agent will hop to B, since it seems more promising to reach the destination sooner.

Figure 4.3: The Encounter Probability hop strategy

Street Map Distance Strategy

The *street map distance* strategy (with acronym *MAP*) assumes that the involved vehicles have digital road maps of their surroundings. The route to the destination is computed following the street layout and the total distance is obtained as the sum of the lengths of all the street segments of the route. The agent hops when another vehicle within the communication range has a route whose length is shorter than the current one.

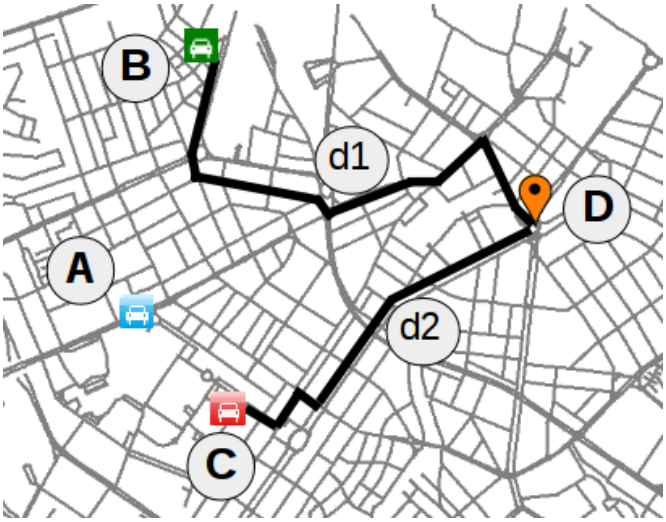


Figure 4.4: The Street Map Distance hop strategy

In Figure 4.4 we can see an example. The mobile agent in vehicle A travels towards its destination in D, and it must decide whether it hops to vehicle B or C. Both vehicles have a similar straight (i.e., Euclidean) distance to D, but if the routes are computed following the streets, the distance *d1* from B to D is longer than the distance *d2* from C to D. Therefore, the mobile agent will choose the vehicle C, since it is the one with the shorter route to the destination.

As stated before, this hop strategy requires that the mobile agent has access to a digital map of the area, in order to be able to compute the route following the streets and taking also into account their directions and lengths. Although nowadays is quite easy to have such maps (e.g., thanks to initiatives such as OpenStreetMap [Ope]), the performance of this strategy could decrease if the maps are not entirely accurate because they are outdated or belong to a region not entirely mapped.

Another consideration is that the computed routes are only a measure of the distance, and in no way it means that the evaluated vehicles will follow that route. This is only an evaluation of how promising the vehicles seem to be, using a distance measure more sophisticated (and hopefully more accurate) than the Euclidean distance.

Trajectories Using Maps Strategy

In the *trajectories using maps* hop strategy (with acronym *MapTraj*), the vehicles have digital road maps of the area (like the MAP strategy) and their drivers follow a pre-computed route, as they would do with a GPS navigator. The mobile agent will hop only when another vehicle follows a shorter route that will travel to the destination. So, as opposed to MAP, this strategy assumes knowledge about the routes of the vehicles.

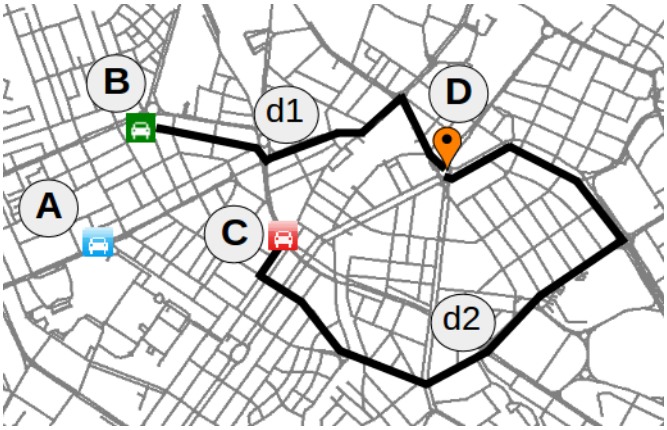


Figure 4.5: The Trajectories Using Maps hop strategy

We can see an example in Figure 4.5. The mobile agent travels in vehicle A towards the destination D and it must decide whether hopping to vehicle B or vehicle C. Vehicle B is located at a longer distance (both in straight line and following the

streets) than vehicle C, which is nearer. However, the route that is following the driver of vehicle C travels more distance (*d2*) than the route followed by the driver of vehicle B, with distance *d1*. Since $d1 < d2$, the mobile agent will hop to vehicle B, as it will follow a shorter route to the destination.

This strategy requires not only access to a digital road map by the agent, but also that the vehicle's driver uses his/her navigation system to program a route and that he/she follows it faithfully. This may seem a bit unrealistic, but it is not entirely unlikely and therefore it is also a hop strategy to consider.

Another variation of this strategy would be considering the time to reach the destination, instead of the distance, if the speed limit of the different street segments is known. This could be another equally-valid alternative, although in such a case there exist a number of external factors that may decrease its accuracy, such as the presence of traffic lights or unexpected traffic jams in the route, that will alter the time calculations.

The Optimal Route Strategy

The *optimal route* hop strategy (with acronym *Optimal*) is an unrealistic strategy where, given the positions of the vehicles along time, the most optimal trajectory to be followed by the agent is computed by searching in the tree of all possible states, using as a weight function the number of hops. This information is only available if we are going to simulate the behavior of the vehicles. When a simulation is performed, this strategy will find the shortest sequence of hops (as a list of vehicles and timestamps of when to hop) that the mobile agent will need to reach the destination.

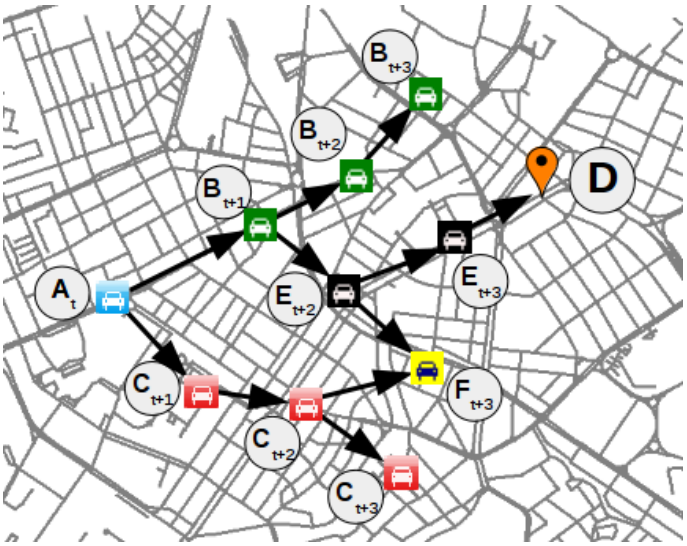


Figure 4.6: The Optimal Route hop strategy

We can see an example in Figure 4.6. The mobile agent starts at vehicle A and its destination is the location at D. The positions of other vehicles designated as B, C, E and F are simulated at different times. Then, the search begins to consider all the possible hops that the mobile agent could make. For example, from A it could hop to B or C at $t+1$. If it hops to vehicle B, then it could stay in B or hop to E at $t+2$, and the process would continue until the mobile agents reaches D and the rest of possibilities of the state tree are explored.

For this simple example, the following sequence of vehicles and hops' timestamps will be obtained:

$$(A,t) \rightarrow (B,t+1) \rightarrow (E,t+2) \rightarrow (E,t+3) \rightarrow (D,t+4)$$

Note that it is perfectly normal that at some moments (for example from $t+2$ to $t+3$) the mobile agent may not perform any hop, but instead stay in the same vehicle if the other options (e.g., hopping to vehicle F at $t+3$) are worst. Also note that the rest of the vehicles, despite participating in the simulation, does not appear in the optimal sequence of hops obtained if they are not relevant to the solution.

The purpose of having this impossible *strategy* is to use it in simulations as an unreachable baseline to compare how efficient the other strategies are.

Summary of Hop Strategies

In the hop strategies presented above, a mobile agent uses different types of information, available from the vehicles and from the environment, to estimate which of the different vehicles within the agent's hop range is the most promising one to reach the interest area sooner. In these strategies, the mobile agent has a passive role regarding the behavior of the vehicles. That is to say, the mobile agent adapts its actions to the movements or trajectories followed by the vehicles, and does not have any influence on them. Table 4.1 shows a summary of the advantages and disadvantages of the different hop strategies, that are evaluated experimentally in Section 6.1.

4.2 Spatial Crowdsourcing

The previously-described hop strategies have some drawbacks. For example, if the traffic density is too low, the mobile agent may not find a path to reach the destination in a reasonable time, or even it may no reach it at all. Similarly, if a vehicle visited by a mobile agent leaves the area, or just parks in an underground parking before the agent hops away, it could get *trapped* or *lost* and the process would be interrupted.

Two possible strategies to try to limit these drawbacks are the setting of a timeout and the use of agent's clones. With the first strategy, once the mobile agent with the required monitoring parameters is launched, a timeout is set, and if the agent does not return any result before the time limit expires, the agent is assumed to be lost and launched again. In the second strategy, the mobile agent creates a number of copies of itself to increase the probability of successfully reaching the interest area by using different alternative routes (see Section 6.2.6). This strategy increases the use

Hop Strategy	Description	Advantages	Disadvantages
EUC	Euclidean distance	Simple to compute	It does not consider the street layout The shortest distance is the only factor considered
ANG	Frontal angle	It considers the vehicle's heading	It does not consider the street layout The direction of the vehicle is the only factor considered
EP	Encounter probability	It considers the vehicle's heading It tries to estimate the probability that the vehicle will reach the target area	It does not consider the street layout
MAP	Street map distance	It considers the street layout	It needs a digital map It is computationally costly
MapTraj	Trajectories using maps	It considers the vehicle's trajectories and the street layout	It needs a digital map The planned trajectories must be known
Optimal	Optimal route	It returns the optimal sequence of hops	It is unrealistic (based on information that cannot be available)

Table 4.1: Summary of advantages and disadvantages of several hop strategies

of the network bandwidth, but maximizing the likelihood of obtaining a result can compensate this cost.

Another strategy for enhancing the behavior of the monitoring process is the use of spatial crowdsourcing techniques [UI16b, UI]. For example, in a certain scenario, the mobile agent might find it difficult to reach certain areas of a city, but the user that starts the monitoring process may be willing to pay a certain amount of virtual money for some help from other vehicles. So, the mobile agent gets the ability to negotiate, with the drivers in the VANET, a way to reach the interest area faster or straighter than in the usual way. Recalling the steps followed to complete the monitoring process (described in Section 3.3), when the mobile agent hops to a vehicle in the second step of the process, it would not only look for other vehicles that seem to travel towards the interest area, but may also try to negotiate to be physically carried there in exchange of a certain amount of virtual money.

As defined in [ZH16], spatial crowdsourcing (SC) implies "location-specific tasks that require people to physically be at specific locations to complete them". With spatial crowdsourcing, a vehicle could be willing to physically transport a mobile agent closer to its destination; we call such a vehicle a *collaborator* or *collaborating vehicle*. However, this collaboration comes at a cost, as it usually requires an effort by the collaborating vehicles. Specifically, we define the *social cost* as the time needed by the collaborating vehicles to deviate from their original routes to carry the mobile agent towards its destination and, once the agent leaves the vehicle, to recover their

previous destinations and continue traveling towards there. In other words, the social cost is the time that collaborators need to invest to help the mobile agent. More specifically, as shown in Figure 4.7, the social cost is computed as the time needed by the collaborator to reach the agent's destination (A), plus the time needed to travel back to the driver's original destination (B), minus the time that it would have taken to directly follow the original route (C). If the agent would have not required this help, the collaborator would have not spent extra time traveling, so this cost needs to be compensated somehow.

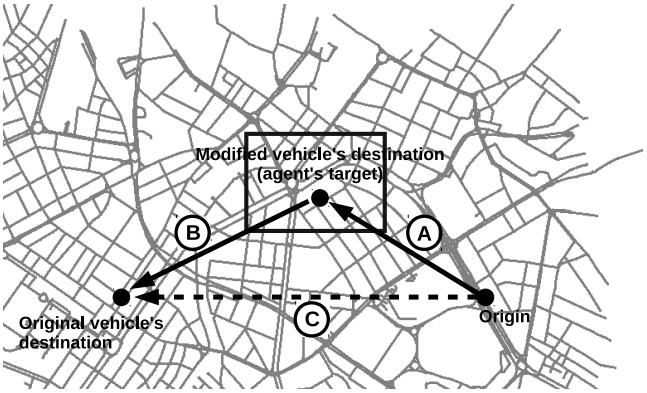


Figure 4.7: Example of social cost calculation for a single vehicle

Specifically, to encourage the collaboration from other vehicles, we propose an approach where a mobile agent can pay *virtual money* to them in exchange of being transported closer to the target area: the agent has a specific budget to process the query and tries to use that budget in the best possible way to reach its objectives and at the same time minimize the amount of money spent. The drivers of the collaborating vehicles could later use the virtual money received to pay other vehicles for similar services. The negotiation required to establish if a driver is willing to alter his/her current trajectory to follow the agent's objectives could be performed automatically whenever the agent is looking for the next vehicle to hop to; in order to accomplish this, the drivers of the area would state previously if they would like to participate in these exchanges and to what extent they are willing to detour from their route to bring the mobile agent closer to its destination. As an alternative, the driver could be explicitly asked if he/she is willing to follow a specified detour.

The basic workflow of the proposed spatial crowdsourcing approach is shown in Figure 4.8, where *SC* is used as an abbreviation of spatial crowdsourcing. In order to save virtual money, the mobile agent will first try to reach the interest area by hopping from one vehicle to another without asking vehicles to collaborate by modifying their original routes. Only if the speed at which the mobile agent approaches its destination is lower than a certain value, which we call the *minimum speed threshold*, the agent will look for nearby collaborating vehicles and will negotiate with them if they are willing to carry the agent. If the potential collaborator accepts, then the mobile agent will

stay in the vehicle and will not hop off it unless the approaching speed exceeds again the threshold value. If the driver does not agree to deviate from his/her trajectory, then the mobile agent will continue trying to reach the destination with the usual (non spatial crowdsourcing) approach, by hopping from one vehicle to another, while at the same time it keeps looking for a potential collaborator. It should be noted that the minimum speed threshold must be either a positive value or 0; a value of 0 for the minimum speed threshold means that the agent will never ask for help to potential collaborating vehicles (i.e., no spatial crowdsourcing will be used). In Section 6.3.1 we perform an experiment to determine the best value for this parameter.

So, in our proposal, we combine the basic monitoring approach, based on hopping from one vehicle to another by exploiting wireless communications, with spatial crowdsourcing. It should be noted that the switching between these two methods can occur several times at any moment during the trip to/from the target area. For example, if the agent is traveling too slowly (maybe because it is in a low-traffic street/area that makes it difficult to find other vehicles traveling towards the desired destination), it can use a collaborator to leave that street/area faster, and once its speed reaches a higher value, use again the method of hopping from one vehicle to another until the destination is reached.

It is also relevant to emphasize that we consider the mobile agent's *approaching speed* to the target area, which may not be the same as the speed at which the agent is traveling. For example, the agent may be in a vehicle moving at 50 km/h, but if its trajectory is inside a street located parallel to the target area then the approaching speed would be 0, as that trajectory does not bring the agent closer to the target area. Similarly, if the mobile agent is traveling in the opposite direction to the target area, its approaching speed will have a negative value.

In order to know to which extent the spatial crowdsourcing is useful or not for the data gathering process, we consider two versions of the mobile agent with different behaviors exhibited once the agent reaches the target area. In the first version, that we call *SCCA* (*Spatial Crowdsourcing Collecting Agent*), the mobile agent always uses spatial crowdsourcing during the data collection phase (i.e., to travel to the target area cells and to return to the area if the vehicle leaves it during the data collection). In the second version, that we call *PHCA* (*Pure Hopper Collecting Agent*), the mobile agent never uses the help of collaborators during the data collection phase; therefore, the agent tries to reach the cells within the target area by only hopping from one vehicle to another. Note that, if the purpose of the agent is processing a query inside the target area (by hopping among the vehicles searching certain data) instead of monitoring the area (by visiting its cells), then the PHCA version does not have any sense. The difference between these two versions only affects phase 3 of the query solving approach (see Section 3.3), since in both versions the mobile agent uses spatial crowdsourcing to travel to the area as well as to return the results to the originator vehicle. In Section 6.3.1 we compare these two strategies by performing a number of experiments using both of them.

Several payment schemes could be considered. In our prototype, the amount paid by an agent is directly proportional to the time spent by the vehicle carrying the agent

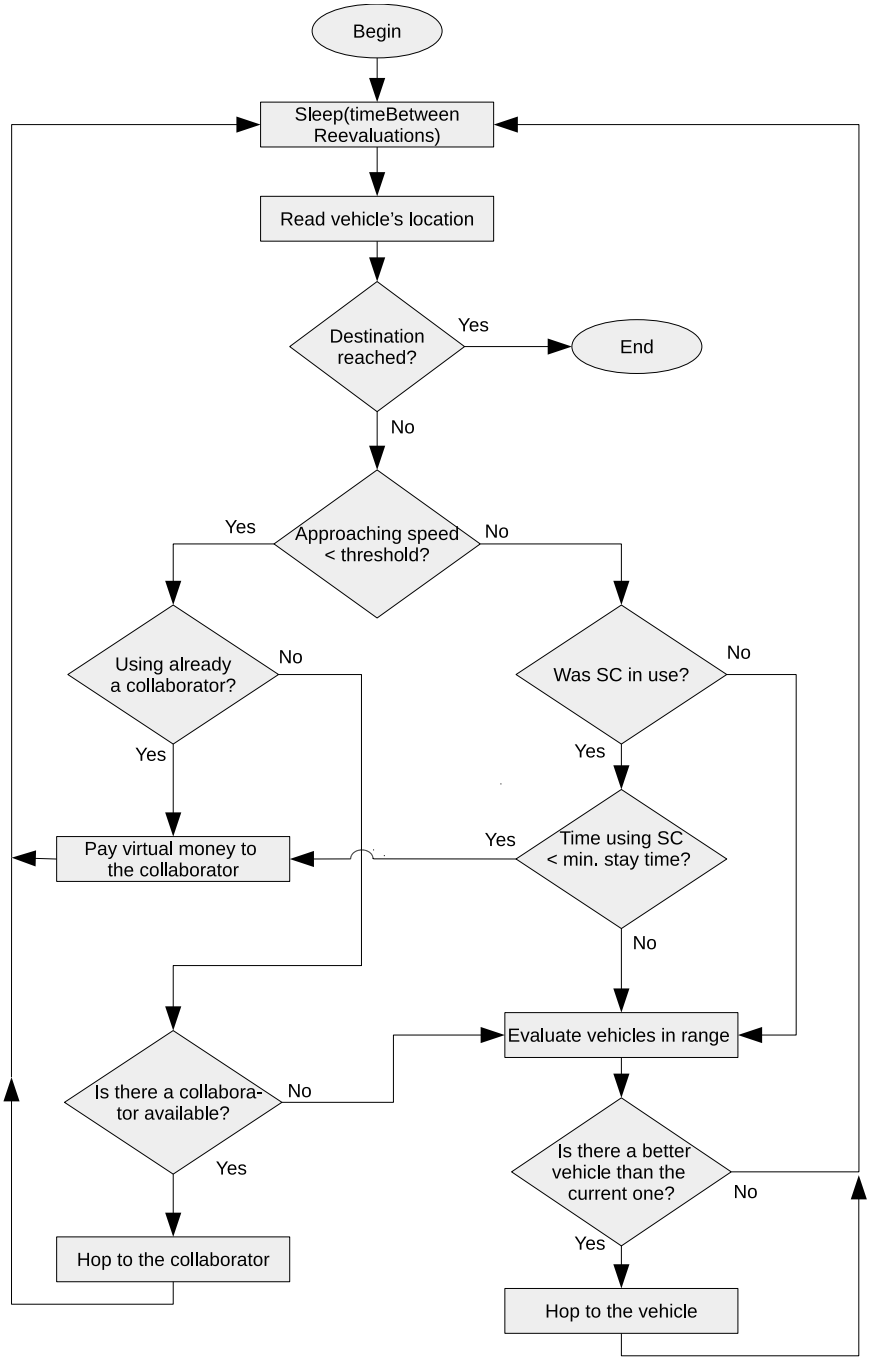


Figure 4.8: Basic workflow of the spatial crowdsourcing approach to travel to an area

and there is no minimum fare. Therefore, the agent will pay one unit of virtual money per every time unit (e.g., every second) spent in the vehicle acting as a "taxi" (i.e., following the route required by the agent). This route will likely be different than the one the vehicle was following originally. However, it might also happen that a driver obtains virtual money even if he/she does not really need to modify his/her trajectory to transport the agent (because the route required by the agent already matches the future trajectory of the vehicle) or that he/she gets a considerable amount of money in exchange of a slight deviation from the previous trajectory. This is unavoidable, as the intended trajectories of the drivers are unknown to the agent, and so the agent cannot know that it could achieve its objectives just by staying in the vehicle, even without paying for the vehicle to change its trajectory.

It should be noted that a *minimum stay time* in the vehicles has been established for the following reason: for low values of the minimum speed threshold, once moving by staying in a collaborator it is very easy to reach the threshold, and thus the agent may immediately hop to a more promising vehicle (i.e., one that approaches the target area faster). This may render the behavior of the mobile agent somewhat unstable, and in addition it may require a higher use of bandwidth, since the agent is constantly hopping to new vehicles. So, in order to limit this effect, a hysteresis value for the agent's stay time (minimum period that the mobile agent must stay in a collaborating vehicle before hopping to others) is considered. In this way, the bandwidth used and the likelihood of switching continuously between spatial crowdsourcing and hopping among vehicles can be reduced. In Section 6.3.4 we perform an experiment to find out the influence of the minimum stay time.

Summary of Spatial Crowdsourcing Strategies

In the spatial crowdsourcing strategies presented above, a mobile agent uses the help of collaborating drivers to travel nearer its destination when its approaching speed is too low. In these strategies, the mobile agent has an influence on the trajectories followed by the collaborating vehicles, that can alter their original trajectories to help the mobile agent to reach its destination sooner, in exchange of a compensation. Table 4.2 shows a summary with the advantages and disadvantages of the different spatial crowdsourcing strategies considered, that are evaluated experimentally in Section 6.3.1.

4.3 Summary of the Chapter

The mobile agents that are used for the data management approach in a VANET must be able to travel to any location in order to carry with them the code to process the data stored locally in the vehicles, and also the results obtained after that processing.

To accomplish this, a mobile agent uses intermediate vehicles as relays, and hops from one to another until it reaches its destination. However, this can be difficult, since the vehicles are constantly moving, the routes they follow are generally unknown, and the network links established among them have a short range and last only a few

SC Strategy	Description	Advantages	Disadvantages
SC/PHCA	Pure hopper collecting agent	It needs less collaborators	The collecting data task in low-density traffic areas can be challenging
SC/SCCA	Spatial crowdsourcing collecting agent	It can collect more data in low-density traffic areas	It needs more collaborators
Without SC	It does not use SC	It does not need to compensate drivers	The whole monitoring process in low-density traffic areas can be challenging

Table 4.2: Summary of advantages and disadvantages of several spatial crowdsourcing (SC) strategies

seconds. Therefore, it is not possible to plan in advance the sequence of vehicles to hop to as relays; instead, the mobile agent must determine it in situ, and while it is traveling to its destination, the best way to reach it.

One way of achieving this is by evaluating the nearby vehicles according to information such as their position, speed, heading, etc., and the agent will hop to the vehicle with a higher score. We call *hop strategies* to these evaluation functions, that can have different complexity and be based on the availability of different type of information.

Another way to reach physically the target area is by means of spatial crowdsourcing, where the mobile agent is carried closer to the intended destination by collaborating human drivers. These drivers must therefore alter their original routes, in order to carry the agent, which has a cost in terms of time and fuel, so they are compensated in exchange according to the extent of the help provided to the mobile agent.

Moreover, we have evaluated both the hop strategies and the spatial crowdsourcing approach (the results of this evaluation are shown in Chapter 6) using a simulator, in order to know which one is better and how they can be influenced by some environmental factors. In the case of the hop strategies, we concluded that MAP is the best one in terms of time and bandwidth usage, as long as a digital map is available. Regarding the use of spatial crowdsourcing, its use clearly enhances the basic hop strategies, especially in the situations where they are weaker, such as in areas with low traffic of vehicles.

Chapter 5

Evaluation Methodology

In this chapter, we explain the methodology followed to evaluate our proposals and their different features by means of experiments that will be detailed in Chapter 6. In Section 5.1, we explain the necessity of performing experiments and the systematic process followed to complete them. In Section 5.2, we establish the experimental setup that are common for the rest of the experiments. In Section 5.3, we present MAVSIM, the VANET simulator with mobile agents that is used in most of the experiments.

5.1 Experimental Methodology

The experimentation is a necessary phase in the development and research of a new idea, in order to understand and learn all the possible aspects and features about the subject of the experiments [WRH+12]. This task must be performed in a systematic and objective way and its results, once they are analyzed, must be used to support the initial idea (and the reason why the experiment was done) and reach a conclusion. For this purpose, a set of variables that are supposed to have any influence on the subject are considered, so they take different values, whereas those other variables without any influence are ignored or their values are randomized. During the execution of the experiment, a number of metrics that are relevant and useful to understand the status or behavior of the subject, are measured until the test ends. Then, these measurements are analyzed to reach an explanation or conclusion about the subject, that may lead to additional experiments under different initial conditions, or to make changes or amends in the subject if the results were unexpected.

Additionally, experiments should be repeatable by a third party in such a way that, under the same circumstances, the obtained results would be the same that in the original experiment, or at least so close that the conclusions are equivalent.

In the case of our data management approach, our global goal is to prove that it is valid and performs efficiently in different scenarios. For *valid* we mean that it meets the requirements described in Chapter 3, and by *efficiently* that the amount of resources needed to accomplish that remains reasonable.

The general process followed to perform the experiments of this work (that could also be followed for others) could be summarized in the following steps:

1. **Define the experiment**, which means to clearly establish what we want to find out and why.

2. **Prepare the experiment**, which usually involves the programming of software and/or utility scripts (for example, to launch sequentially different tests with different initial variable values).

3. **Execute the experiment** by performing a number of actions, according to the purposes of the first step, by means of the software programmed in the second step. Depending on the type of experiment, it may be useful to follow its execution and measure its consumption of time and other resources (e.g., RAM memory, CPU, disk space). If at some point some of them grow abnormally, it can mean that there exists some problem with the experiment, and that some action could be taken in order to solve the problem, or save time and/or resources (for example, aborting the execution instead of waiting unnecessarily for its end).

4. **Obtain the results and analyze them** by processing the log files or records generated or measured by the software programmed in the Step 2, or by some other means. These data are processed using some mathematical tool (e.g., a spreadsheet) to obtain statistics, graphics and any other calculation that helps to understand the results of the experiment.

5. **Reach a conclusion** after analyzing the results, and establish the next actions to take. For example, publishing it along with the experiment if it is something new and useful; or repeat the experiments varying the initial conditions for a better understanding of the subject of the experimentation; or create a new experiment to extend the original or to test another subject according to the conclusions learned in the former, etc.

In our case, to perform most of the experiments, it was necessary to program a mobile agent and a script to launch the simulator with many different values (Step 2). Moreover, in order to guarantee that the results were statistically relevant, every simulation was repeated at least fifty times varying only the initial position of the vehicles, and maintaining the rest of variables with the same values. The subsequent steps consisted of checking that the execution did not find runtime problems (Step 3), and importing the data that were written as a result in different files into a spreadsheet software for their analysis (Step 4).

5.2 Setup for the Experimental Evaluation

In this section, we define the base parameters that are used for the experiments performed to evaluate the proposed data management approach. We also describe

the main metrics that we obtain from the experiments, once they are executed, to understand the performance and behavior of the tested approaches.

5.2.1 Experimental Settings

In the following, we present the general experimental settings (see Table 5.1), that have been used in the experiments performed unless specified otherwise in the individual descriptions of the experiments (several experiments evaluate the impact of variations of these parameters). For evaluation, we consider real road networks (extracted from *OpenStreetMap* [Ope]) and set a fixed vehicle density value (by default, a *medium* traffic density of 100 vehicles / km^2 is considered). We simulate the movements of vehicles according to a *pathway mobility model* [HFB09] (i.e., the shortest path between two random nodes of a graph is computed, and that path is used to simulate the behavior of a vehicle) with a speed

The distribution of the streets and buildings in a city constrains the movements of vehicles and can also determine how efficiently the data can be transmitted using wireless communications, due to the presence of obstacles or open wide areas. Therefore, the experiments are performed with three different street layouts, and two different cities for every type of layout:

- **Squared layout**, with long and straight avenues. We have chosen New York and Barcelona as representatives of this type (see Figure 5.1).
- **Old city layout**, with many short and curved streets. We have chosen some portions of London and Madrid (see Figure 5.2).
- **Mixed layout**, with a mix of long-straight and short-curved streets. We have chosen San Francisco and Zaragoza (see Figure 5.3).

(a) New York (USA) (b) Barcelona (Spain)

Figure 5.1: Cities with squared-layout streets

The wireless communication range considered between the communication devices of the vehicles is 250 meters [MFT+13] with a bandwidth of 54 Mbps (nominal transfer

(a) London (UK) (b) Madrid (Spain)

Figure 5.2: Cities with old-city layout streets

(a) San Francisco (USA) (b) Zaragoza (Spain)

Figure 5.3: Cities with mixed-layout streets

rate of IEEE 802.11g). Moreover, buildings are simulated appropriately, as they may block the signal propagation.

A few other parameters deserve further explanations:

- *Distance to the target area.* The initial position of the vehicle is at a distance of 1 or 2 kilometers (depending of the experiment) from the target area, which has a surface of 0.25 square kilometers.

- *Density of vehicles.* The vehicle density can be defined as the number of vehicles present per surface unit [DB15] and it can be measured in terms of vehicles per square kilometer. Another similar vehicle density measure unit is the number of vehicles per road length unit [Ker09], that is, vehicles per kilometer or per mile; this unit is useful for measuring the traffic flow in a mostly-linear road topology (such as a highway), but it is not suitable for urban scenarios, where vehicles can communicate not only with others present in the same street but also with vehicles traveling along other nearby parallel or perpendicular streets.

Parameter	Default value
Dimensions of the map fragment	4 km × 4 km
Size of the target area	0.25 km^2
Distance to the target area	1 or 2 km
Density of vehicles	Medium (100 vehicles/km^2)
Speed of the vehicles	50 km/h ± 10% (random variability)
Mobility model	Pathway mobility model
Hop strategy	MAP (map distance)
Sensor reading delay	5 s
Data to collect (number of samples)	6 (1 from each cell of a 3 × 2 grid)
Data collection timeout	3 minutes
Total agent size	200 KB
Communication bandwidth	54 Mbps (IEEE 802.11g)
Communication range	250 m
Mobile agent's hop delay	1 s
Buildings block communication signals	Yes
Relevance of information	50% of vehicles
Time limit	10 minutes

Table 5.1: Experimental settings

Of course, this is not always possible due to the presence of obstacles such as buildings but, even in that case, opportunities can appear to perform the communication in open city areas such as main squares or street junctions. For these reasons, we use the *vehicles per square kilometer* (or vehicles/km^2) unit, since it is more appropriate for urban scenarios like those we are testing.

- *Hop strategy.* We have chosen by default a strategy that uses as a criterion the remaining distance to the destination (*map distance*), computed by adding the lengths of the streets to traverse in the road network. This hop strategy provides good results (see Section 6.1.1).

- *Mobile agent's hop delay.* This is the time needed by the mobile agent to complete a hop from one vehicle to another. We assume that in our test scenario the wireless technology used is the widespread 802.11g, that operates in ideal conditions at a maximum speed of 54 Mbps. For the mobile agent's hop delay we use the value we obtained in the experiment in Section 5.2.2, that is, 1 second.

- *Data to collect.* Without loss of generality, in the experiments we assume that the target area is partitioned based on a 3 × 2 grid, which makes a total of six cells, which the mobile agent will have to visit to recover data from them.

- *Data collection timeout.* To avoid a potential situation where the data collection task takes too long, a timeout of 180 seconds is set. Reaching this timeout will

cause the mobile agent to finish the task and start the process that intends to return the data to the originator vehicle or node where the query process started. In this case, it might happen that the amount of data collected by the agent is smaller than the desired quantity. Therefore, the amount of collected data is a performance metric to consider.

- *Time limit.* To avoid situations where the query process takes too much time to complete (for example, if an agent finds an unusually hard-to-solve situation), an overall time limit of 10 minutes is set. If this limit is reached, the simulation is aborted and its results are ignored, but the failure is counted and used to measure the reliability of the algorithm under the conditions that were being simulated. Thus, the time limit represents the maximum overall time that the user is willing to wait to obtain an answer.

With this setup, we repeat every simulation 50 times, with different random starting positions for the vehicles, and compute the average of the results obtained.

5.2.2 Determination of a Mobile Agent's Hop Time

In order to achieve their task, mobile agents must transfer themselves from one vehicle to another using wireless communication devices. With this experiment, our goal is to measure the amount of time needed by an agent to jump from one vehicle to another. Then, we will use this value as an input for the rest of the experiments.

In this test, we use the mobile agent platform SPRINGS [ITLM06], and we program a simple mobile agent to follow a fixed route through a number of execution *places* hosted on different devices that can communicate among them wirelessly by using their built-in Wi-Fi devices. Specifically, we use three different Android devices (see Figure 5.4) with the following features and roles: 1) a Samsung Galaxy Tab *tablet* with 512 MB of RAM and Android 2.3, that hosts place *C1* and the *RNS* (*Region Name Server*) of SPRINGS, which is a process in charge of some tracking services for the execution places belonging to its administrative domain (its *region*); 2) a Samsung Galaxy Nexus high-end *smartphone*, with 1 GB of RAM and Android 4.2, that hosts place *C2*; and 3) a Sony-Ericsson Xperia 10 Mini Pro (low-end) *smartphone* with 128 MB of RAM and Android 2.1, that hosts place *C3*. They communicate among them by connecting to the same wireless network through an IEEE 802.11g Wi-Fi router.

For this experiment, a mobile agent visits the three previously-described devices and measures the time taken to hop from one place to another. The mobile agent starts its execution on place C3, hops to place C1, and returns again to C3. This is repeated 10 times (performing a total of 20 hops). The total time is summed and the mean value to hop between C1 and C3 (*hop1*) is computed. A similar process is repeated with the mobile agent starting again in C3 and moving to C2, obtaining the time of *hop2*. Finally, the same process is repeated with C1 and C2 (*hop3*). Notice that, even though we use the terms *hop1*, *hop2*, and *hop3*, these three hops are not a sequence of hops performed consecutively.

The need to measure the hop time for every pair of places separately and with that back-and-forth procedure is due to clock precision reasons: the mobile agent uses

Figure 5.4: Devices used to measure the hop times

the local clock of every device to measure the time, and if it moves to another device and gets the time there both clocks should be synchronized to obtain accurate delay measures. Due to security concerns, Android does not support changes to the device's time unless the device is *rooted* or registered to operate in a GSM mobile telephony network, which is not the case in our test environment, where only Wi-Fi connections are used. For these reasons, only the local clock of each device is used to assure the accuracy of time computations.

In this experiment, we specifically measure the time that the mobile agent needs to complete each hop for three different sizes of the data that the mobile agent carries: small (50 KB), medium (200 KB), and big (1000 KB). The data consists of a byte array of the stated size initialized with random values, so they are transferred along with the mobile agent's code every time it hops from one place to another. Figure 5.5 shows the results. As it would be expected, the mobile agent's transmission time is proportional to its size. The transmission times are between 651 milliseconds (for the smallest agent's size) and about 2115 milliseconds (for the biggest agent's size). It is interesting to note that they are smaller when the devices involved in the hop have newer technology. The highest time occurs in *hop1* between the low-end *smartphone* and the *tablet*. The second highest time (*hop2*) is among the low-end and high-end *smartphones*. Finally, the better time (*hop3*) is measured when the mobile agent hops among the most advanced devices (the *tablet* and the high-end *smartphone*).

For the rest of the subsequent experiments described in this section, we consider a medium-size mobile agent, as a typical agent to perform query processing tasks similar to the ones considered in our context is not expected to exceed a size of 200 KB. We also assume that the vehicles carry devices similar to conventional smartphones (not necessarily the latest models) having an 802.11g wireless interface. So, according to the results of this test, we decided to simulate the hop time of an agent based on the average value of the hop times observed for agents of medium sizes when jumping

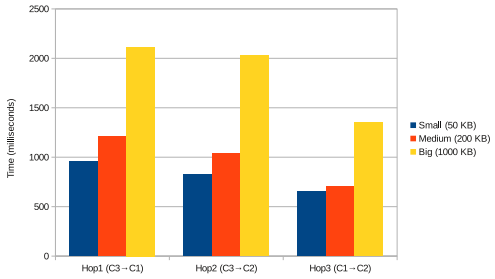


Figure 5.5: Hop times for a mobile agent hopping among different devices

between mid-range devices. The exact value, according to the experiments performed, is 982 milliseconds, so we will round up that value to one second for our experimental settings. A similar value, close to one second, was obtained in a previous set of experiments [UIM08] using the same mobile agent platform (SPRINGS), and PDAs (Personal Digital Assistants) as mobile devices instead of smartphones.

5.2.3 Evaluation Metrics

When the different experiments are executed and they finish successfully, we obtain a result, in the form of a number of metrics. These metrics help us to understand the performance and behavior of the tested scenario, and how it reacts to the initial parameters that define the experimental scenario. The main metrics considered for evaluation purposes are the following:

- *Time.* An obvious metric to evaluate is the amount of time needed by the mobile agent to reach the interest area and/or perform the monitoring task.

- *Number of hops.* This metric counts the number of times that the mobile agent transfers itself from one vehicle to another one by using the wireless connection (i.e., the number of hops performed by the agent), and therefore it can be used as a measure of the bandwidth used by the whole process.

- *Reliability.* The reliability of the query processing is defined as the percentage of simulations that ended successfully within the simulation time limit of 10 minutes.

- *Amount of collected data.* This metric indicates the number of cells in the interest area that the mobile agent visits and reads data from. In our settings there are 6 cells in the interest area, as we divided the interest area in a 3×2 grid, so we count the number of such cells that the mobile agent visits within the timeout of 3 minutes established for the data gathering phase. If this timeout is reached, then the mobile agent stops the data collection task and returns to the origin point with the data collected until that moment.

- *Virtual money spent.* In the experiments related to the spatial crowdsourcing evaluation, this metric represents the payments made by the mobile agent in exchange of help from the collaborating vehicles.

- *Social cost.* In the experiments related to the spatial crowdsourcing evaluation, the social cost, as defined in Section 4.2, represents the extra time invested by collaborating vehicles in helping the mobile agent to perform its task, as compared to a situation where no help is provided (i.e., all the vehicles follow their intended routes).

These are the default parameters that are obtained in most of the experiments that are described in detail in Chapter 6, with the exception of some other special metrics that are properly described if it is required by each experiment.

5.3 Simulation Methodology

In the development process of any software, there exists one phase where it must be tested in order to know if it meets the initial requirements and works as expected, and also assure that the possible error conditions are treated in such a way that the execution of the programs never interrupts or behaves in an unexpected way. Besides, when new algorithms or strategies for finding the solution to a problem are being researched, it is necessary to prove that the provided solutions are correct and that they effectively solve the original problem. When these algorithms or strategies are relatively simple, it may be possible to formally prove their correctness, but for more complicated systems, which are influenced by several variables, this type of demonstrations can become too complex. One alternative is to perform different tests or experiments, setting the variables with different values as the input and, once the output is obtained, analyzing it to know to what extent the result meets the requirements.

The data management approach proposed in this thesis is not an exception, and therefore it is necessary to test it extensively. However, doing so in a real scenario would be extremely inconvenient and costly, since it would involve vehicles moving along roads and streets being driven by actual people, as well as carrying computer devices equipped with wireless communication devices capable of executing a mobile agent platform. This would be only the *physical* or *hardware* part, but it would also be necessary to have some procedure to load the tested software in all the vehicles, make them begin their routes at certain fixed points and follow pre-established routes, or to be present at certain location at a given time. Moreover, some of these and other conditions would be not only costly, but merely impossible to meet, such as the repetition by all the involved vehicles of the same movements performed previously with the same timing, since they would be subject to external random influences, such as the traffic conditions, the weather, mechanical issues, etc.

Thus, in the context of vehicular networks, real-world tests are usually limited to very small and controlled scenarios, mainly as a proof of concept or to obtain measures

that can be used to fine-tune some parameters or to perform proper simulations. The most logical alternative is, indeed, to simulate the behavior of the developed software in a controlled and simplified environment that can be managed more easily and that supports a large-scale simulation with a high number of vehicles.

There exists a variety of simulation software to test and analyze, with the required precision, different aspects of communication networks and vehicular traffic. However, as far as we know, none of the most popular simulators can be easily used to simulate scenarios with the goal of evaluating data management solutions based on the use of mobile agents. Besides, it is not possible to simulate how they may influence the movement of the vehicles as in the case of spatial crowdsourcing approaches.

Motivated by this, we developed MAVSIM [UIL14, UI16a], a vehicular network simulator in which mobile agents can also be simulated, as well as a set of tools intended to ease the analysis of the results obtained in a variety of scenarios. In the rest of this section, we first overview some popular simulation programs and then we describe our proposal.

5.3.1 Types of Simulators in the Context of VANETs

A significant number of network and traffic simulators have been developed, both commercial and free or open source. Most of them have noteworthy features when they simulate their respective intended scenario types. However, when it comes to the simulation of data management strategies using mobile agents in VANETs, they are simply not adequate, since they lack a number of features required in such a specialized scenario. In this section we briefly present some of the most popular ones.

Network Simulators

Network simulators allow the configuration and simulation of detailed parameters of the devices and the communication process. For example, they can simulate the technology used for data transmission (wireless, copper wire, fiber optic, etc.), the data loss ratio, latencies, shadowing effects that make wireless communications more difficult, distance attenuation, etc. Some of the most used simulators of this type are *NS-3* [Ns3] and *Qualnet* (*Quality Network*) [Qua].

Traffic Simulators

Traffic simulators are specialized in the movement of the vehicles and allow to generate traces of their movement, following different patterns and behaviors in different scenarios. Some examples of such simulators are SUMO and VanetMobiSim.

SUMO (*Simulation of Urban MObility*) [SUM] is an open source microscopic road traffic simulator. It allows the simulation of vehicles as single entities, with the ability to traveling through specific routes, changing the road lane, and following the traffic rules. It can handle scenarios with large road networks and a high number of vehicles. It can be enhanced with plugins and can interoperate with other software both by importing and exporting data using different file formats.

VanetMobiSim [Van] features realistic automotive motion models at both macroscopic and microscopic levels. At macroscopic level, it can import maps from the US Census Bureau database, or randomly generate them using a Voronoi tessellation. It has also support for multi-lane roads, separate directional flows, differentiated speed constraints, and traffic signs at intersections. At microscopic level, it implements different mobility models, providing realistic car-to-car and car-to-infrastructure interaction. According to these models, vehicles regulate their speed depending on nearby cars, overtake each other, and act according to traffic signs in the presence of intersections.

Hybrid Simulators

A hybrid traffic-network simulator can simulate both traffic and network elements in a geographic scenario. Some examples of such simulators are EstiNet [Est] and VEINS [NCT].

EstiNet is a commercial product consisting on an extensible network simulator and emulator capable of simulating various protocols used in both wired and wireless IP networks, as well as wireless vehicular networks (including V2V and V2I communications), among others. Regarding its traffic simulation capabilities, it can simulate multi-lane road networks, it incorporates different microscopic vehicle mobility models, and the behavior of any vehicle can be changed as it receives messages from the vehicular network. It has been used for modeling VANETs and other ad hoc networks as well as for the evaluation of real-life P2P applications and traffic signal control algorithms.

NCTUns (*National Chiao Tung University Network Simulator*) is an extensible network simulator and emulator capable of simulating various protocols used in both wired and wireless IP networks, as well as wireless vehicular networks (including V2V and V2I communications), multi-interface mobile nodes for heterogeneous wireless networks, IEEE 802.16(e) mobile WiMAX networks, IEEE 802.11(p)/1609 WAVE wireless vehicular networks, various realistic wireless channel models, IEEE 802.16(j) transparent mode and non-transparent mode WiMAX networks, etc. It has been used for modeling VANETs and other ad hoc networks as well as for the evaluation of real-life P2P applications and traffic signal control algorithms.

VEINS (*Vehicles in Network Simulator*) [VEI] is an open source software that supports online re-configuration and re-routing of vehicles in reaction to network packets, it supports different vehicular mobility models, and relies on detailed models of the IEEE 802.11p and IEEE 1609.4 DSRC/WAVE network layers (including multi-channel operation, quality of service channel access, as well as noise and interference effects). It can import scenarios from OpenStreetMap, including buildings, speed limits, lane counts, traffic lights, and access and turn restrictions. It can also employ validated and computationally inexpensive models of shadowing effects caused by buildings as well as by vehicles. Finally, it supplies data sources for a wide range of metrics, including travel time and vehicle emissions.

It is also interesting to mention that the use of videogames to facilitate the evaluation of data management approaches for vehicular networks has also been proposed

recently (see the *VANET-X* videogame [IMR13]).

5.3.2 A Simulator for VANETs with Mobile Agents

Network simulators have very limited (or non-existing) capabilities to simulate moving objects such as vehicles, whereas traffic simulators cannot simulate network communications, although some of them can export mobility traces to be imported later in a network simulator. A hybrid simulator can simulate both aspects at the same time.

However, none of the simulators mentioned can directly support the simulation of mobile agents, and so they cannot be easily used to evaluate data management approaches based on mobile agent technology. As this is the focus of our research, we were compelled to develop our own simulator (*MAVSIM*, *Mobile Agents in VANETs SIMulator*), that offers interesting functionalities for that context. The simulator has been developed in a quite generic way, with several configurable parameters and a modular and extensible architecture, to facilitate its use in a variety of scenarios to test different data management approaches.

In the rest of this section, we describe the main features of the simulator developed, its architecture, and a use case. See `http://webdiis.unizar.es/~silarri/MAVSIM` for additional information, screenshots, videos, and the possibility to download it.

Features

The main features of MAVSIM are the following:

- It is written in Java, which makes it portable among different architectures and operating systems. It has been successfully tested in Microsoft Windows 7 and 8 (32 and 64 bits), GNU/Linux (32 and 64 bits), and Sun Solaris (64 bits).
- It can run both in graphic interactive mode with a Graphical User Interface (GUI), or in batch mode (useful to execute a large number of simulations or experiments in an easy way).
- Simulations can be recorded and replayed later (with step by step, pause, and rewind and forward functionalities), which facilitates a careful analysis of the whole process. For this purpose, the replay tool shown in Figure 5.6 is used; in that screenshot, some information is shown that is relevant for the monitoring task scenario described earlier.
- Any road map can be downloaded from OpenStreetMap. There is no limitation in the type of layout imported (cities, highways, rural areas, etc.).
- The simulated mobile agents can be programmed in a similar way as they would be programmed using real agent platforms such as JADE [BPR01] or SPRINGS [ITLM06] (e.g., we provide methods such as *moveTo(targetDevice)* to simulate the movement actions performed by the agents). A generic mobile agent platform is simulated with the most common methods and primitives necessary for this.

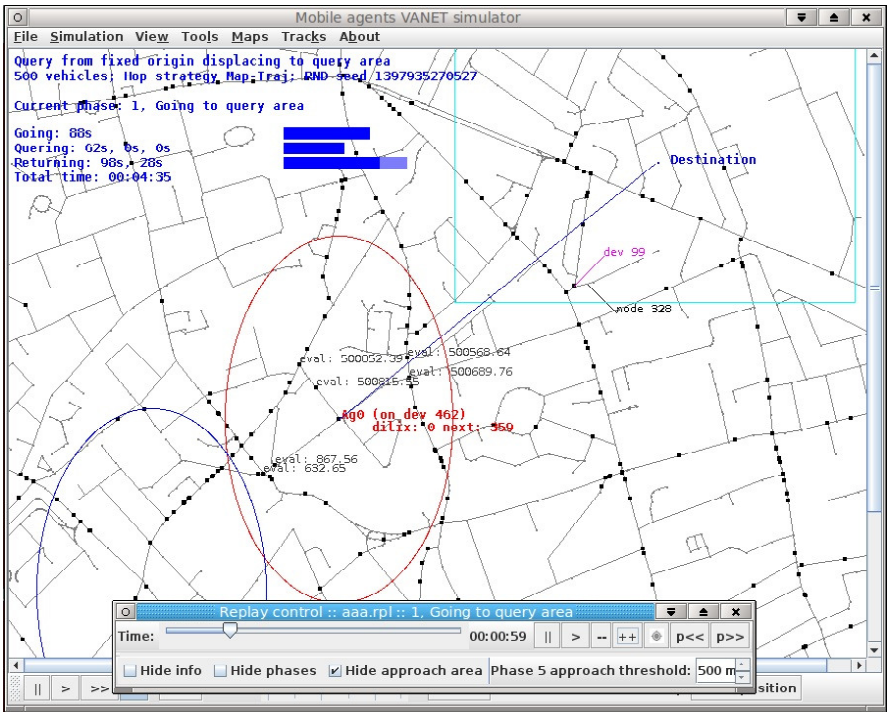


Figure 5.6: The replay tool of MAVSIM

- It can import traces generated by real vehicles or by other traffic simulators, and use them for the experiments with mobile agents.

- It can simulate public transport routes, such as those of buses, tramways or similar vehicles. It can import lists of stopping places (i.e., bus stops or stations) and simulate the movement of many simultaneous instances of public vehicles that travel from one stopping place to the next. When the vehicles arrive at them, they pause their movements for a few seconds before resuming their travel.

- Roadside units (or fixed communication devices) can be simulated. These units can be interconnected by wired connections, which allow high-speed communications among them.

- It includes a variety of movement algorithms for the simulated vehicles, such as *Random way-point*, *Gauss-Markov*, and others.

- In urban areas, it can simulate the presence of buildings that block the wireless signal, constraining the communications. For this purpose, the geometric method described in [MFT+13] is used.

- It can simulate random network communication errors. According to the specified rate, the transmissions will fail and it will be necessary to retry them again.

- The initial scenario conditions for the experiments can be set randomly or using a known *seed*, which allows to reproduce the same conditions and repeat exactly the same experiment (with the same trajectories for the vehicles and other random conditions) if necessary.

A number of parameters can be set to configure in detail the simulation scenario. Table 5.2 shows a summary of some of the general parameters that can be configured, as well as parameters that are applicable for the specific case of the evaluation of monitoring approaches based on mobile agents. Most parameters, when omitted, take a default value; others are optional and do not have a default value.

Parameter	Description	Default
General configuration parameters		
-mcr	Mobile communication range	250 m
-v	Number of vehicles to simulate	100
-vld	Vehicle Linear Density (vehicles/km). Overrides -v	
-s	Average speed of vehicles (km/h)	50
-lat	Latency for a mobile agent's trip	1 s
-errRate	Error rate for communications	0%
-batch	Execute in batch mode	false
-map	Load a scenario map	Last used
-mbx	Number of *mailboxes* connected through a wired network (static nodes with storage capacity and the capability to provide wide-area network coverage)	0
-f	Number of fixed (non-moving) wireless devices	1
-mob	Mobility strategy for vehicles (1 = Random, 2 = Routes, 3 = Straight, 4 = Heuristic & Destination, 5 = GaussMarkov, 6 = ManhattanDynamic)	2
-nobldg	Do not simulate buildings in the map as communication obstacles	false
-rec	Record the experiment to a file for later replay and analysis	
-seed	Seed for random numbers (0 for random seed)	0
Configuration parameters to evaluate monitoring approaches		
-d	Initial distance (in meters) to the interest area	1000
-j	Mobile agent hop strategy (1 = RND, 2 = BEP, 3 = Approach, 4 = ANG, 5 = MAP, 6 = EP, 7 = EUC, 8 = Map-Traj, 9 = Optimal)	8
-m	Maximum number of simulation iterations during which to collect data	899
-mih	Minimum expected improvement to enable an agent's hop	0%
-dimArea	Interest area dimensions (height, width, longitude, and latitude: -mzh, -mzw, -mzw, -mzy)	
-prd	Probability that a device contains relevant data / appropriate sensors	50%
-rph	Max distance to hop (as a percentage of the communication range)	100%
-snru	Selection of the nearest *roadside unit* (0 = none, 1 = once, 2 = continuous)	0

Table 5.2: Summary of configuration parameters of MAVSIM

Structure and Classes

The simulator is designed to be extensible. It contains a number of classes with different functionalities (as shown in Figure 5.7), and the classes are organized in modules (*packages*): one package contains everything related to graphics, another one the functionalities related to mobility strategies, etc. In this way it is easier to maintain the code and perform changes. Moreover, to facilitate its extensibility, some abstract classes and interfaces have been defined, that allow to add new functionalities in a quick and easy way. For example, if it is necessary to add a new mobility strategy for vehicles, there is an abstract class *Mobility* available that includes the necessary methods and attributes for all the mobility strategies and makes adding a new one easier: it would be enough to define a new subclass implementing the methods of *Mobility* and extend the configuration files to add support to select and use that new mobility strategy in the simulations. In the same way, there also exists an abstract class (*TrackReader*) to read different file formats of mobility traces. So, if a new format is needed, only the appropriate subclass must be implemented. Finally, one particularly important abstract class and interface are *Agent* and *IAgent*, respectively: they are intended to allow the programming of new different types of simulated mobile agents.

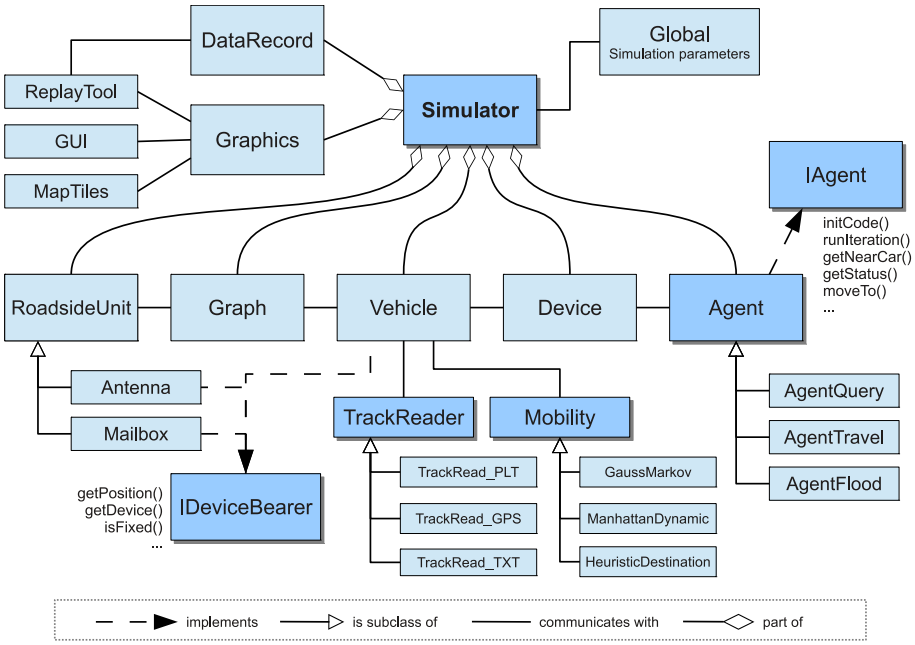


Figure 5.7: High-level overview of the architecture of MAVSIM

In this way, the simulator developed can be easily extended in a number of ways, such as: to incorporate new behaviors for the vehicles (e.g., new mobility strategies),

to define new types of mobile agents with different behavior (e.g., different agent mobility strategies), to define different phases in the data management approach for easy analysis of each of them with the replay tool (e.g., in the monitoring task scenario, phases such as "going to the area", "measuring environment data", and "coming back to the query originator" are defined, as shown in the upper-left part of Figure 5.6), or to add new simulation parameters. We could also simulate data management approaches that do not use mobile agents, as we can simulate that there is a single static agent (i.e., a mobile agent with no mobility) in each equipped vehicle, that implements the required data management functionalities that would be available in the equipped vehicles. Most extensions require defining appropriate subclasses and configuration parameters. Of course, for a comfortable use of the extensions in an interactive simulation mode, changes to the graphical user interface may also be required.

5.3.3 Performance Evaluation of the Simulator

In this section, we present several tests that we have performed to evaluate the behavior of the simulator, as well as its throughput, when it executes simulations in scenarios with different numbers of entities and with varied complexity. Firstly, we evaluate the scalability of the simulator when we increase the number of vehicles. Secondly, we evaluate the scalability when the number of mobile agents in the tested application increases. Finally, we evaluate the simulation overhead when buildings are simulated.

The tests were executed using a computer with the following features: Linux CentOS 5.11 operating system, with 16 GB of RAM memory and one six-core Intel Xeon X5660 processor running at 2.80 GHz. The simulator described in this chapter does not attempt to perform real-time simulations, as it uses an internal clock based on iterations of simulation, rather than a global real-time clock; by default, each iteration in the simulator represents 1 second of changes in the real-world.

Scalability with the Number of Vehicles

This test measures the speed (in iterations per second) achieved by the simulator with an increasing number of vehicles moving in an urban scenario, with the goal of finding out how well the simulator scales with the number of vehicles.

The selected scenario is a portion of the city of San Francisco (as shown in Figure 5.3(a)) extracted from OpenStreetMap. A varying number of vehicles traveling along its roads and streets are simulated by following a Random way-point mobility model [KP11], considering an average speed of 50 km/h. The number of vehicles goes from 1000 to 10000 in intervals of 1000, and for each amount of vehicles the simulation is executed during 1000 iterations.

The results of the test can be seen in Figure 5.8, that shows the average of the results obtained with 50 repetitions of each test (in each test the initial random positions of the vehicles were different). In the figure, we can see how the speed of the simulator decreases as the number of vehicles increases. This was expected, as a

higher number of vehicles implies a higher simulation overload. However, it can be noticed that the performance decrement is not linear, since the slope is steeper for values of less than 5000 vehicles and flatter for a higher number of vehicles. Thus, the results follow a logarithmic curve, which means that for more than 5000 vehicles the difference in terms of the simulation speed is smaller.

The results obtained show that the overhead of the simulations is kept under control. In the worst case, for the evaluated scenario, we obtain a performance of around 10 iterations per second, which shows (if we assume that each iteration computes 1 second of changes —i.e., that it is equivalent to 1 second in the real world—, as it is the default in our simulator) that the simulator could also be used for real-time simulations.

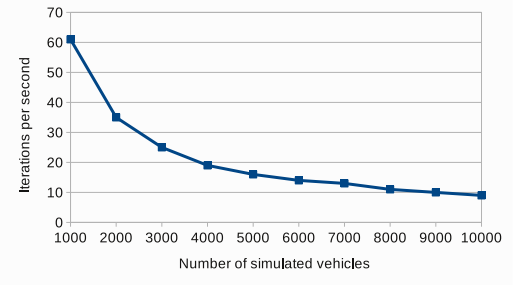


Figure 5.8: Speed of the simulator depending on the number of simulated vehicles

We have also measured the amount of RAM memory needed to simulate a scenario with an increasing number of vehicles. During each simulation, the amount of RAM memory used by the Java process was measured, keeping track of the highest memory usage values during the execution of the simulation (lower values were discarded). As before, each experiment was repeated 10 times and an average delay for all the simulations was computed. The results of the test can be seen in Figure 5.9, which shows how the amount of RAM memory needed increases with the number of vehicles, from approximately 1.5 GB for 1000 vehicles to almost exactly 8 GB for 10000 vehicles, following a linear trend.

Scalability with the Number of Mobile Agents

In this test, we measure the speed of the simulator (in iterations of simulation executed per second) depending on the number of simultaneous mobile agents that are simulated in a given scenario. The goal is thus to assess the scalability of the simulator in terms of the number of mobile agents in the application/system being tested.

As before, the scenario map is that of the city of San Francisco, and the number of vehicles is set to 1000. The number of mobile agents tested varies from 10 to 100 in increments of 10, and for each amount of agents the simulation is executed during 1000 iterations. Each test is repeated 50 times with different initial random vehicle positions, and the average simulation speed is computed from all the obtained values.

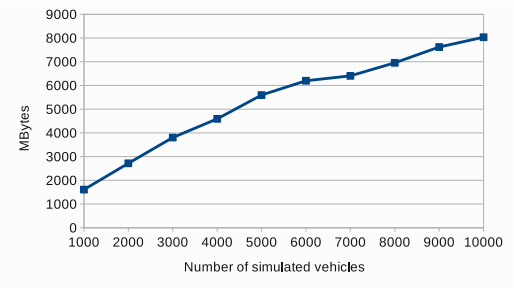


Figure 5.9: Memory usage of the simulator depending on the number of simulated vehicles

The simulated agents behave in a way similar to what was described in Section 3.3, that is, they try to reach an interest area by hopping from one vehicle to another, in such a way that with every hop the agent moves closer to the area. This implies that the mobile agents are constantly evaluating the suitability of all the nearby surrounding vehicles, regarding the possibility to reach the interest area faster than with the vehicle where the agent is currently traveling. Moreover, when a mobile agent is within the interest area, it processes some data stored locally on the vehicles within the area. For the simulation to be more realistic, the presence of buildings that block communication signals are also simulated, for which the simulator must perform additional computations which slow down the simulation speed.

The results of the test can be seen in Figure 5.10, where we can see how the speed of the simulator decreases as the number of simulated mobile agents increases. This decrement is not linear, since the slope is steeper for values of less than 40 agents, and flatter for amounts higher than that number. In any case, what is particularly important is that the overhead of mobile agents is not significant and that the system scales well with the number of mobile agents used.

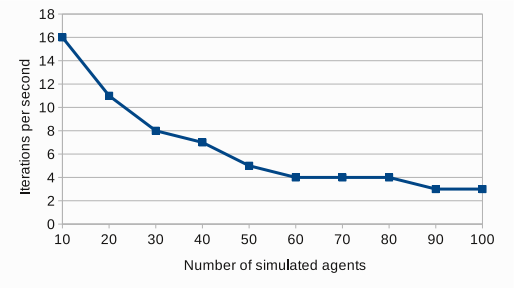


Figure 5.10: Speed of the simulator depending on the number of mobile agents

We have also measured the amount of RAM memory needed to simulate a scenario with an increasing number of mobile agents. During each simulation, the amount of RAM memory used by the Java process was measured, keeping track of the high-

est memory usage values during the execution of the simulation. As before, each experiment was repeated 50 times and an average delay for all the simulations was computed. The results of the test can be seen in Figure 5.11, which shows how the amount of RAM memory needed increases with the number of mobile agents, from approximately 3.3 GB for 10 agents, to about 3.4 GB for 100 agents. The total size may seem high, but it must be remembered that there are not only mobile agents' data in the simulation but also the vehicles, graph map, and others.

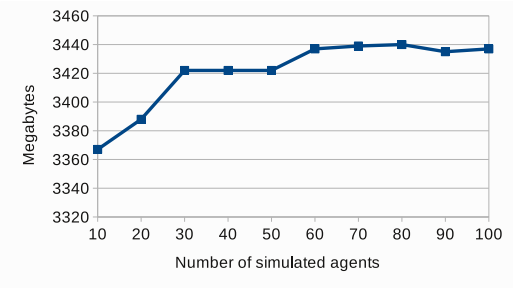


Figure 5.11: Memory usage of the simulator depending on the number of mobile agents

Impact of the Simulation of Buildings on the Performance

This test measures the speed (in iterations of the simulation per second) achieved by the simulator depending on whether buildings are simulated or not, with the goal of finding out if the simulation of buildings implies a significant overhead or not. This study is motivated by the fact that the algorithm used to determine if a communication signal is blocked by buildings needs to compute a number of geometric equations [MFT+13], which can affect the amount of time that every iteration takes to complete.

For this test, we simulate a fixed number of vehicles (5000) and a mobile agent that tries to reach an interest area by hopping from one car to another and that processes some data stored locally in the vehicles within the area (as described in Section 3.3). If the mobile agent infers that a nearby car is a better candidate to travel towards the intended area than the current one that is physically carrying it, it will try to transfer itself to that other vehicle. However, the transmission of the agent could fail if, for example, the target car is behind a building that blocks the signal. Simulating the presence of a mobile agent or some type of communication between vehicles is necessary for this test, as computing the presence or not of buildings is unnecessary in the absence of communications (buildings have an impact only on the performance of communications, as they may block communication signals).

For this test, we execute 1000 iterations using three different maps, with different street layouts that can affect the complexity of computing the impact of buildings, as it is explained in Section 5.2. The cities and their maps are New York (shown

in Figure 5.1(a)), London (shown in Figure 5.2(a)) and San Francisco (shown in Figure 5.3(a)).

For each map, we repeat the simulation with buildings and without buildings and with 50 different random initial vehicle positions. The average of the simulation speeds is computed and shown in Figure 5.12.

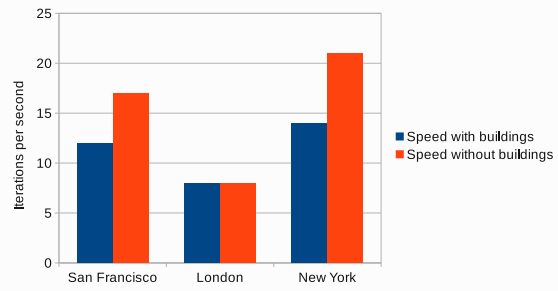


Figure 5.12: Speed of the simulator depending on the simulation or not of buildings

We can see that, in general, the simulation speed is lower when the presence of buildings is simulated, as it was expected, due to the overhead of computing the equations necessary to evaluate the effect of buildings. A notable exception is the case of London, where both values are very similar; this is due to the presence of short and curved streets, which allows the algorithm used for the simulation of buildings to determine at a very early stage that the communication is not possible, without hardly performing additional calculations.

We can also analyze the impact of the city topology on the simulation speed. The performance of simulations in the city with straighter layout (New York) is higher than in the city with a more complicated arrangement (London).

We also notice that, independently of whether buildings are simulated or not, if we compare the results obtained in the test for 5000 vehicles and the case of San Francisco (16 iterations per second, see Figure 5.8) with the equivalent results shown in this experiment (14 iterations per second, see Figure 5.12), the simulation speed is a bit lower for this last one. The reason is that when only moving vehicles are being simulated we avoid the additional overhead due to the execution of the mobile agent, which constantly evaluates the convenience of the vehicle which is currently carrying it (i.e., it keeps itself on alert looking for a better "taxi" to hop to).

5.4 Summary of the Chapter

In this chapter, we described the evaluation methodology followed in the thesis to perform experiments. One of the most important parameters in the simulations is the time needed by a mobile agent to hop between two execution devices, so we first performed a test with real devices to measure this time. After that, we established all the parameters and features of the scenarios where the simulations will take place.

We also presented MAVSIM, the simulator we developed with the ability to perform simulations that involve at the same time vehicles, ad hoc communications, and mobile agents. The reason for developing or own simulator is because existing software in the context of VANET simulations lacks the ability to simulate all those elements, and therefore it could not be used to test and develop data management strategies based on mobile agents.

The simulator also has a number of features that make it a valuable tool, such as its portability and its ability to use maps of real cities and simulate a number of elements such as vehicles, public transportation, roadside units, and mobile agents. It can also simulate other limiting factors typical of VANETs, such as the instability of wireless network links and the blocking of communication signals by buildings. It was built in a modular way, so its features can be easily extended.

Some performance metrics of the simulator were evaluated, such as its memory usage, the simulation speed according to the number of simulated vehicles, the number of simulated agents, and the impact of the simulation or not of buildings.

Chapter 6

Experimental Evaluation

In this chapter, we present different experiments performed to validate that the proposed data management approaches work as expected and to measure their performance in different situations. In Section 6.1, we evaluate different hop strategies. In Section 6.2, we evaluate several factors that have an influence on the basic data management strategy using mobile agents and hop strategies. Finally, in Section 6.3, we evaluate the use of spatial crowdsourcing techniques to enhance the original data management approach. In all the following experiments, we use the MAVSIM simulator and the experimental settings defined in Section 5.2.

6.1 Evaluation of Travel Approaches

In this section, we evaluate the performance of the travel strategies by means of experiments that take place in different cities and also varying other conditions that might have an influence on the strategies' behavior.

6.1.1 Basic Hop Strategies

Given the variety of possible hop strategies, and their dependence on unknown or random factors (e.g., the routes followed by the vehicles), it is not entirely clear which of these (or other) strategies are the best. Moreover, the term *best* may have different meanings depending on which factors are considered the most important one, such as the total time needed by the mobile agent to complete its task, the bandwidth usage, the reliability, etc.

In this section, we perform an evaluation of the basic hop strategies described in Section 4.1 by means of different experiments. By *basic* we mean that the mobile agent travels to the target without asking explicitly to other drivers to be carried to a place, as it occurs with the spatial crowdsourcing approach, which will be the subject of the experiments presented in Section 6.1.2.

In these experiments, the starting point is supposed to be 1 kilometer from the target area. We perform them in the context of the proposed query processing approach described in Section 3.3, and we compare different hop strategies that can be used by a mobile agent when it evaluates the convenience to hop to another vehicle in order to reach the target area. If the target vehicle considered is assumed to be a better carrier (i.e., it has a higher probability to reach the target area) than the vehicle where the agent is currently executing, then the agent will hop to the new vehicle; otherwise, the agent will keep itself in the current vehicle.

Data Collection Time and Number of Hops

The data collection time (time needed to obtain the data to answer a query, which corresponds to steps 1-3 of the proposed approach described in Section 3.3) with the different hop strategies, for a medium vehicle density, can be seen in Figure 6.1. The worst strategy is clearly ANG (in the worst case, with the map of New York, it needs more than seven minutes to collect the data). The best strategy (if we do not consider the optimal strategy) is MAP, taking less than two minutes for data collection in the worst case.

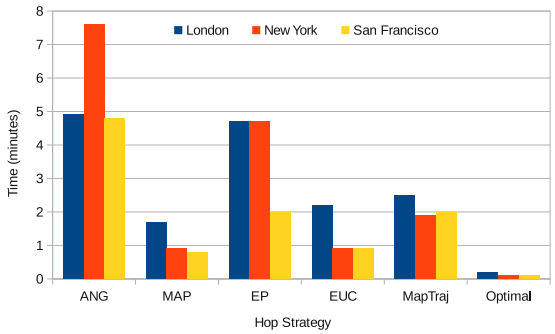


Figure 6.1: Performance of the hop strategies: data collection time

When we consider the optimal strategy, we can see that the mobile agent takes a very small amount of time for data collection. However, as it was explained in Section 4.1, this is an *impossible* strategy, since it computes the mobile agent's ideal actions (e.g., at which moment it should hop to another car or stay in the same one, and which one is the ideal car to select for each hop) using the knowledge of all the positions of the vehicles for the whole simulation (current and future locations).

The reason for the good behavior of MAP and MapTraj is that they have knowledge of the surrounding scenario, and thus the mobile agent can take better decisions when it evaluates if a candidate vehicle will reach the destination sooner or not. The drawbacks are: it is mandatory to have a map of the area and its streets, and that information must be accurate (otherwise, the mobile agent could take incorrect decisions). In the case of MapTraj, it also needs to know the planned trajectories that the vehicles will follow, which is not very realistic since it is unlikely that all the drivers in

the area will use GPS navigators. Moreover, privacy concerns may also be an obstacle to the adoption of that strategy, as it requires the exchange of some information about the expected routes. However, MapTraj is slightly worse that MAP, despite using more information in the decision process. The reason is that with MapTraj the mobile agent requires quite good conditions to hop to another car: it only hops if the route of the target vehicle goes through the target area.

We can also consider the *number of hops* performed by the mobile agent, which can be seen as an indicator of the bandwidth usage. That is, every time the mobile agent hops from one vehicle to another, it must transfer itself through the wireless connection, so a small number of hops implies a low bandwidth usage. However, a smaller number of hops does not imply a smaller data collection time; indeed, traveling by jumping through the wireless medium is expected to be faster than traveling by staying in a moving vehicle. For example, the MAP strategy has slightly better performance than MapTraj but, as it can be seen in Figure 6.2, it needs a higher number of hops than MapTraj.

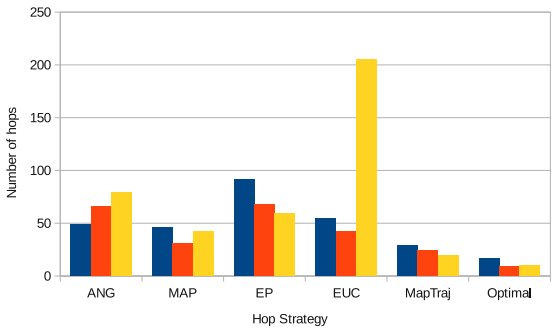


Figure 6.2: Performance of the hop strategies: number of hops

Effective Traveling Speed and Traveling Reliability

Another interesting metric is the *effective traveling speed* of the mobile agent. It can be computed using two key parameters: the existing distance from the origin point to the target area and the time taken by the mobile agent to reach it. In the case of the distance, the straight line between the origin and the spatial destination will be considered the minimum distance possible, since it is highly likely that the mobile agent will need to follow a less-direct path through the streets of the city. Computing the effective traveling speed is better for comparison purposes among different scenarios, since its value is independent of the distance to the target area.

Figure 6.3 shows the results of the simulations. The better (i.e., fastest) strategy is MAP, since it uses maps of the streets in the scenario. Regarding the street layout, the cities with straight and long streets (such as New York, and to a lesser extent San Francisco) clearly lead to better results, whereas the *old city* layout (London) usually implies the worst results in terms of the effective traveling speed. The reason is that

straight streets usually allow more direct routes and the communication signal can also propagate better.

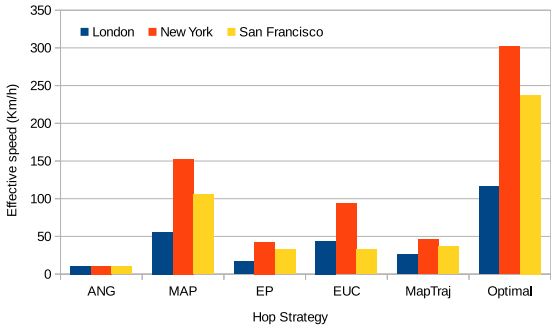


Figure 6.3: Performance of the hop strategies: effective traveling speed

Finally, another interesting parameter that can be evaluated for the different hop strategies is their *traveling reliability*, that we define here as the ability of the mobile agent to reach the target area within a specified time limit. According to Figure 6.4, the best strategies (besides the Optimal strategy) are MAP and MapTraj, since the mobile agent reaches its destination in 100% of the simulations. The worst is again ANG, with slightly more than a 60% of success in the best case scenario for that strategy (London). The other strategies (EP, EUC) have different degrees of reliability, but all of them below 100%.

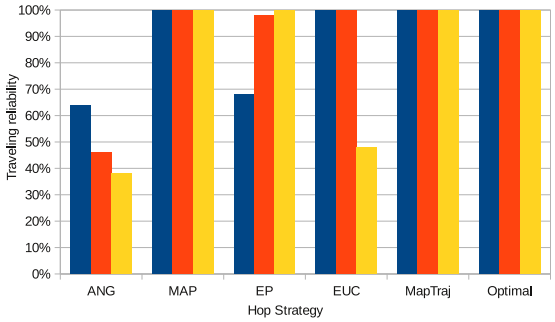


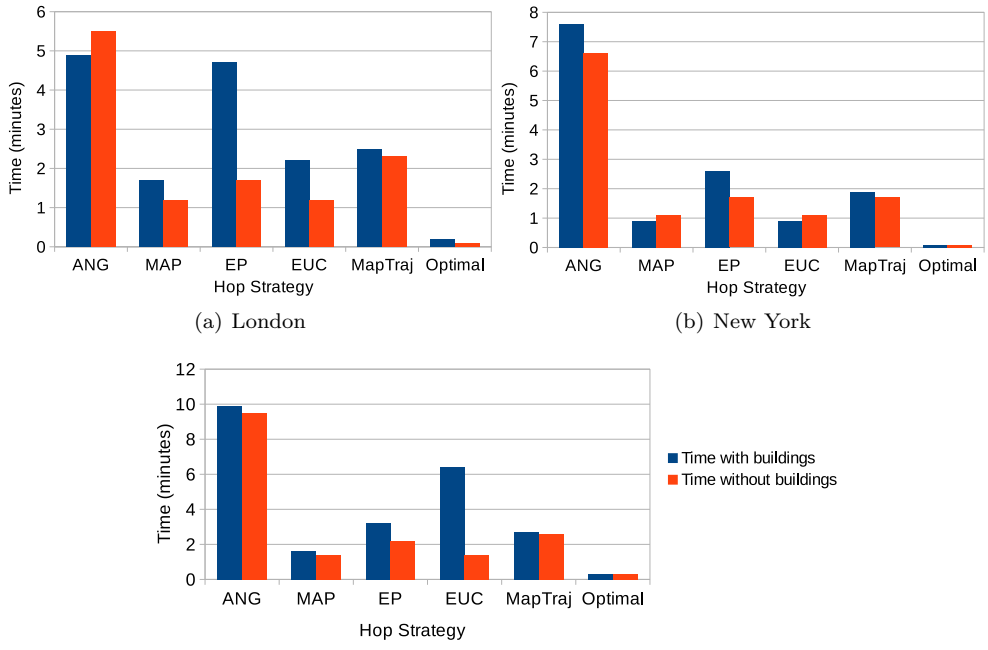
Figure 6.4: Performance of the hop strategies: traveling reliability

Influence of Buildings in Urban Scenarios

The short-range wireless communications used in a VANET operate with radio signals with a very low depth of penetration [SEGD11]. Therefore, buildings and other obstacles can block the signal propagation, making the communication among vehicles possible only if they have a direct line of sight. This means that, in urban scenarios,

there will exist a smaller number of candidate vehicles to be evaluated using the corresponding hop strategy when the mobile agent tries to reach the target area, thus reducing the number of options to find a faster path. It also means that the trajectory followed by the mobile agent will be quite constrained by the street layout, since streets are surrounded by buildings that cannot be crossed by the wireless signals.

In this experiment, we simulate the absence or presence of buildings, using the geometric method described in [MFT+13], to compare their influence on the data collection time for scenarios with medium traffic density. As can be seen in Figure 6.5, the presence of buildings has an effect on the time needed. When no buildings are simulated the whole process is faster, since the agent can hop directly to any other vehicle within the whole communication radius, thus reaching the target area in a more straightforward way. Regarding the behavior of the different hop strategies, when buildings are not simulated the *intelligence* of the most advanced strategies (such as MAP) gets *blurred* and they approach the times obtained with the other strategies (such as EP and EUC), since the previous knowledge of the position of the streets (provided by the digital maps) is less useful when such obstacles are not considered. The opposite occurs when the presence of buildings is simulated. In such a case, the simpler hop strategies (that do not take into account the topology of streets) provide the worst results.



(a) London

(b) New York

(c) San Francisco

Figure 6.5: Performance of the hop strategies: data collection time with and without the simulation of buildings

These experiments show the importance of taking the impact of buildings into account, as we have done by default in our experimental evaluation.

Conclusions

Since MAP is the best strategy when considering globally all the key features studied (data collection time, network overhead in terms of the number of hops, effective traveling speed, and traveling reliability) and besides it does not need to know in advance the routes followed by vehicles (as opposed to MapTraj), we choose it as the default strategy to be used in the upcoming experiments. The fact that it needs a digital road map stored in the vehicles is not considered problematic, given that today such data can be easily obtained from public sources such as *OpenStreetMap*.

We would also like to emphasize that, although the strategies evaluated in this section could be used also as geographic routing strategies in ad hoc networks, the hop strategies are encapsulated here as part of the behavior of the mobile agents (so, it is not part of the routing behavior of nodes) and are used autonomously by an agent to decide potential vehicles to move to. Strategies more sophisticated than the ones evaluated in this section could be developed. However, our work does not focus on geographic routing but on the application of mobile agent technology in vehicular networks.

6.1.2 Hop Strategies with Spatial Crowdsourcing

In this section, we present the experiments that we have performed to test the feasibility of the proposed data management approach in VANETs using mobile agents with spatial crowdsourcing abilities.

In this set of experiments, we evaluate the potential benefits that spatial crowdsourcing can provide. To show this, we compare the proposed spatial crowdsourcing approach with another one that does not use spatial crowdsourcing, by measuring the performance of the process followed by the mobile agent to reach the interest area. The simulations are performed in three cities with different street layouts (defined in Section 5.2): Madrid, Barcelona, and Zaragoza.

Influence of the Size of a Low-Traffic Area

In this first experiment, we perform a test to see how useful the use of spatial crowdsourcing might be to the agent for reaching an area with a small amount of vehicles. For this purpose, we consider an area where the traffic density is low, called *cold area*, and we characterize it as a prolongation of the interest area that is extended a certain length around it (see Figure 6.6). Specifically, from all the trajectories of vehicles randomly created by the simulator for this experiment, a 15% of them cross the cold area, and thus the traffic inside it is lower than in the rest of the scenario.

In this cold area, which has a shape similar to a belt around the interest area, the density of vehicles is much lower than in the rest of the scenario, and therefore the mobile agent will have a lower probability of finding a vehicle traveling towards

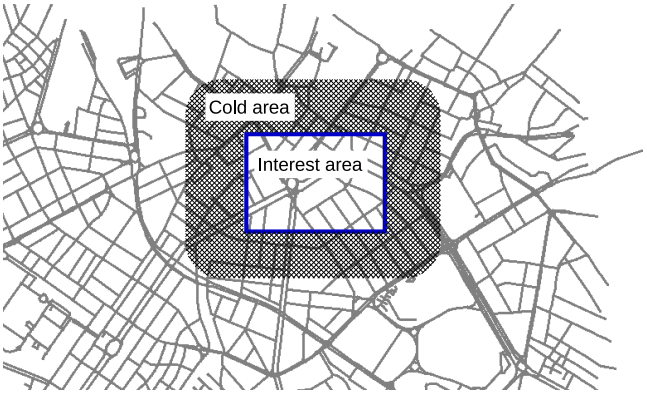
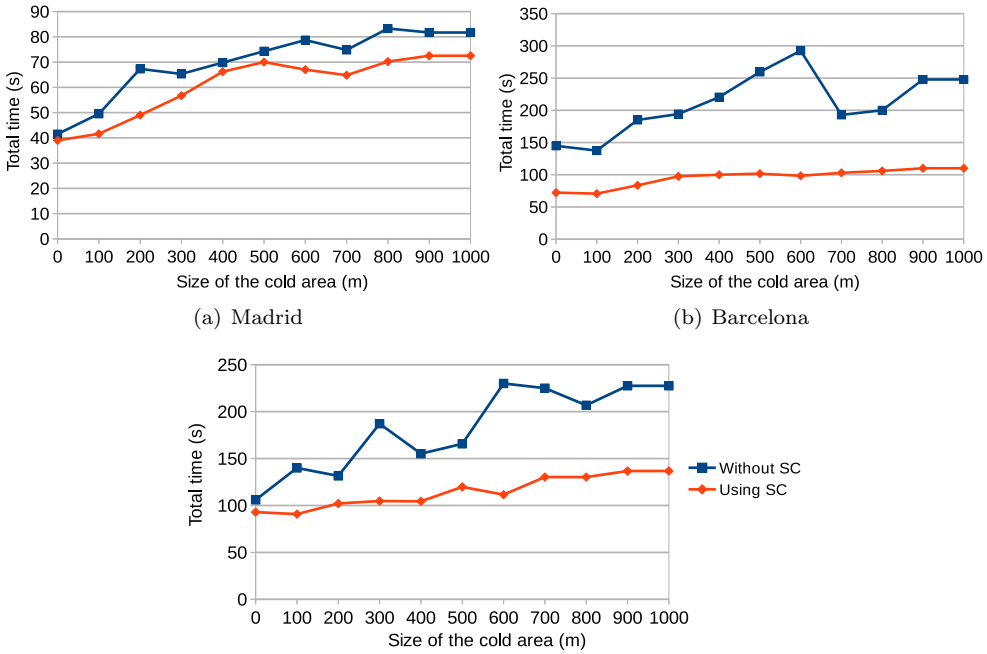


Figure 6.6: Example of a cold area around the interest area

the interest area to use it opportunistically to be carried there, or as an intermediate place to hop while looking for another better candidate. In this situation, the use of spatial crowdsourcing can benefit the query processing considerably, since the mobile agent can find a collaborator willing to alter its original route in order to carry the agent straight to the interest area in exchange of an amount of virtual money.

With this setup, we performed a series of simulations where we varied the size of the cold area (i.e., the length of the low-traffic interest area extension) from 0 meters to 1000 meters. In Figure 6.7, we can see the total time needed by the mobile agent to complete the travel to the interest area in the three cities. We compare the results when spatial crowdsourcing is used (*Using SC*) with a situation where it is never used (*Without SC*). The larger the cold area, the longer it takes for the mobile agent to reach the interest area, as expected. However, when spatial crowdsourcing help is used, the times are considerably reduced. The differences observed in the total times for the three cities are due to the fact that the trajectories of the vehicles and the communications among them vary with the different topologies of the streets.

In Figure 6.8, the total number of hops is shown. When spatial crowdsourcing is used, the number of times that the mobile agent hops (and so the bandwidth needed) is smaller: when the agent finds a collaborating vehicle, it is directly carried towards the interest area and does not need to use any other vehicle to hop to, unless the agent's approaching speed increases above the minimum speed threshold and there is some promising neighboring vehicle; in this experiment, this is unlikely because the carrying vehicle is usually moving through the cold area, which is a low-traffic zone. On the contrary, when no collaborators are used, the mobile agent is constantly looking for a carrier better than the one it is currently traveling on. In this case, the current carrier may follow unpredictable routes, not only carrying the agent nearer the interest area; indeed, it might take a route that travels farther from the interest area. The size of the cold area seems to have little influence on the number of hops. There are two reasons for that. When using spatial crowdsourcing help, once a collaborator is found the mobile agent does not usually need to hop to other vehicles and stays in



(a) Madrid

(b) Barcelona

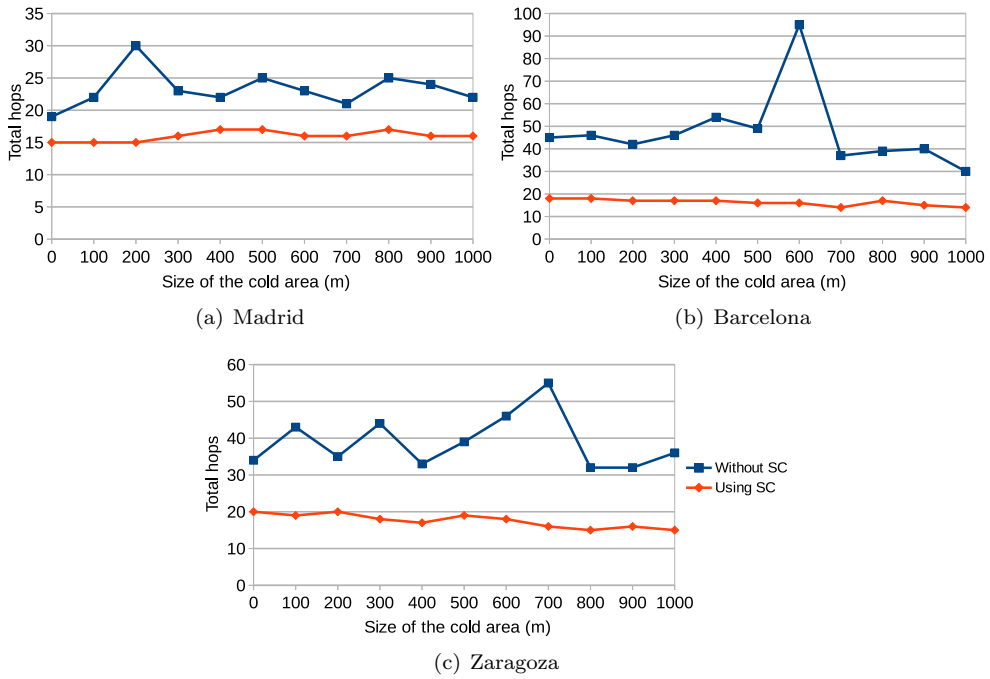
(c) Zaragoza

Figure 6.7: Traveling to the interest area: comparison of the total time needed by using spatial crowdsourcing (SC) or not, varying the size of the cold area

the same vehicle moving towards the interest area. When no spatial crowdsourcing is used, the number of hops varies with the size of the cold area in a more irregular way. This is due to the unpredictability of the routes and locations of the vehicles: the mobile agent must constantly evaluate its environment to decide the most suitable vehicle.

We also evaluated the social cost of using spatial crowdsourcing. In Figure 6.9, we show the *social cost* (in minutes) in the different cities, when the mobile agent uses spatial crowdsourcing (when spatial crowdsourcing is not used, there is no social cost). Regarding the size of the cold area, it has little effect in the social cost, although a slight increase with its size is observed.

To summarize the conclusions regarding the influence of the size of the cold area, when it is larger the total time needed by the mobile agent to reach the interest area grows, although it is smaller when spatial crowdsourcing is used. The bandwidth usage remains similar, but when no spatial crowdsourcing is used it is higher and more irregular. Finally, the social cost, which only applies when using spatial crowdsourcing, is only slightly influenced by the size of the cold area.



(a) Madrid

(b) Barcelona

(c) Zaragoza

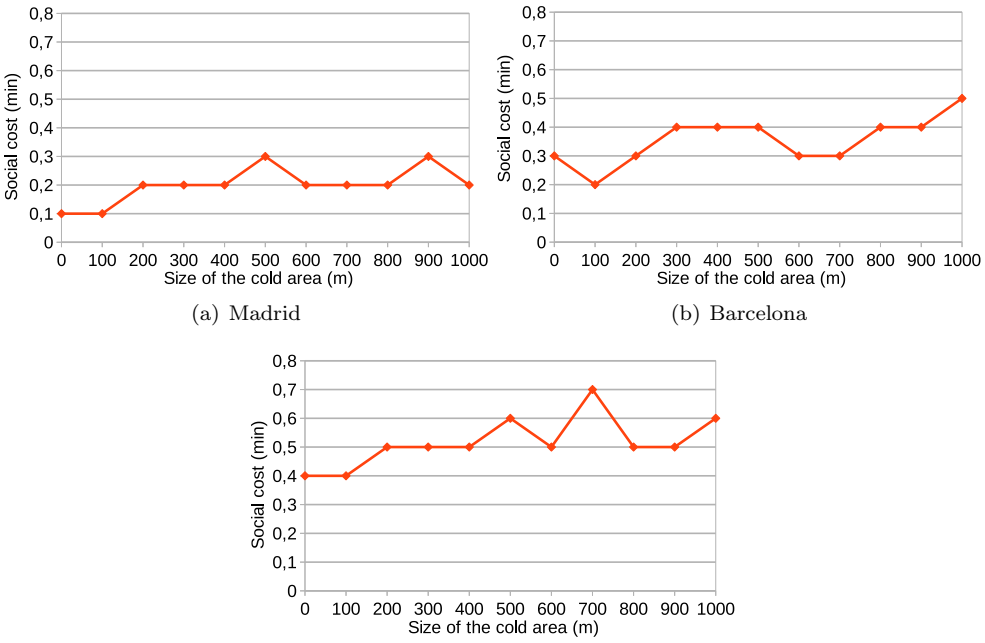
Figure 6.8: Traveling to the interest area: comparison of the total number of hops needed by using spatial crowdsourcing (SC) or not, varying the size of the cold area

Comparison Varying the Initial Distance to the Interest Area

The previous experiment indicates that spatial crowdsourcing can be very useful in scenarios with areas where the density of vehicles is not high. Now, we will perform another experiment with higher traffic density in the whole scenario, in order to see if the proposed approach also behaves well when the number of potential carriers for agents increases.

In this experiment, there is no cold area and there is a uniform density of vehicles in the scenario, which is set to 100 vehicles/km^2 (a medium density). We vary the initial distance from the point where the mobile agent starts its execution to the limit of the interest area, and we evaluate the potential benefits of spatial crowdsourcing for different values of such a parameter.

In Figure 6.10, we can see how the total time to reach the interest area varies with the initial distance. In general, the times are slightly higher when no spatial crowdsourcing is used for the smaller to medium initial distances, whereas for higher distances the differences are near 0 and practically the same. A minor exception can be found in Figure 6.10(c), for the city of Zaragoza, in a couple of cases where for some distance values (e.g., 1750 and 2250 meters) the times are slightly higher when spatial crowdsourcing is used, but this difference is quite small (about 3 seconds, from



(a) Madrid

(b) Barcelona

(c) Zaragoza

Figure 6.9: Traveling to the interest area: collaborators' social cost

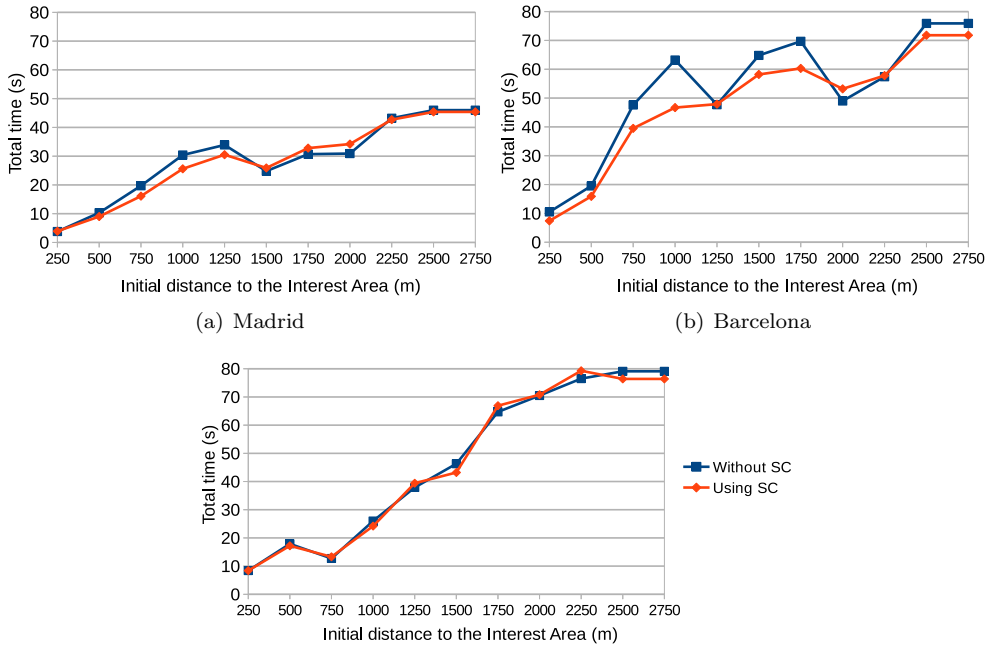
a total of around 67 seconds and 80 seconds, respectively).

Regarding the number of hops that the mobile agent performs to reach the interest area, shown in Figure 6.11, the figures for the three cities have a similar shape: the number of hops is slightly higher when no spatial crowdsourcing is used, and it grows linearly with the distance to the interest area.

As a summary, the time needed to reach the interest area grows with the distance. For short and medium distances, the spatial crowdsourcing option is better, but for longer distances the difference between both options is smaller. Regarding the total number of hops needed, the initial distance has a linear effect and in all the cities the crowdsourcing option takes less hops than the one without spatial crowdsourcing.

Comparison Varying the Density of Vehicles

The density of vehicles is very important for the performance of the agent, since it can be difficult to reach the interest area if the number of vehicles that can be used as a physical transport or as an intermediate relay is low. In this experiment, we want to know in which circumstances the use of spatial crowdsourcing helps the mobile agent to reach the interest area. In this scenario, there is no cold area and in all the zone there is a uniform density of vehicles, which varies from 10 vehicles/km^2 (a very low density) to 175 vehicles/km^2 (a high density).



(a) Madrid

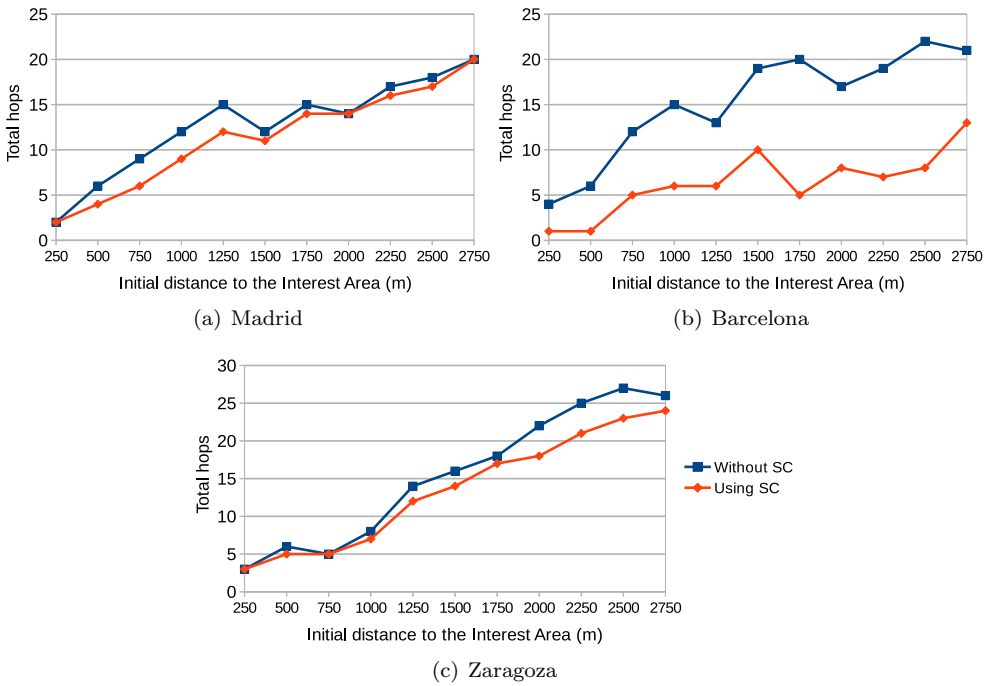
(b) Barcelona

(c) Zaragoza

Figure 6.10: Traveling to the interest area: comparison of the total time needed by using spatial crowdsourcing (SC) or not, varying the initial distance (no cold area)

In Figure 6.12, we can see how the total time to reach the interest area varies with the density of vehicles. In all the cities, the time is lower when spatial crowdsourcing is used than when it is not. The differences are higher with low vehicle density values, especially for the lowest value of 10 vehicles/km^2. For density values higher than 30 to 45 vehicles/km^2 the difference between using spatial crowdsourcing or not is practically negligible. The reason is that, when spatial crowdsourcing is not used, the mobile agent must go to the interest area by hopping from one vehicle to another that seems more promising to reach the target sooner, and finding candidates is more difficult when the density of vehicles (and therefore, its total amount) is lower.

Figure 6.13 shows that, as the vehicular density grows, the number of hops that the mobile agent needs to perform to reach the interest area decreases when spatial crowdsourcing is not used. On the other hand, when it is used, the number of hops increases with the density of traffic until around 45 vehicles/km^2, when the number of hops remains stable with small variations for higher density values. In all the cases, the number of hops when spatial crowdsourcing is used is smaller than when it is not used, although the difference decreases as the vehicular density grows. This is due to the fact that, once the mobile agent takes a collaborator, it stays in there and does not leave it until the interest area is reached, being unnecessary to perform more hops



(a) Madrid

(b) Barcelona

(c) Zaragoza

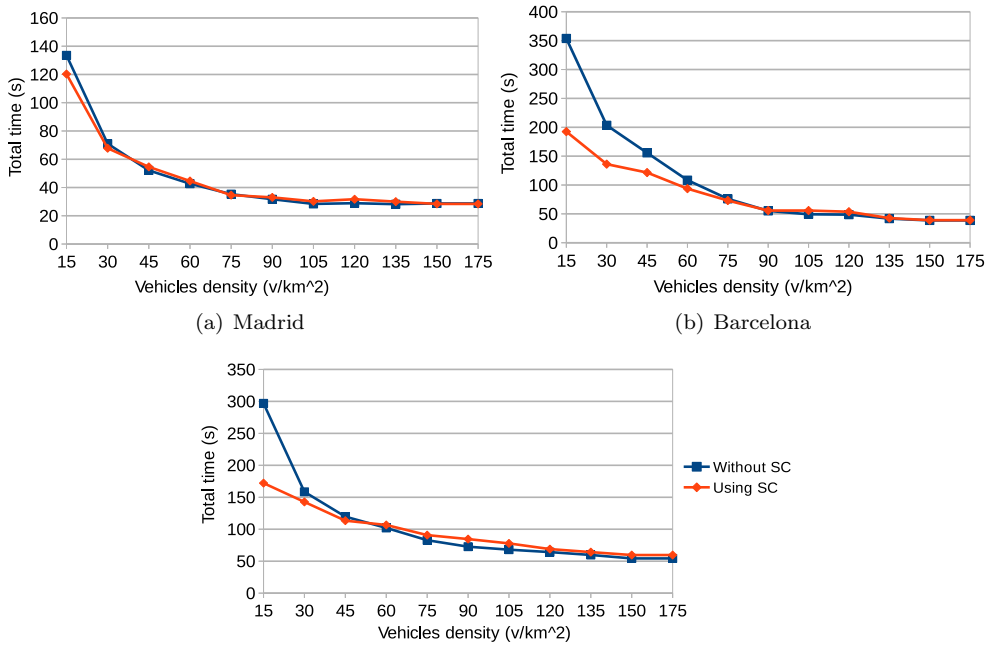
Figure 6.11: Traveling to the interest area: comparison of the total number of hops needed by using spatial crowdsourcing (SC) or not, varying the initial distance (no cold area)

to other vehicles.

Summing up, we conclude that the spatial crowdsourcing approach is particularly beneficial in terms of time and bandwidth usage in scenarios with a low density of vehicles. Moreover, when the traffic density is higher, the use of spatial crowdsourcing does not harm either the performance. Indeed, based on the minimum speed threshold, the mobile agent can choose to hop among the vehicles when it considers that paying a collaborator is not needed.

6.2 Evaluation of the Basic Query Processing Approach

In the previous section, we evaluated different travel approaches that a mobile agent can use to reach the interest area in a fast and efficient way. In the following, we evaluate other factors over the whole basic query processing approach (i.e., involving all the steps). Again, by *basic* we mean that the mobile agent does not ask for help of collaborating human drivers to be carried nearer the target area or any other places. The chosen default hop strategy for all the next experiments is MAP, and the mobile



(a) Madrid

(b) Barcelona

(c) Zaragoza

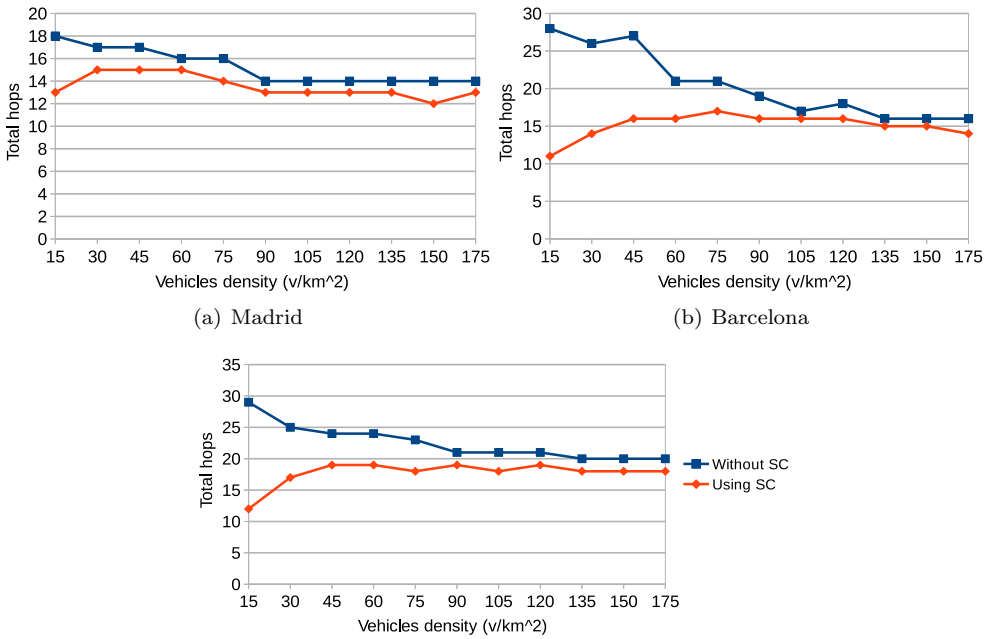
Figure 6.12: Traveling to the interest area: comparison of the total time needed by using spatial crowdsourcing (SC) or not, varying the vehicle density (no cold area)

agent's starting point is 1 kilometer from the target area.

6.2.1 Influence of the Uncertainty of the Location of the Query Originator

As commented in Section 3.3, routing the results back to the query originator in vehicular networks using only short-range wireless communications could still be considered an open problem in the literature, and therefore, in this thesis, we have adopted a simple approach based on the availability of an estimation of the location of the query originator vehicle. In this section, we present an experiment to evaluate the impact that an imprecise location estimation of the query originator may have on the performance of the whole query processing (steps 1-4 of the proposed approach, described in Section 3.3). In this experiment, if the mobile agent does not find the query originator at the expected location, it will try to find it by moving randomly in the surroundings of that location.

Figure 6.14(a) shows the total time needed to finish the query processing depending on the existing location uncertainty regarding the query originator (e.g., a location uncertainty of 500 meters means that the actual location can be anywhere in a circu-



(a) Madrid

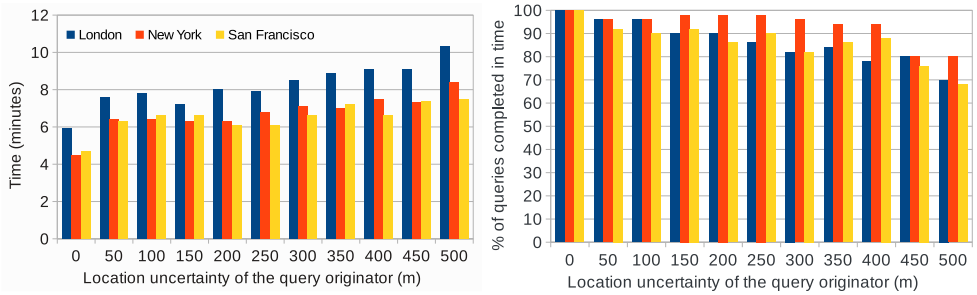
(b) Barcelona

(c) Zaragoza

Figure 6.13: Traveling to the interest area: comparison of the total number of hops needed by using spatial crowdsourcing (SC) or not, varying the vehicle density (no cold area)

lar area of radius 500 meters). The query processing time increases with the location uncertainty, as expected, since the agent needs more time to find the query originator vehicle. The average values are computed considering only the queries that are processed within 1000 seconds. Therefore, it is also interesting to look at Figure 6.14(b), which shows the number of queries that are completed in time. For example, with the San Francisco layout and a location uncertainty of 500 meters, which is the most challenging scenario shown in the figure, the ratio of success in processing a query is below 70%.

In general, a high reliability is obtained if the location uncertainty of the query originator is about 250 meters or lower. Moreover, it would be possible to considerably increase the reliability when the location uncertainty is high, for example, by improving the simple random-search approach tested here to try to locate the query originator vehicle, by combining the use of pure ad hoc communications with wide-area communications (e.g., 3G/4G) or vehicle-to-infrastructure communications (use of fixed support nodes on the roads, usually called roadside units) when they are available and the extra economic cost that their use implies is affordable [IDTL15], or by using mailboxes to store the query results at fixed locations to be retrieved by the query originator [DMIH11].



(a) Query processing time (b) Queries processed

Figure 6.14: Influence of the location uncertainty of the query originator on the query processing

6.2.2 Influence of the Relevance of Information

In the scenario that we are simulating, once the mobile agent arrives at the target area, it must hop among the vehicles within, looking for relevant information and gathering a certain number of partial solutions before being able to solve the query and return the result. We assume that not all the vehicles contain data the mobile agent is interested in, so the agent will have to visit one vehicle after another until it finds one containing relevant information. Then, the mobile agent will process the data to obtain a part of the query result, and the process will continue. The higher the number of vehicles that contain relevant data, the faster the agent will complete the process and the smaller the number of vehicles that will have to be visited.

In this experiment, we test different values of the existing *relevance of information* for the query (defined as the percentage of vehicles that contains relevant data to solve the query) in the three city maps. Figure 6.15 shows that, as expected, when the relevance of the information present on the vehicles is low, the time the agent takes to solve the query may have relatively-high values. As the value of the relevance grows, the time decreases, and for values of relevance higher than the 50% the improvement obtained with progressively-higher relevance values reduces to almost zero and remains under five minutes in the worst case.

As a conclusion, this experiment shows that a relatively-low value of relevant data (e.g., about 40%) is enough to find the query solution *on the fly* in an acceptable time, considering that the data are extracted directly from their sources in a distributed and ad hoc way.

6.2.3 Influence of the Vehicle Density

In this experiment, we evaluate the performance and reliability of different values of vehicles density. For our test scenarios, we use the following values for vehicle traffic density (friendly labels are indicated in brackets, with the purpose of facilitating reading): 5 v/km^2 (*nearly-absent*), 12 v/km^2 (*extremely low*), 25 v/km^2 (*very low*),

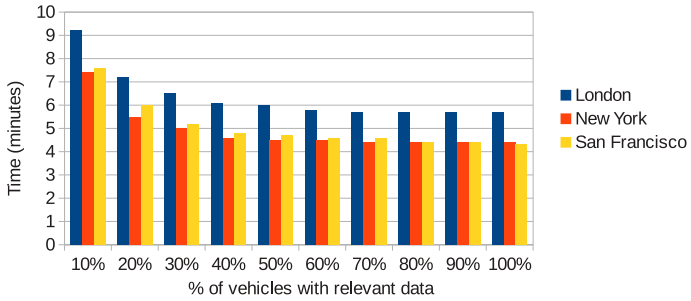


Figure 6.15: Influence of the amount of vehicles with relevant data on the query processing

50 v/km^2 (*low*), 100 v/km^2 (*medium*), and 200 v/km^2 (*high*). These density values used in the simulation are inspired by those used in works such as [MFT+13, BGF+13].

Figure 6.16 shows that, in general, the query processing time decreases with higher traffic density values. This is due to the fact that, when the number of vehicles the mobile agent can jump to increases, the chances to choose appropriate vehicles for transportation improve. Of course, the benefits obtained by adding more vehicles are insignificant when the traffic density is already high; thus, as an example, we can observe in the figure that the difference in the performance between the cases of medium and high density is close to zero.

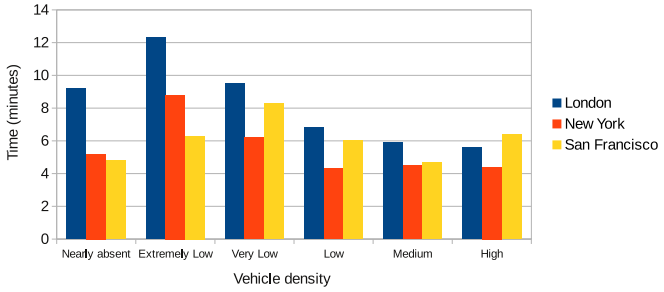


Figure 6.16: Influence of the vehicle density on the query processing: query processing time

By looking only at Figure 6.16, it might seem that a nearly-absent traffic density achieves a good performance. However, it must be highlighted that the figure is only showing the average processing times of the queries completed, which in that case are a small percentage of the total number of queries submitted. So, in Figure 6.17 we can see the reliability of the whole process in terms of the percentage of queries finished within a timeout of 1000 seconds. It can be seen that the approach based on mobile agents does not require a high traffic density to perform well. So, all the queries finish in time even with *very low* density values, except in the case of London (where the percentage of completed queries is 92%). The percentage of success is quite high even

with *extremely low* traffic values (above 90% for the mixed and squared city layouts), except in the case of the *old city* layout where the percentage drops to only 30% of finished queries.

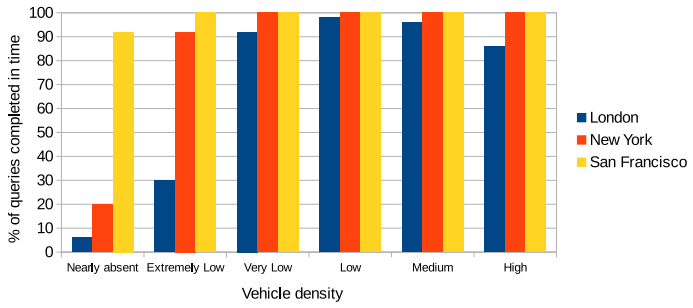


Figure 6.17: Influence of the vehicle density on the query processing: queries processed

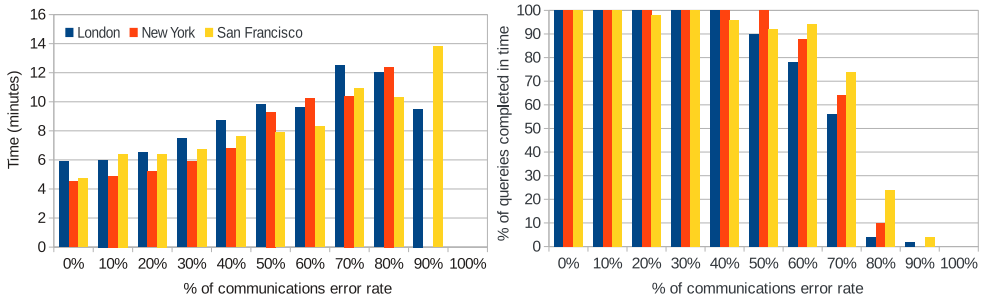
As all the queries are completed with *low* traffic (and higher traffic densities) and most of them are also completed with *very low* traffic, we can conclude that the proposal performs well in most scenarios, even if the traffic density is low.

6.2.4 Influence of Communication Errors

In this section, we evaluate experimentally the impact of unreliable wireless communications. For that purpose, we simulate different *communication error rates*, which indicate the probability that a communication (e.g., an agent jumping from one car to another) fails. If the communication fails when an agent tries to jump to another car, the movement fails and it will need to be re-tried; moreover, due to the constant movement of the vehicles, the intended target car might move out of range and become unreachable when re-attempting. The simulation of this error rate is in addition to factors such as the influence of buildings that act as obstacles (evaluated in Section 6.1.1); so, a building may block a communication even if communication error rates are not simulated.

We vary the communication error rate from 0% (no extra failures simulated) to 100% (all the communications fail always). Figure 6.18(a) shows the total time required to complete the query processing. The query processing time increases with the communication error rate, as expected. Again, it must be noted that these values are computed taking into account only the simulations that end within a timeout of 1000 seconds. Therefore, Figure 6.18(a) has to be analyzed by considering also Figure 6.18(b), which shows that starting with an error rate of around 40% the ratio of queries processed starts to decrease, dropping significantly when the error rate is 80% and reaching near 0% for higher rates. Figure 6.18(a) shows no value for New York when the error rate is 90% and no value at all for an error rate of 100%, as in those cases no query finishes successfully in that scenario before the timeout.

These experiments show that, as expected, the query processing approach performs better when the wireless communication medium is reliable, but that the ap-



(a) Query processing time (b) Queries processed

Figure 6.18: Impact of communication failures on the query processing

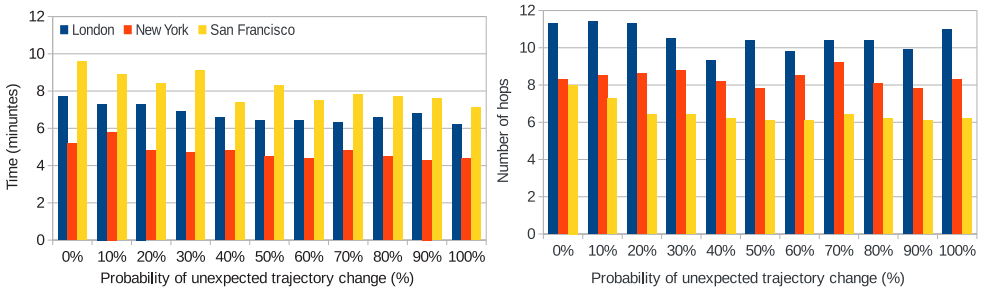
proach is also flexible enough to adapt to communication failures. Mobile agents are able to react to these failures and the errors need to be quite high to affect the reliability of the query processing.

6.2.5 Influence of Unexpected Changes in Routes

In this experiment, we analyze the reliability and performance of the query processing approach when vehicles change their route unexpectedly. We focus on the *MapTraj* hop strategy, as it is the only strategy that benefits from the assumption that the expected trajectories of the vehicles can be obtained, and we simulate random changes in the intended trajectory. To evaluate a worst case scenario, we simulate that a vehicle can change its trajectory with a certain probability (*trajectory changing rate*) as soon as the agent jumps to it. Therefore, an agent may decide to hop to a car because the car's trajectory is promising and immediately after taking that decision the situation may change (the car may modify its trajectory and therefore may not be a suitable vehicle to reach the target area anymore). In other words, the agent may take a decision based on information that quickly becomes obsolete.

Figure 6.19(a) shows the total time needed to process a query when the trajectory changing rate varies from 0% (none of the vehicles change their intended trajectory upon the arrival of the mobile agent) to 100% (all the vehicles change their trajectory), and Figure 6.19(b) shows the number of hops performed by the mobile agent during steps 1 and 4 of the query processing approach (traveling to the target area and returning to the query originator). It can be seen that unexpected changes of the trajectories do not affect the performance significantly, as mobile agents continuously re-evaluate the current situation and jump to another car if needed; so, they can quickly fix incorrect decisions when the information they use to decide becomes incorrect. Indeed, the mobile agent is constantly evaluating the situation and will jump to a different car if it is more promising, independently of whether its current vehicle has changed its trajectory or not; this is the reason why Figure 6.19(b) does not show a significant increase in the number of hops when the trajectories change

more frequently. It should also be noted that when the trajectory changing rate is 100% the *MapTraj* hop strategy becomes similar in practice to the *MAP* strategy.



(a) Query processing time

(b) Number of hops (steps 1 and 4)

Figure 6.19: Influence of trajectory changes on the query processing for the MapTraj strategy

6.2.6 Influence of the Cloning Strategies

In this section, we evaluate the potential benefits of using clones and the influence of the specific number of clones created during the query processing described in Section 3.3. With that purpose, in the following experiments, we set the value of *minPercentData* (minimum acceptable percentage of data that must be retrieved) at 75% and we evaluate two different strategies.

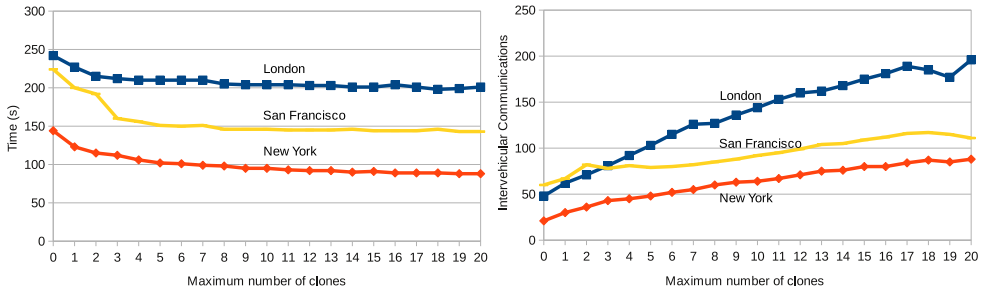
Cloning Strategy 1: Clones for Data Collection

In the first experiment, we vary the number of clones created for data collection from 0 (no use of clones) to 20, and set the percentage of vehicles with relevant data to 50%. Following Algorithm 4 in Section 3.3, the clones are created only in the data collection phase if they are needed to increase the observed collection rate.

In Figure 6.20(a), we show the query processing time required to get a first answer to the query depending on the number of clones. As expected, increasing the number of clones contributes to decreasing the time required to get the first answer, but only to a certain extent. Besides, increasing the number of clones also leads to a higher overhead in the network, as shown in Figure 6.20(b). Thus, above a certain number of clones there is little improvement and the overhead introduced does not pay off.

Cloning Strategy 2: Clones for the Whole Process

In the second experiment, the mobile agent creates copies of itself in the first step of the process (all at once), and they hop randomly among the nearby vehicles for a certain interval of time. Afterwards, the original agent and its clones start using the



(a) Query processing time (b) Network overhead

Figure 6.20: Performance of the Cloning Strategy 1: use of clones for data collection

predefined hop strategy (MAP) to travel to the target area, executing individually the algorithms described in Section 3.3, and gathering the collected data in the origin.

The question now is how long the clones should be hopping randomly at the beginning of the process before starting traveling to the target area. If this amount of time is too small, then they will not spread far enough from the initial point and the number of alternative routes found will be low, since all the agent's copies could behave similarly. On the other hand, if it is too large, then they might travel to places too far from the target area. Therefore, we have first performed an experiment to evaluate the query processing time for different values of the *Interval of Random Hopping* (IRH). Figure 6.21, where the number of clones to use is set to 10, shows that an IRH value of five leads to the overall smallest processing time.

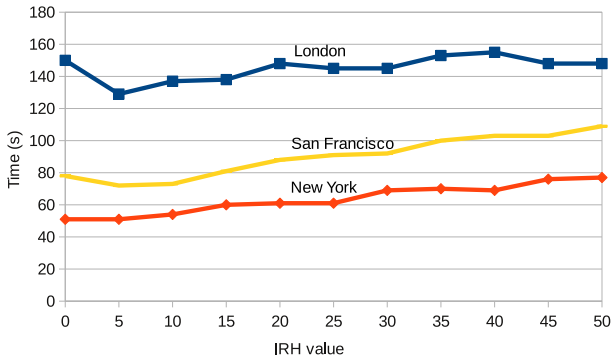
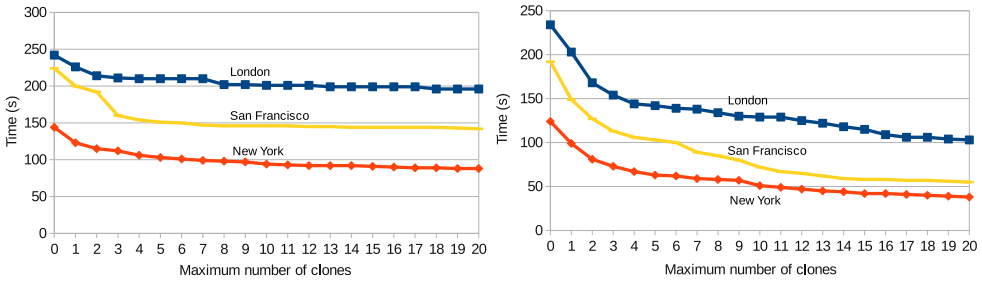


Figure 6.21: Performance of the Cloning Strategy 2: query processing time for different values of IRH in different scenarios

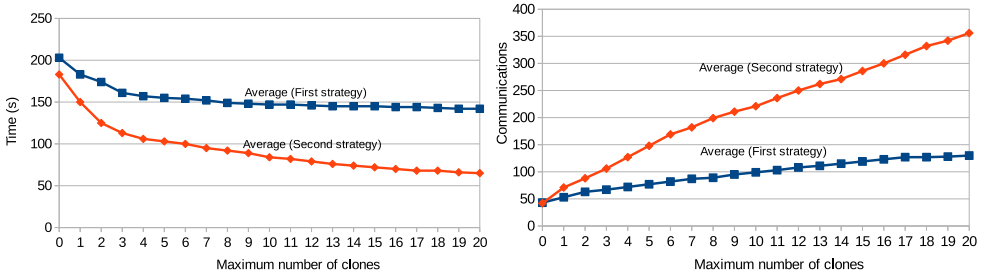
Comparison of Both Cloning Strategies

Figures 6.22(a) and 6.22(b) show the total time spent to process the query using both mobile agent cloning strategies in the different city scenarios. As we can see in the figures, the use of the maximum number of clones in the beginning of the process (second strategy, with IRH=5) minimizes the total time needed to process the query. This can be seen more clearly in Figure 6.23(a), that compares the average of the processing times. However, as shown in Figure 6.23(b), the network overhead for the second cloning strategy is higher than when using the first one, since all the agents are created at the beginning of the process.



(a) First cloning strategy (b) Second cloning strategy with IRH=5

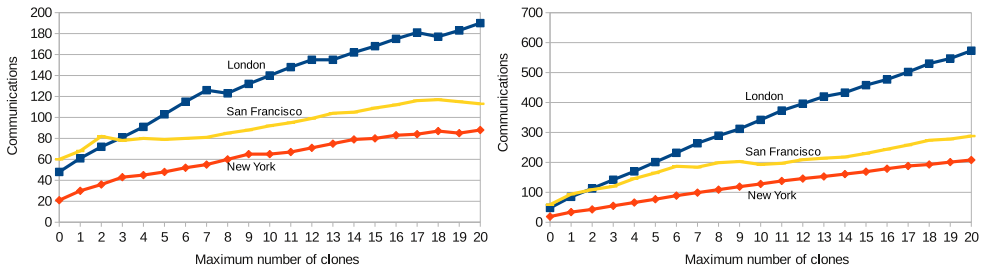
Figure 6.22: Time to solve the query with two cloning strategies in different scenarios



(a) Query processing time (b) Network overhead

Figure 6.23: Comparison of two different cloning strategies

Regarding the city topology, as Figures 6.24(a) and 6.24(b) show, the benefits of using clones is similar for the three types of street layouts considered. However, the number of intervehicular communications grows with the number of clones in a more sharply way in the *old city* layout. This is due to the difficulty for the mobile agents to travel long distances by hopping to other vehicles, as the proximity of buildings in narrow and curved streets can block the signal propagation. This circumstance leads the mobile agents to hop more frequently, thus increasing the network usage.



(a) First cloning strategy (b) Second cloning strategy with IRH=5

Figure 6.24: Network overhead using two cloning strategies in different scenarios

6.3 Evaluation of the Spatial Crowdsourcing Query Processing Approach

In the this set of experiments, we analyze different parameters related to the spatial crowdsourcing approach (where the mobile agent asks to other users for they collaboration in exchange of a compensation), and the impact they have on the performance of the whole query solving process. We study the effect of parameters such as the minimum speed threshold used by a mobile agent to decide when collaborators should be used, the percentage of collaborators, the existing traffic density, and the required minimum stay time of mobile agents in the vehicles. In these experiments, the mobile agent starts at a distance of 2 kilometers from the interest area, and follows the complete query solving process described in Section 3.3: it first travels to the interest area and, once the interest area is reached, it visits every cell into which it is divided to read the required data by using the available sensors in its carrying vehicle. When the mobile agent has read all the sensors required (or the maximum timeout expires), it returns to its origin carrying with it the data collected.

6.3.1 Minimum Approaching Speed Threshold Determination

As explained in Section 4.2, the travel to the target area, as well as the return to the origin with the results, is performed by the mobile agent thanks to a combination of two techniques: initially, the mobile agent starts hopping from one vehicle to another one that it considers to be a more promising carrier to reach the target area and, additionally, the agent may also use spatial crowdsourcing to be physically carried nearer the target area by a collaborating vehicle. To decide whether using one technique or the other at any time, a minimum speed threshold is set: if the mobile agent approaches the destination area with a speed smaller than this threshold value, it will ask for help to nearby collaborating vehicles to be carried nearer the destination area faster, until the approaching speed raises again above the threshold. Otherwise, the mobile agent will keep hopping from one vehicle to another, as described before.

It is important to establish the minimum speed threshold carefully. If it is too

high, the mobile agent will likely look for collaborators too early, and therefore it will spend too much virtual money needlessly. On the other hand, if the value is too low, then the agent will ask for help later, but then it may be already too far from the interest area and thus take too much time to reach the destination. In this experiment, we test different values for the minimum speed threshold, in the range from 0 km/h to 50 km/h (usual maximum speed within a city). The ratio of collaborating vehicles is set to 50%, taken as an intermediate value, and there is no cold area. As we want to see the benefits of using spatial crowdsourcing also within the interest area, the probability that the trajectory of a given vehicle will travel inside the interest area is 10%, as this implies a low-traffic situation where the difference between using spatial crowdsourcing or not using it during the data collection phase can be more relevant.

We perform the experiments using two versions of the mobile agent (*SCCA* and *PHCA*), with different behaviors exhibited once the agent reaches the target area (see Section 4.2).

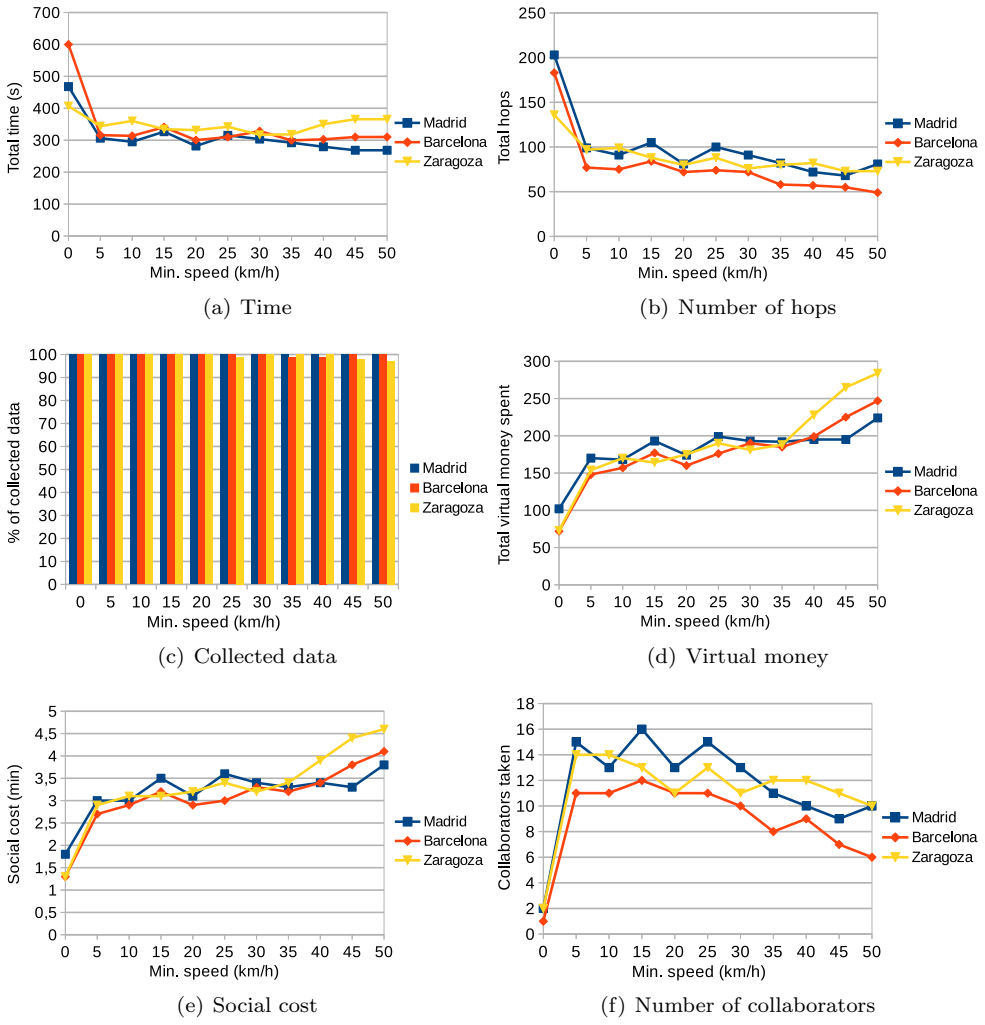
Results with SCCA

Figure 6.25(a) shows the time required for completing the query solving process in the three cities, by using the SCCA variant of the mobile agent, as described before. The extreme case when the minimum speed threshold equals 0 km/h means that the condition for using spatial crowdsourcing is *never* met, and therefore the agent must reach the target area only by hopping from one vehicle to another (spatial crowdsourcing is not used at all). However, once the target area is reached, the SCCA mobile agent *always* uses spatial crowdsourcing to travel to all the cells, with independence of the minimum approaching speed threshold (the agent is already within the target area, and so the concept of approaching speed does not apply). As a result, in general, in all the maps, the total time required is higher with a value of 0 km/h than with other higher values, and the variations for higher minimum speed thresholds are quite small.

In Figure 6.25(b), we can observe that the number of hops needed by the mobile agent to complete the process is much higher for the minimum speed threshold value of 0 km/h (when no spatial crowdsourcing is used), and for values higher than 5 km/h the number of hops decreases. The reason is that when no spatial crowdsourcing is used the mobile agent needs to constantly hop from one vehicle to another until it reaches the interest area.

In Figure 6.25(c), the percentage of collected data is shown. The result for all the minimum speed threshold values tested are 100% (i.e., all the requested data are collected), but it must be taken into account that for the value of 0 km/h (i.e., when no spatial crowdsourcing is used) the reliability is below 90%, so the collected data percentage shown in Figure 6.25(c) refers only to the simulations ended within the simulation time limit.

In Figure 6.25(d), the amount of virtual money spent by the mobile agent is shown. As explained in Section 4.2, in our experiments, we assume that the agent pays one unit of virtual money for every second it is carried by a collaborating vehicle. When the minimum speed threshold value is 0 km/h, the amount paid is very low, since the



(a) Time (b) Number of hops

(c) Collected data (d) Virtual money

(e) Social cost (f) Number of collaborators

Figure 6.25: Query solving process: results with SCCA

collaborators are only used when the mobile agent collects data when traveling to the target area cells within the interest area, but not for reaching the area when returning to it when the agent leaves it unintentionally, or for returning to the monitoring origin. For higher values of the minimum speed threshold, the virtual money spent increases, being the best overall value around 5 km/h.

Regarding the social cost, shown in Figure 6.25(e), the SCCA agent exhibit a similar behavior in the three cities: when the minimum speed threshold value is 0 km/h, the cost is very low because spatial crowdsourcing is used only when the mobile agent

is inside the target area collecting data. For higher speed threshold values, the social cost also increases, since the agent makes use of spatial crowdsourcing more frequently. Note that this graphic is very similar to the one of the virtual money spent (though not exactly the same), which means that the established mechanism to compensate the collaborators (paying them according to their invested time) is quite fair, since the compensation paid is proportional to the costs incurred by the collaborators.

Finally, Figure 6.25(f) shows the number of collaborating vehicles that the SCCA agent needs to complete the process. For the minimum speed threshold value of 0 km/h, the number of collaborators is very low, because they are only used in the data collection phase, but not for traveling to/from the target area. For the next minimum speed threshold value (5 km/h), the number of collaborators grows to between 11 and 15, but then it decreases as the minimum speed threshold value increases, until 6 to 10 collaborators for 50 km/h. This is interesting, as compared to the results of the total time, which has a low variability for minimum approaching speed threshold values in the range of 5 to 50 km/h (see Figure 6.25(a)). When the minimum approaching speed is set to a low value (e.g., 5 km/h), the mobile agent uses more collaborators but for a shorter time. On the other hand, when the minimum approaching speed is set to high values (e.g., 45 km/h), less collaborators are used by the mobile agent, but it travels physically aboard them for a longer time. The reason is the following: when the agent uses spatial crowdsourcing it will stay in the collaborating vehicle until the approaching speed exceeds the minimum speed threshold, and shortly after that moment the agent will leave the collaborator if it finds a more promising vehicle, and will continue approaching the target area, hopping from one vehicle to another by using the established hop strategy. For low values of the minimum speed threshold, it will be easier to reach the threshold value, and the agent will leave the collaborator sooner than for high values. As the minimum approaching speed threshold grows, the mobile agent will travel longer in a collaborator, and will approach nearer the target area, so it will also need a smaller number of collaborators to reach it.

It should be noted that the performance results of unended simulations are not available, and so another parameter tested in the experiment is the reliability of the query solving. In Table 6.1, we can observe that the reliability of the process is between 56% and 90% when then minimum speed threshold equals 0 km/h (i.e., when no spatial crowdsourcing is used). For higher values of the minimum speed threshold, the reliability reaches 100% (with one single exception for the minimum speed threshold value of 40 km/h and the city of Zaragoza, with a reliability of 98%), meaning that the simulations end within the time limit. That is, the use of spatial crowdsourcing for values above or equal to 5 km/h increases the reliability of the whole query solving process considerably.

minspeed (km/h)	0	5	10	15	20	25	30	35	40	45	50
Madrid (% of ended simulations)	56	100	100	100	100	100	100	100	100	100	100
Barcelona (% of ended simulations)	90	100	100	100	100	100	100	100	100	100	100
Zaragoza (% of ended simulations)	68	100	100	100	100	100	100	100	98	100	100

Table 6.1: Query solving process: reliability with SCCA

Results with PHCA

If we test the same minimum speed threshold values with the variant PHCA of the mobile agent, we obtain another set of results, that are shown in Figure 6.26. This version of the agent never uses spatial crowdsourcing when it is inside the target area, nor when the agent leaves it accidentally and must return to resume the monitoring task. The mobile agent only hops from car to car by using the MAP hop strategy to estimate which car in range will reach the target area earlier.

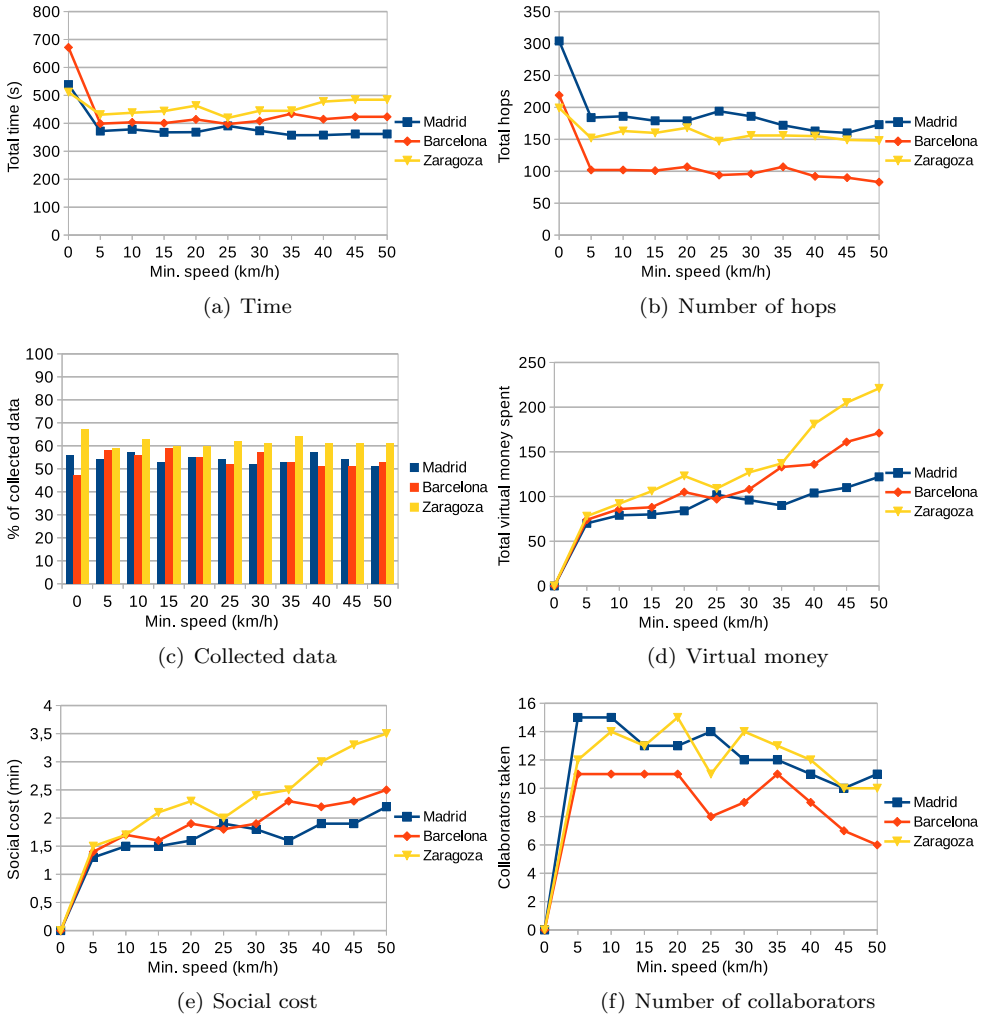


Figure 6.26: Query solving process: results with PHCA

In Figure 6.26(a), the total time is shown and the best results are, in general, for

minimum speed threshold values around 5 km/h, whereas for higher values the total time increases very slightly.

The total number of hops is shown in Figure 6.26(b), which exhibits a behavior similar to SCCA. For the extreme value of 0 km/h, the agent must only hop among the vehicles without using spatial crowdsourcing, and therefore the number of hops is much higher. For higher values of the minimum speed threshold, the number of hops decreases very slightly due to the use of spatial crowdsourcing to reach the target area and return to the origin.

The results of the amount of collected data, shown in Figure 6.26(c), are noteworthy: all the results are only around 60%, which is much lower than with SCCA. The reason is that, once the agent reaches the target area, it must visit the cells through only car-to-car hops, without using spatial crowdsourcing, which is harder to accomplish, as the number of vehicles inside the area is low and their trajectories unknown. Due to this, the agent needs more time to perform the task and in many cases it reaches the data collection timeout of 3 minutes even before it is completed.

In Figure 6.26(d), the virtual money spent by the agent is shown. For the minimum speed threshold value of 0 km/h, the amount of money spent is 0, since the agent never uses spatial crowdsourcing inside the target area, and neither to reach or leave it. For higher values, the best results are obtained for 5 km/h and the worst results for values higher than 30 km/h.

Regarding the social cost shown in Figure 6.26(e), we can observe the same behavior as with the SCCA version, and thus the collaborators are compensated according to their actual cost in quite a fair way.

Finally, in Figure 6.26(f), the number of collaborators taken by the agent is shown. This amount is exactly 0 for the minimum speed threshold value of 0 km/h, since spatial crowdsourcing is never used for traveling to/from the target area, nor for collecting data inside it when it is reached. For minimum speed threshold values of 5 km/h and above, the collaborators are used in a similar way to the SCCA variant of the agent: as the minimum speed threshold grows, the number of collaborators decreases. Since the total time remains more or less constant, this means that for low values of the minimum speed threshold, a collaborator is usually taken for a smaller time than for higher threshold values.

Regarding the reliability of the whole process (shown in Table 6.2) the best results are for minimum speed threshold values between 10 km/h and 45 km/h, with reliabilities of 100%. On the other hand, the worst results are for the value of 0 km/h, when no spatial crowdsourcing is used, but there are also some cases when the reliability does not reach 100% and stays in 98%.

minspeed (km/h)	0	5	10	15	20	25	30	35	40	45	50
Madrid (% of ended simulations)	68	98	100	100	100	100	100	100	100	100	100
Barcelona (% of ended simulations)	82	100	100	100	100	100	100	100	100	100	98
Zaragoza (% of ended simulations)	80	98	100	98	100	100	100	100	98	100	100

Table 6.2: Query solving process: reliability with PHCA

Comparison between SCCA and PHCA

Comparing the results of these experiments (regarding the determination of the minimum approaching speed threshold), we can conclude that:

- The variant SCCA of the agent takes less time than PHCA to complete the query solving process, due to the advantage of using spatial crowdsourcing inside the target area (on average, around 300 s in the first case and slightly more than 400 s in the second).
- Regarding the reliability, both approaches yield similar results, being not smaller than 98% for minimum approaching speeds of at least 5 km/h.
- Considering the total number of hops, SCCA is better (less than 100 hops, versus a value between 100 and 200 hops with PHCA) since the use of spatial crowdsourcing reduces the necessity to keep looking for potentially better vehicles to hop to.
- Regarding the data collection rate, again SCCA outperforms PHCA, being the first one around 100% and the second one around 60%.
- Concerning the virtual money cost, however, PHCA is better than the SCCA (with an average cost of 93 versus 126 units of virtual money), due to the fact that the former never uses spatial crowdsourcing once the mobile agent reaches the target area, which saves money.
- For both approaches, the use of spatial crowdsourcing with a minimum approaching speed threshold is always better than not using it, in terms of time, reliability, bandwidth usage and amount of collected data, as the results for the minimum speed threshold value of 0 km/h show.
- For both variants, a minimum approaching speed threshold around 5 km/h is appropriate, if we take into account all the results from these experiments.

Therefore, we can establish that the best algorithm in all terms (except in the virtual money cost) is SCCA, and we choose for the minimum approaching speed threshold 5 km/h, which is quite a low value. This value, as well as the SCCA strategy, are used by default in the experiments described in the following sections.

6.3.2 Influence of the Percentage of Potential Collaborating Vehicles

Figure 6.27(a) shows how the total time needed to complete the whole query solving process varies with the ratio of vehicles willing to act as collaborators in spatial crowdsourcing tasks. The required time is around 500 to 700 seconds for 0% of potential collaborators, that is, when no spatial crowdsourcing can be used and the mobile agent can only rely on hopping from one vehicle to another to reach the destination

and perform the query solving task inside the target area. The time decreases to around 400 seconds when the ratio of potential collaborators increases to 10%, and then decreases slightly until it reaches around between 300 and 350 seconds for 100% collaborators (i.e., all the vehicles in the scenario can act as collaborators). The reason is that once the mobile agent finds a suitable collaborator it is not necessary to keep looking for another one, and thus having more potential collaborators to choose from is useless.

Figures 6.27(b), 6.27(c) and 6.27(e) show the number of hops performed by the mobile agent, the amount of data collected, and the social cost, respectively. In all these figures, we can see the same pattern: when spatial crowdsourcing is not used (because the ratio of potential collaborators is 0%), then the corresponding metric measured is significantly worse than when it is used. The results improve dramatically with only about 10% collaborators. However, when the ratio of potential collaborators is higher than this value, the measured parameters do not show further improvement (the mobile agent does not significantly benefit from a higher number of potential collaborators). The exception to this pattern is the virtual money spent, shown in Figure 6.27(d), which is obviously 0 when no collaborators are used for performing spatial crowdsourcing.

Regarding the number of collaborators used by the agent, shown in Figure 6.27(f), it is obviously 0 when the ratio of potential collaborators is 0%. For higher ratios, the number of effective collaborators used by the agent varies more or less randomly, but for all the cases its number is within the range of 10 to 21 collaborators. When the mobile agent commutes from spatial crowdsourcing to hopping to other vehicles and back again to spatial crowdsourcing, it seems to find a new collaborator easily, as the total time for the monitoring (shown in Figure 6.27(a)) does not increase significantly.

In Table 6.3, we can see how the *reliability* is between 68% and 80% (depending on the city map) when the ratio of potential collaborators is 0%. For 10% of potential collaborators, it increases to between 96% and 100%, and for higher values the reliability is always 100%, with the exception of a few cases where it is 98%. Therefore, not being able to use spatial crowdsourcing (i.e., 0% of potential collaborators) makes the process more unreliable than using it.

% of collaborators	0	10	20	30	40	50	60	70	80	90	100
Madrid (% of ended simulations)	68	100	100	100	98	100	100	98	100	98	100
Barcelona (% of ended simulations)	82	100	100	100	100	100	100	100	100	100	100
Zaragoza (% of ended simulations)	80	96	100	98	100	100	100	98	100	100	100

Table 6.3: Query solving process: reliability varying the percentage of collaborating vehicles

As a conclusion for this experiment, we can say that the benefits of using spatial crowdsourcing are remarkable even with a ratio of potential collaborators as low as 10%. Therefore, we do not need a high number of drivers willing to modify their trajectories to benefit from the spatial crowdsourcing approach and increase the performance of the process.

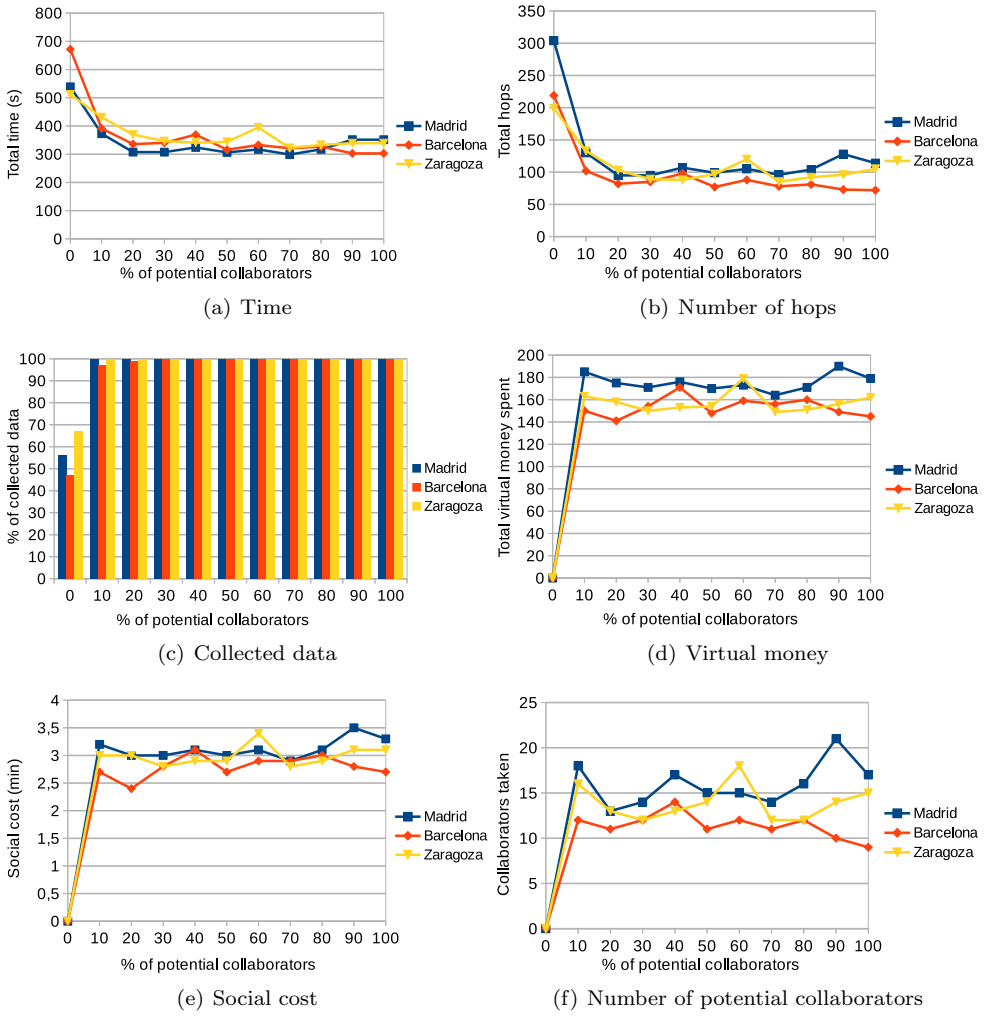
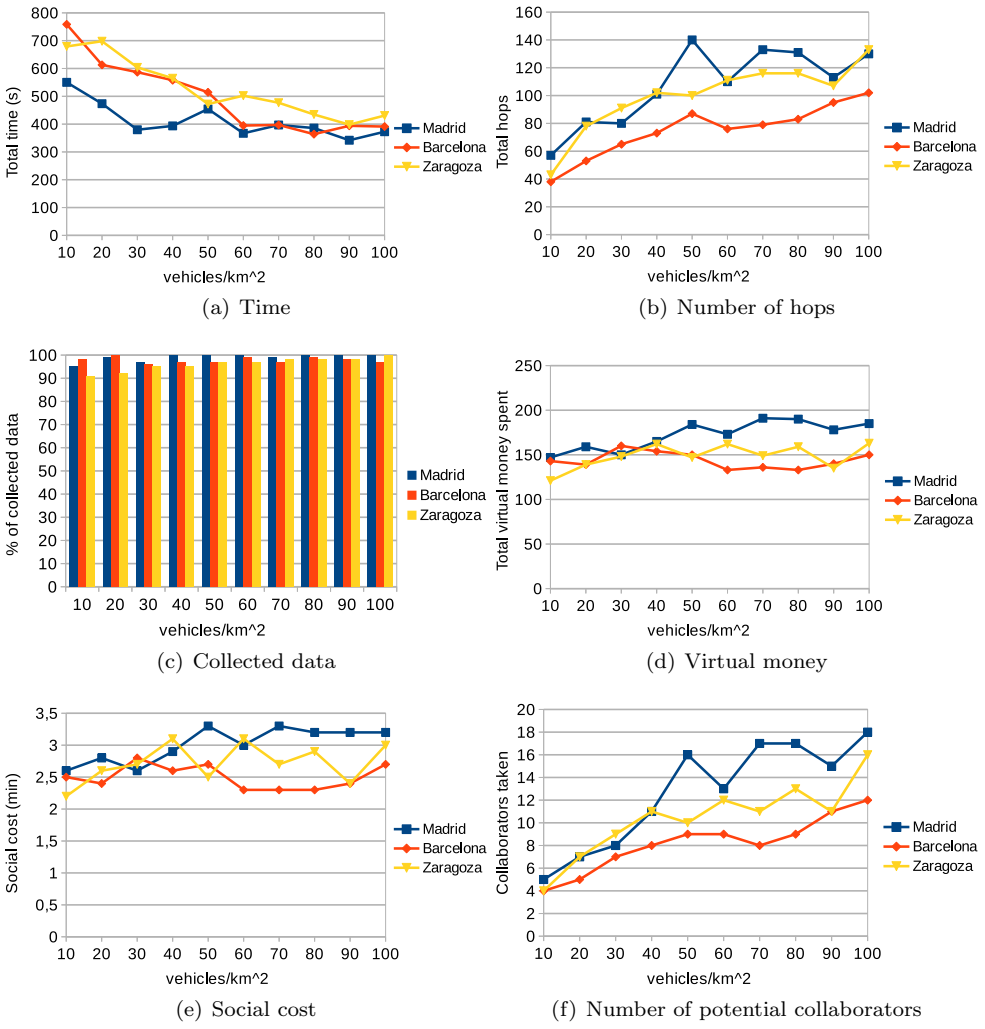


Figure 6.27: Query solving process: results varying the percentage of collaborating vehicles

6.3.3 Influence of the Density of Vehicles

In this experiment, we test the influence of the density of vehicles on the performance of the query solving process. In the previous experiments, where the density was set to 100 vehicles/km^2 (a medium value), the results showed that even with a small number of potential collaborators the performance improves dramatically. However, we also want to test it in a more challenging scenario with low and very low vehicle density, to find out the limits of spatial crowdsourcing. The chosen density values and their denomination (low density, medium density, etc.) are inspired by those used in

proposals such as [MFT+13, BGF+13], and for this experiment we vary them from 10 vehicles/km^2 (very low density) to 100 vehicles/km^2 (medium density). Based on the results of the experiment presented in Section 6.3.2, in this experiment the percentage of potential collaborating vehicles is set to 10%.



(a) Time (b) Number of hops

(c) Collected data (d) Virtual money

(e) Social cost (f) Number of potential collaborators

Figure 6.28: Query solving process: results varying the density of vehicles

In Figure 6.28(a), the total time to complete the query solving process is shown. As the number of vehicles increases, the time needed decreases, with little variance when the density reaches around between 70 and 80 vehicles/km^2. The reason is that, with a high number of vehicles, the mobile agent has less difficulties to find a car to

be carried nearer the target area, whether it uses spatial crowdsourcing or just hops from one vehicle to another.

Regarding the number of hops performed by the mobile agent, shown in Figure 6.28(b), its number increases with the vehicle density, since the agent is constantly looking for a better option to reach its destination sooner: if there exist more vehicles, then the number of opportunities is also higher and the agent will likely hop to another car more frequently.

The amount of data collected within the timeout of 3 minutes, shown in Figure 6.28(c), increases with the vehicle density, although there is little variation and all the values are higher than 90%.

Figures 6.28(d) and 6.28(e) show the virtual money paid by the mobile agent to the collaborators and the social cost, respectively. As in the previous experiments, these figures are closely related, and both values increase slightly with the density of vehicles. With a higher number of vehicles there is also a higher number of potential collaborators, and the agent can obtain help more easily, and therefore it could end up spending more virtual money. On the other hand, with less vehicles there is a smaller number of potential collaborators, so it is more likely that the agent cannot find one of them when needed and, consequently, it might spend less virtual money.

Finally, the number of collaborators taken by the mobile agent, shown in Figure 6.28(f), increases significantly with the density of vehicles, for the same reasons: since the number of potential collaborators is higher, the agent has more opportunities to take advantage of them and they are used more frequently.

Table 6.4 shows the reliability of the query solving process for the different traffic density values. As the vehicle density increases, so it does the reliability, and starting with 50 vehicles/km^2 there is little variance. This result is related to the total time required for the query solving, as the sooner the mobile agent completes all the phases of the process, the less likely the time limit will be reached.

Density (vehicles/km^2)	10	20	30	40	50	60	70	80	90	100
Madrid (% of ended simulations)	86	98	98	100	98	100	98	100	98	100
Barcelona (% of ended simulations)	74	86	100	92	96	98	100	100	100	100
Zaragoza (% of ended simulations)	80	82	96	88	98	100	100	100	100	96

Table 6.4: Query solving process: reliability varying the density of vehicles

As a conclusion for this experiment, we can say that, regarding the influence of the vehicle density on spatial crowdsourcing, the best results in terms of time, reliability and amount of collected data are obtained for values around 60 vehicles/km^2 (a *low* density) and higher. For lower densities, the results are worse, but not so bad for the spatial crowdsourcing to be considered useless under those circumstances: for example, for the extremely low density value of 10 vehicles/km^2 the amount of collected data is higher than 90% and the reliability is near 75% in the worst case, which may be enough for query solving tasks with soft requirements.

6.3.4 Influence of the Minimum Stay Time in Collaborators

In this experiment, we evaluate the impact of the minimum stay time in the collaborating vehicles. Specifically, we vary this value from 0 to 50 seconds in increments of 5 seconds, to evaluate its impact. Besides, based on the results of the experiment presented in Section 6.3.2, in this experiment the percentage of potential collaborating vehicles is set to 10%. In Figure 6.29(a), we can see how the total time to complete the process decreases slightly as the minimum stay time increases. The reason is that the likelihood of wasting time decreases when the agent stays in a collaborator longer, as the collaborator commits itself to bring the agent to its destination. Besides, the number of hops also decreases, as shown in Figure 6.29(b).

Regarding the amount of collected data, shown in Figure 6.29(c), it remains more or less constant as the minimum stay time increases, and it is always above 96%. The mobile agent takes into account the minimum stay time only in the phases of the process where it approaches the target area and returns to its origin, but not in the data gathering phase, so it has no influence during the data collection step.

The invested virtual money and the social cost, that can be seen in Figures 6.29(d) and 6.29(e), respectively, exhibit a similar behavior, both growing with the minimum stay time. The difference between the values observed for the lowest minimum stay value (0 seconds) and the highest minimum stay value (50 seconds) is about 25%, and the reason is that the mobile agent wastes less time in hops to other vehicles and stays longer in the collaborating vehicles, so these collaborators receive more compensations in the form of virtual money. This fact is confirmed by the number of collaborating vehicles taken by the agent, shown in Figure 6.29(f). As the minimum stay time grows, the number of collaborators decreases strongly. Along with the low variability of the total time, this means that the mobile agent takes a smaller number of collaborators but stays longer in them.

As shown in Table 6.5, the reliability parameter is 100% for almost all the minimum stay times (with only one exception, with 96%, for the city of Zaragoza and a minimum stay time of 0 seconds). The reliability is related to the total time of the process (and whether it takes longer than the timeout or not), and given the small variance of the total time, the minimum stay time has no appreciable effect on the reliability.

Minimum stay time (s)	0	5	10	15	20	25	30	35	40	45	50
Madrid (% of ended simulations)	100	100	100	100	100	100	100	100	100	100	100
Barcelona (% of ended simulations)	100	100	100	100	100	100	100	100	100	100	100
Zaragoza (% of ended simulations)	96	100	100	100	100	100	100	100	100	100	100

Table 6.5: Query solving process: Reliability varying the minimum stay time of the mobile agent in collaborators

As a conclusion, the use of a minimum stay time in a collaborator has a certain influence on the number of hops that the agent needs to take, decreasing them as the minimum stay time grows. However, this reduction has no influence on the reliability or the amount of collected data, and increases the social cost of the collaborators and the amount of virtual money spent by the agent.

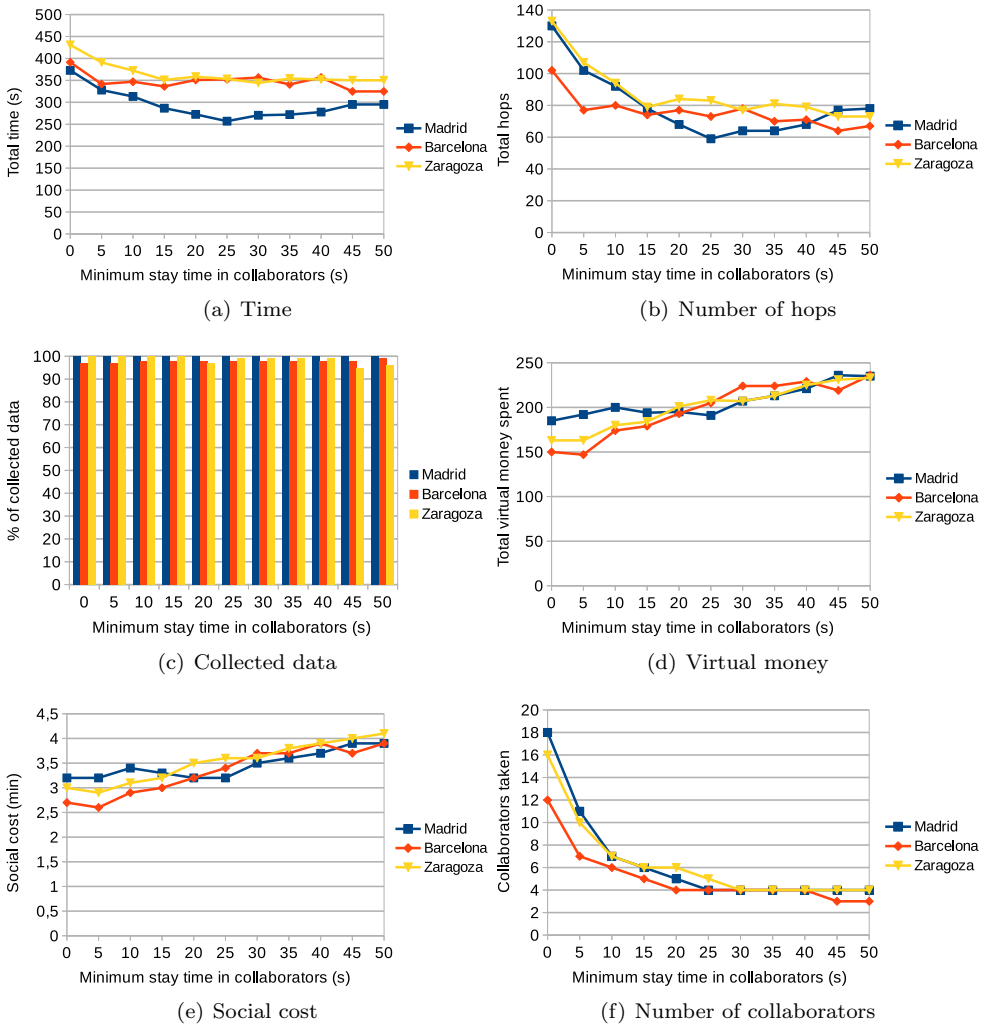


Figure 6.29: Query solving process: results varying the minimum stay time of the mobile agent in collaborators

6.4 Summary of the Chapter

In this chapter, we evaluated experimentally the travel strategies and the whole query processing approach. In both cases, we considered them using the *basic* approach and the version enhanced with spatial crowdsourcing.

To evaluate the basic approach, we tested a number of factors that might have an influence on the performance of the query processing, such as the uncertainty in

the position of the query originator, the relevance of the information stored in the vehicles, the vehicle density, and others.

We also explored the potential benefits and disadvantages of cloning a mobile agent (i.e., the deployment of multiple copies of the agent) and we tested and compared two variations of this strategy. In the first variant, the clones were only used for collecting data once a single copy of the mobile agent reaches the target area. In the other variant, the clones were used during the whole process, and the agent starts to replicate itself a few seconds after departing from its originator.

We also evaluated the approach based on spatial crowdsourcing, where the mobile agent asks for help to potential collaborators when its approaching speed to the target area is too slow. The first experiment focused on determining a suitable speed threshold to switch to spatial crowdsourcing. After that, we tested two variations of a spatial crowdsourcing algorithm: SCCA and PHCA. In SCCA, the mobile agent only asks for the help of collaborators in the data collecting step of the process. In PHCA, the agent only uses spatial crowdsourcing to reach the target area but, once there, it only hops among the vehicles to collect the required data.

Finally, we chose the best of the variants (SCCA) and performed another set of experiments to evaluate the influence of parameters such as the percentage of potential collaborators, the density of vehicles, or the minimum time that the mobile agent must stay in a vehicle.

The experiments presented in this chapter show the feasibility of the proposed approach and the additional benefits of spatial crowdsourcing even if only a small amount of drivers are willing to cooperate.

Chapter 7

Use Case Example: Parking Spaces in a Mixed Urban/Interurban Scenario

Looking for available parking spaces is a task that consumes both time and money from drivers, especially in big cities. According to a recent study by INRIX Research [Coo17], "searching for parking imposes a significant economic burden with drivers in the U.S., U.K. and Germany wasting 17, 44 and 41 hours a year respectively at an estimated cost of $72.7 billion, £23.3 billion and €40.4 billion a year in these countries". In another article published in the Washington Post [Hal10], professor D. Shoup "studied a 15-block business district in Los Angeles and determined that cruising about 2.5 times around the block for the average of 3.3 min required to find a space added up to 950,000 excess miles traveled, 47,000 gallons of gas wasted and 730 tons of carbon dioxide produced in the course of a year".

Therefore, there exists a lot of interest in the research community in finding methods, related to communications and information technologies, that help drivers to find available parking spaces more easily, so that they do not need to spend so much time, fuel, and money in such an ungrateful and inefficient task.

In this chapter, we present a use case scenario that illustrates the techniques related to data management in VANETs using mobile agents described in this thesis. It tackles the problem of searching for a number of available parking places in an area towards where the user is driving, by retrieving data from vehicles located in its vicinity. It has the particularity of taking place in a scenario where there exist both urban and interurban roads, which is not usual in the literature. In Section 7.1, we describe in detail all the features of the scenario and the setup of the experimental evaluation. In Section 7.2, we perform an experiment to test the influence of both the urban and interurban vehicle density on the data retrieval task. In Section 7.3, we test the influence of the initial density of occupied parking places, and in Section 7.4

the influence of the number of requested places.

7.1 Scenario Evaluated

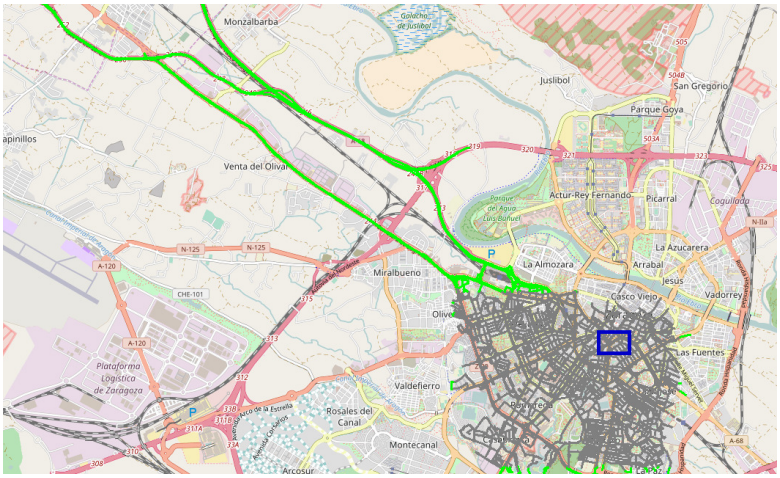
In the use case considered, a person traveling along a highway approaches a city to attend a meeting [UI17a]. When he/she is 15 km from the city, he/she wants to obtain information about available parking spaces near the meeting point (located in the city center). With that goal, a mobile agent is launched, that travels to the destination area (the city center) with the purpose of collecting information about available parking spaces by querying the vehicles present in that area; we assume that the driver will be satisfied when he/she retrieves information about a specific number of parking spaces, that will provide him/her with an overview of the parking availability in the area.

As shown in Figure 7.1, the map scenario chosen is a portion of the city of Zaragoza (Spain) and some fragments of interurban roads where the vehicle with the driver asking for parking spaces is located when the experiment starts. We consider for this scenario two areas. On the one hand, we can observe the *urban area*, which includes the city streets, where the maximum speed of vehicles is around 50 km/h and the presence of buildings block the propagation of wireless signals. On the other hand, the *interurban area* includes roads and highways that extend northwest, where vehicles travel at speeds of up to 120 km/h, and where there are no buildings blocking radio signals (and therefore the vehicles can transmit data to distances close to the maximum theoretical communication range). The rectangle shown in Figure 7.1 shows the destination area in the city (i.e., the city area near the meeting place), whose size is 0.25 km^2. Some parameters of the simulations considered in the experiments carried out are shown in Table 7.1, while the rest of parameters (such as the hop strategy used by the mobile agent, the wireless communication range, the mobility model, the mobile agent size, or its hop delay) have the same values defined in Table 5.1. A few parameters deserve further explanations:

- The *percentage of vehicles with relevant data* is set to 50% by default, which means that only half of the vehicles within the area are really relevant to collect data for the query processing. This percentage allows to simulate cases where, for example, some vehicles do not participate in the data sharing and therefore store no information about available parking spaces.

- The *percentage of available parking spaces per vehicle* is the percentage of available parking places within the area that are stored in a vehicle within that area. With a perfect push-based data sharing approach, the value of this parameter could reach 100%. However, due to a number of reasons it is not reasonable to assume that all the vehicles will have all the information about the available parking spaces in the vicinity. For that reason, we have decided to choose by default a value of 10% for this parameter, which is quite pessimistic. This means that, for example, if there are 50 available parking spaces, each vehicle with data

(a) Roadways selected for the experiments



(b) Whole area as seen in OpenStreetMap

Figure 7.1: Scenario map used in the use case simulation

would have information about 5 of those available parking places. The smaller this value, the higher the number of vehicles that the mobile agent will need to visit in order to collect information about a specific number of parking places required. Independently of the specific value of this parameter, as the value of the parameter *percentage of vehicles with relevant data* is smaller than 100%, some vehicles will store no data about parking spaces.

- The *data processing delay* represents the amount of time needed by the agent

Parameter	Default value
Map dimensions	4 $km\times$ 4 km (urban area) 20 km (interurban area)
Map scenario	Zaragoza (Spain) and its surroundings
Size of the destination area	0.25 km^2
Distance to the destination area	15000 m
Density of vehicles	Medium density: 50 vehicles / km^2 (urban area) 10 vehicles / km (interurban area)
Speed of the vehicles	50 $km/h \pm 10\%$ (urban area) 120 $km/h \pm 15\%$ (interurban area)
Percentage of vehicles with relevant data	50%
Total number of parking spaces in the destination area	220 *parking spaces*
Percentage of available parking spaces per vehicle	10%
Data processing delay	5 s
Number of available parking spaces to retrieve	10 *available parking spaces*
Parking occupancy	90%
Data collection timeout	4 *minutes*
Warning timeout	60% of the data collection timeout

Table 7.1: Use case simulation parameters

to extract the relevant data from the vehicle that it is visiting.

- The *warning timeout* represents an amount of time (over the total data collection time available, represented by the *data collecting timeout* parameter) that, if exceeded, will lead the agent to enlarge the search area (100 meters in each direction) to try to find available parking spaces more quickly. This represents a classical parking search behavior: the driver starts looking for a parking space as close to the destination area as possible, but he/she will search further also if there are difficulties to find parking there.

With these settings, we perform different experiments to test the influence that a number of parameters have on the retrieval of information about available parking spaces. As a result, we obtain some metrics that help to evaluate the performance of the process, such as the total query processing time, the number of hops performed by the mobile agent, the amount of available parking places found, or the effective speed of the agent.

7.2 Influence of the Interurban Vehicle Density

In this experiment, we evaluate how the retrieval of information about available parking spaces is influenced by the density of vehicles in the interurban area. Interurban areas are characterized by long roads (lengths of tens of kilometers), a constant number of lanes, a small number of intersections with other roads, and the lack of buildings that block wireless communication signals. Due to the long length of these roads, the

most suitable traffic density measurement unit is the *number of vehicles per linear kilometer*, or simply the *vehicles per kilometer*. In this experiment, we vary this value from 2 $vehicles/km$ (very low density) up to 20 $vehicles/km$ (high density).

On the one hand, Figure 7.2 shows how the time required to obtain the desired information about available parking spaces decreases as the density of vehicles in the interurban area increases. Besides, the figure shows that the delays are quite reasonable for this type of query. On the other hand, Figure 7.3 shows the total number of hops performed by the agent. This value is the number of times that the mobile agent moves successfully from one vehicle to another, and it can be used as a measure of the bandwidth required by the query processing, since the completion of such hops requires that the code and data of the agent must be transferred using the wireless connection. When the density of vehicles increases, the number of hops also increases, due to the fact that the mobile agent is constantly looking for more promising vehicles to try to reach sooner the destination area, and the agent will move immediately to any of them as soon as it enters within its communication range; moreover, when the agent transports itself through the wireless medium (transportation via communications) it can travel faster than if it just lets itself be transported by a car (transportation via locomotion). The growth in the number of hops, however, is not very steep and it stabilizes for about 8 to 10 vehicles per kilometer.

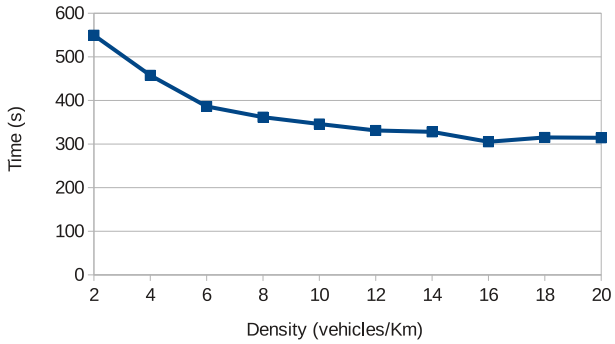


Figure 7.2: Searching for parking spaces: time to solve the query according to the density of vehicles

In Figure 7.4, we show the effective speed of the mobile agent when it solves the query about available parking places. The speed grows with the density of vehicles in the interurban area from around 100 km/h to slightly more than 100 km/h when the density is around 14 $vehicles/km$, remaining more or less constant from that value. Note that the vehicles travel physically at no more than 130 km/h, and therefore for densities greater than 4 $vehicles/km$ the mobile agent travels faster than the vehicles it uses for moving, thanks to the use of wireless ad hoc connections.

Figure 7.5 shows the number of vehicles whose data have been processed by the mobile agent looking for information about available parking spaces. Note that this number refers to the number of vehicles that have data about parking places, since

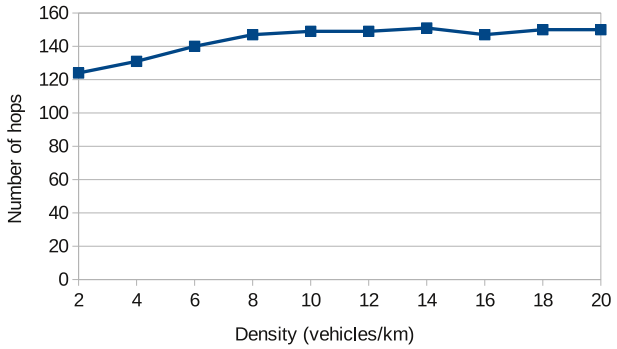


Figure 7.3: Searching for parking spaces: number of hops performed by the agent according to the density of vehicles

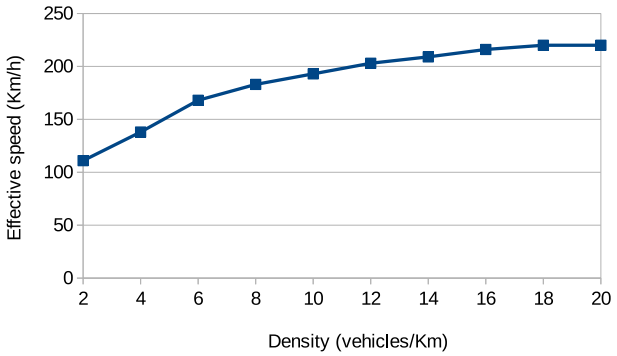


Figure 7.4: Searching for parking spaces: effective speed of the agent according to the density of vehicles

there also exist other irrelevant vehicles where the agent can move but that do not contain data about parking spots (the amount of relevant vehicles set for these experiments is 50%, as shown in Table 7.1). Since, in this experiment, we vary the *interurban* density of vehicles and the query is solved in the phase of data collection, that takes place in the urban area, the interurban density has no influence on the number of vehicles processed by the mobile agent, and therefore it remains approximately constant in the experiments.

7.3 Influence of the Occupancy of Parking Spaces

In this experiment, we vary the occupancy rate of the parking spaces (i.e., the ratio of available parking places in the scenario), to evaluate its impact on the performance of our proposal. For this, we vary the parking occupancy from 0% (all the parking places are available) to 100% (there is no available parking space). Note that in the

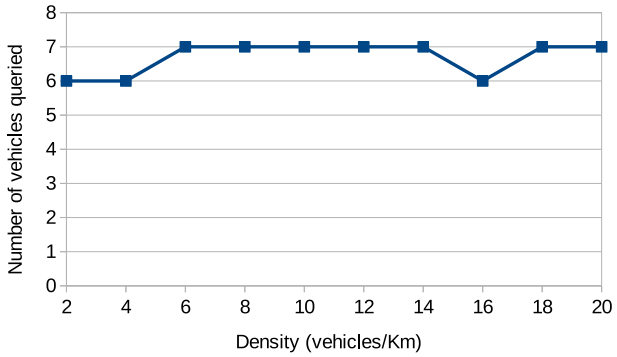


Figure 7.5: Searching for parking spaces: number of vehicles with data queried by the agent according to the density of vehicles

latter case the mobile agent will always fail in accomplishing its task.

Figure 7.6 shows the total time needed to solve the query searching for parking spaces. We can see that the rate of parking occupancy has little impact on the total time. The reason is that the mobile agent spends most of the time traveling to the destination area and returning to the originator vehicle, whereas the actual data collection process is performed quite fast. The average time spent in the different phases of the query processing is shown in Table 7.2. The first phase (mobile agent traveling to the destination area) takes always the same time independently of the occupancy of the parking spaces, since the ratio of available parking spaces does not have any influence on it. In the second phase (mobile agent collecting data), the time invested increases with the occupancy rate, since the agent will need to visit more vehicles to find information about the required number of available parking spaces. Finally, the time invested in the fourth phase (mobile agent traveling back to the origin vehicle that submitted the query) decreases as the occupancy of parking spaces increases; this may seem surprising, but there is an explanation: whereas the mobile agent is collecting data in the urban area, the origin vehicle is also traveling towards that area, and therefore the distance that the agent will need to traverse to reach the vehicle decreases along time.

Parking occupancy	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Phase 1 time (s)	160	160	160	160	160	160	160	160	160	160	160
Phase 2 time (s)	11	11	11	11	11	11	22	22	31	77	77
Phase 3 time (s)	0	0	0	0	0	0	2	2	2	2	2
Phase 4 time (s)	123	123	121	121	120	122	120	122	110	108	108
Total time (s)	294	294	292	292	291	293	304	306	303	347	347

Table 7.2: Searching for parking spaces: average time per phase in the query processing for the use case

Figure 7.7 shows the percentage of the number of parking spaces requested by the user that are effectively collected and returned to the driver by the mobile agent.

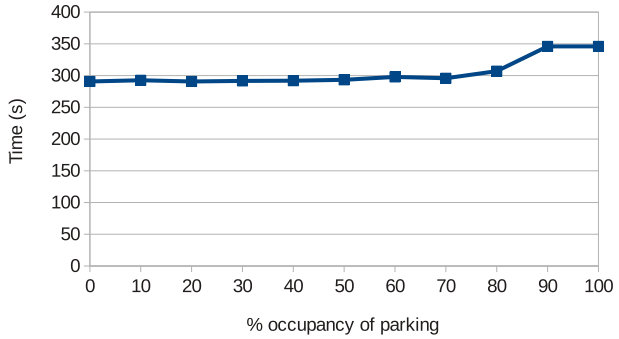


Figure 7.6: Searching for parking spaces: time to solve the query according to the parking occupancy

For all the parking occupancy rates, the result is 100% (i.e., the information about all the spaces requested was found by the agent), with the exception of the case where there are no available parking spaces. In that case, the result is 0%, since it is impossible for the agent to find any parking spot available. As in the experiment we increase the occupancy rate by 10% increments, the second worse case scenario evaluated corresponds to a situation where 90% of the parking spaces are occupied, but with this occupancy rate there are still enough available parking spaces within the destination area (specifically, 22 available parking spaces) to satisfy the query.

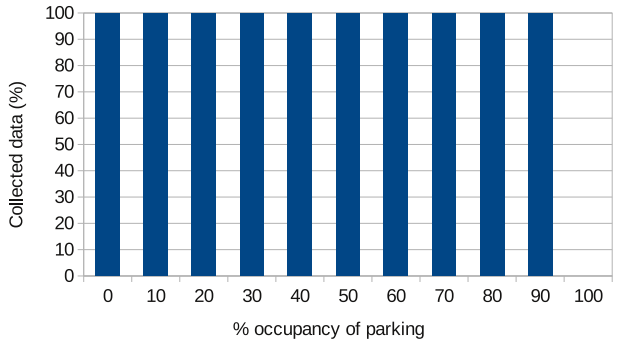


Figure 7.7: Searching for parking spaces: percentage of collected data according to the parking occupancy

Figure 7.8 shows the number of vehicles whose data have been processed by the mobile agent. This number remains quite small until the parking occupancy increases considerably. Besides, when there is no available parking space (occupancy rate of 100%), the number of processed vehicles rises abruptly: this is due to the futile efforts performed by the mobile agent to try to solve the query, looking for non-existing data. In such a case, the timeout established for data collection is reached and the agent

returns to its query originator vehicle with an empty response, which on the other hand might be a useful answer for the user (i.e., the area is packed, so it will be very difficult to park there!).

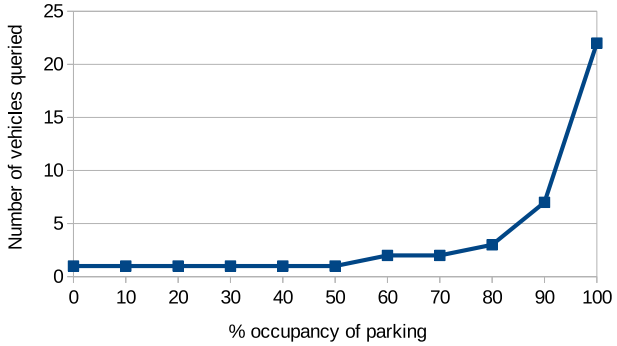


Figure 7.8: Searching for parking spaces: number of vehicles with data queried by the agent according to the parking occupancy

7.4 Influence of the Number of Requested Available Parking Spaces

In this experiment, we vary the number of available parking spaces that the driver wants to retrieve to have a good overview of the availability of parking spaces. The parking occupancy ratio is set to 90%, as this is a quite challenging scenario. Therefore, the total number of available parking spots in the destination area is initially 22, and thus we vary the requested number of parking spaces to retrieve from 2 to 22; it should be noted that if the agent cannot find enough available parking spaces within the destination area (e.g., because the requested number of parking spaces is higher than 22), then it would enlarge its searching area, according to the parameter *warning timeout* (explained in Section 7.1). Figure 7.9 shows the total time needed to solve the query searching for parking spaces. As expected, the higher the number of parking spots to collect, the higher the time needed by the mobile agent to complete the process. However, the query processing times are not excessive in any case.

Figure 7.10 shows the total number of hops performed by the agent. When the number of requested available parking spaces increases, the number of hops also increases, since the mobile agent needs to visit more vehicles (as each vehicle has information about a limited number of places). Consistently with this result, Figure 7.11 shows the number of relevant vehicles that the mobile agent needs to visit to get the requested data. Unlike in previous experiments, in this case the number of processed vehicles grows, since the mobile agent needs to visit an increasing number of them to find the information they have about the existing available parking spots.

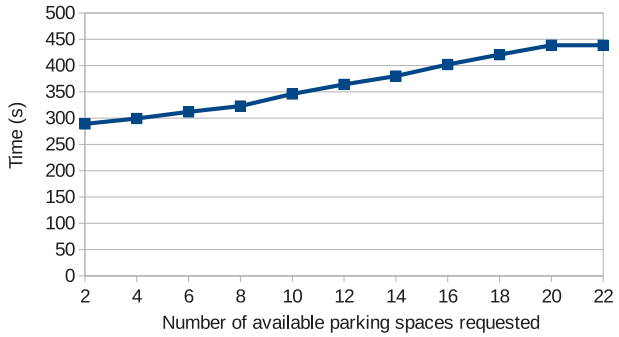


Figure 7.9: Searching for parking spaces: time to solve the query according to the number of requested parking spaces

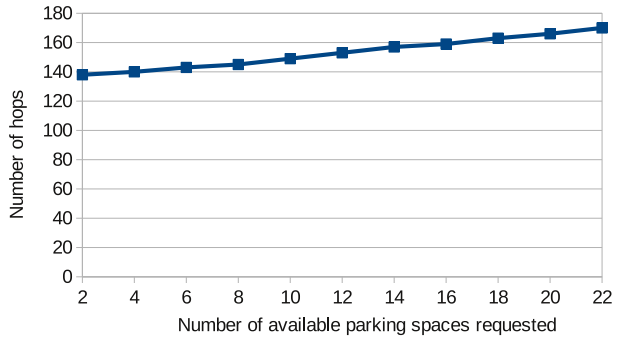


Figure 7.10: Searching for parking spaces: number of hops performed by the agent according to the number of requested parking spaces

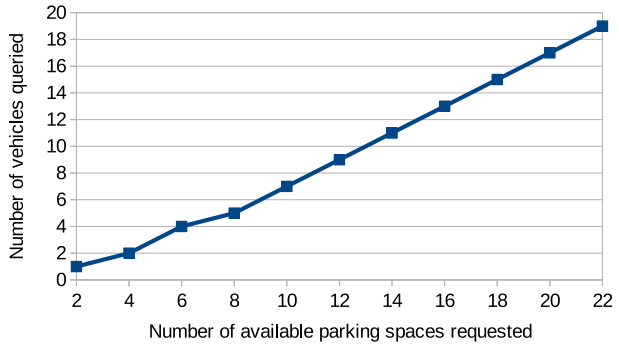


Figure 7.11: Searching for parking spaces: number of relevant vehicles visited by the agent according to the number of requested parking spaces

Finally, the percentage of collected data was 100% in all the cases, with the exception of the case of 22 parking spaces, where a 98% of the requested parking spaces (i.e., around 21.5 places, on average) were found. Despite the efforts performed by the mobile agent, that enlarges the searching area to find parking places that are located further when the warning timeout is reached, in some cases it is not possible to find the requested number of available parking spaces within the data collection timeout established, due to the randomness of the information available in the vehicles as well as the high parking occupancy.

7.5 Conclusions of the Evaluation for the Use Case

The experimental results presented show the performance of the proposal in different conditions and its feasibility:

- The mobile agent behaves well in both types of roads and is able to travel long distances in a relatively short time.
- The agent spends most of the time in traveling from the origin vehicle to the destination area and in returning to the origin vehicle once the required data have been collected, whereas the process of searching data inside the interest area takes comparatively less time.
- The whole process is completed with enough time for the mobile agent to return to the origin vehicle with the result before the vehicle reaches the destination area. Thus, the driver can retrieve the desired information with enough time in advance.
- A higher density of vehicles leads to an improvement of the performance of the query processing. Similarly, the query processing is favored when there is a higher number of available parking spaces stored by the vehicles, a lower parking occupancy rate, or a lower number of places requested by the driver.

Up to the authors' knowledge, existing proposals to provide information about available parking spaces are push-based, and therefore they cannot be used in scenarios like the one studied in this use case, where the driver needs to retrieve information about parking spaces in areas that are not nearby. Moreover, the use of mobile agents to search available parking spaces is also new. Finally, the experimental evaluation of approaches for VANETs usually focus either on urban scenarios or highways, whereas we have considered a mixed scenario that includes both a urban area and an inter-urban region.

7.6 Summary of the Chapter

In this chapter, we presented a use case that uses the technology of mobile agents to find available parking places in a city area by looking *in situ* for that information

among the data collected individually by the vehicles that circulate in that area. The novelty of this study resides in the use of a pull-based approach to query data about parking spaces in areas that are not located near the driver who submits the query (thus allowing new interesting use cases), the use of mobile agents in an on-street parking space searching scenario, and the experimental evaluation with real maps combining both city areas and interurban areas.

Upon this scenario, a number of experiments were performed to test the solution and analyze the influence of parameters such as the vehicle density, the occupancy of parking places, and the number of places requested by the user. As a result, we obtained some metrics about the invested time, the number of parking spaces found, or the bandwidth usage needed to complete the query searching for the requested data. With all these results, we concluded that the use of mobile agents for retrieving data about available parking spaces is an adequate choice that performs well in a mixed urban/inter-urban scenario.

Chapter 8

Related Work

In the previous chapters, we described the main features of our approach for data management in vehicular networks using mobile agents. In this chapter, we present some works about topics related to our proposal. In Section 8.1, we first review works about the usage of mobile agent technology in vehicular networks. In Section 8.2, we present works about processing queries in VANETs. In Section 8.3, we focus on spatial crowdsourcing techniques and crowdsensing. In Section 8.4, we briefly describe some works about monitoring and the use of vehicular networks as mobile networks of sensors. Finally, in Section 8.5, we review works related to searching available parking spaces, as this is the problem tackled in the use case scenario presented in Chapter 7.

8.1 Mobile Agents in VANETs

As described in Section 2.5, mobile agents can provide significant benefits in distributed and mobile environments, thanks to their ability to move from one execution environment (in a computer or device) to another in an autonomous way by means of wireless communications. Along with their flexibility, that allows them to be programmed to perform any task, this makes them an adequate option for the development of applications in an environment such as a VANET, which is a highly-dynamic mobile and distributed system. In the following, we present some works related to the use of mobile agents in the context of vehicular networks.

In [dFHC+11], mobile agents travel to areas where different environmental sensors are used to collect data using the vehicles in a taxi fleet. This is probably the most similar work to ours, concerning the use of mobile agents in VANETs, but it focuses on data from environmental sensors and does not cover query processing. Besides, the experimental evaluation is not as extensive and realistic as ours, since the authors do not use real city maps (instead, they use grid-like maps). That work is in the same spirit as a previous work that we had presented in [UIDM09]. We later presented also a methodological approach and outlined our research plans to extend that initial work to go beyond environmental monitoring and tackle the processing of queries in

vehicular networks [UI13], and in subsequent works we fully developed our proposal.

In [KD10], mobile agent technology is used in VANETs to manage the Quality of Service (QoS) in the network. Specifically, an architecture involving stationary and mobile agents, both in the moving vehicles and in the fixed roadside equipment, is proposed. The agents constantly search ad hoc paths to send messages, according to certain constraints in terms of bandwidth, latency, and other parameters. In this way, the messages can be delivered more efficiently to their destination (the drivers in the VANET), with the goal of improving the security and flow of the road traffic. In our proposal, mobile agents also take routing decisions to transport data and queries through the vehicular network, but they perform these tasks within the scope of a whole query processing framework. Besides, no experimental evaluation is presented in [KD10].

In [RNBI07], a system called TJam is used to predict traffic jams in certain areas of highways using short-range vehicle-to-vehicle communications. This system is presented as a proof-of-concept to test a framework built to provide migratory services in ad hoc networks. These *migratory services* can be seen as mobile agents (i.e., code that moves from one execution place to another to offer a service), and therefore that work shows how useful they can be in a vehicular network scenario.

In [VS11], the authors propose a system that uses both mobile and static agents to find parking spaces. The static agents are called URAs (Unique Routing Agents) and they are hosted in fixed network nodes near intersections and parking lots. The URAs periodically collect routing and service availability information and, when a vehicle looks for a parking space, it launches a mobile agent, called a Discovery Agent, to the nearest URA node. Then, the URA provides the Discovery Agent with information about nearby parking spaces, and it guides the driver towards them. This work focuses on the specific use case of searching parking spaces, rather than on distributed data management in vehicular networks to support different types of queries. Besides, as opposed to our proposal for the parking space case study described in Chapter 7, this approach relies on the availability of fixed nodes that index data about parking spaces. Finally, the URAs provide data about nearby parking spaces, and therefore this solution cannot be applied in the case of a driver approaching a city and who requires information about the availability of parking spaces in the city center well in advance.

Dynamic clustering is a technique to form groups of vehicles on the fly, which is a challenging problem due to the rapidly-changing network topology and frequent network disconnections of vehicles. In [KM12], the authors propose a multiagent-driven dynamic clustering scheme for VANETs, by considering the vehicle's speed, direction, connectivity degree to other vehicles, and mobility pattern. The scheme comprises heavyweight static agents and lightweight mobile agents. The authors evaluate the performance and effectiveness of the proposed scheme by comparing it with an existing clustering scheme. According to the experimental results presented, it performs better in terms of cluster formation time, cluster member selection time, cluster head selection time, and control overheads. Instead of query processing issues, this work tackles the problem of vehicle clustering, which can be useful, for example, as

part of a contention-based forwarding approach for data dissemination in a vehicular network. For example, in [VN15] a specific vehicle within the cluster (the *cluster head*) is in charge of forwarding messages.

In [HZS+15], S-Aframe is presented, which is a framework for the development of vehicular social networks which uses mobile agents as the underlying technology. The framework consists of different layers, and mobile agents are used to provide the needed flexibility to adapt the system to the changing conditions of a mobile environment, including network connectivity and the contexts of the users of the vehicular social network. Although this work also uses mobile agents in a vehicular network, it focuses on a problem that is out of the scope of this thesis.

Finally, in [CCP09], mobile agents are also used for a vehicle-related topic. However, that work is focused on urban traffic management using a fixed infrastructure, instead of considering the general case of mobile agents that hop among vehicles using ad hoc network connections.

Overall, as described in [IDTL15], the use of mobile agents in Intelligent Transportation Systems (ITS) and vehicular networks has been quite unexplored so far. This claim can be extended to standard agents in general, according to [DIdCRH14]. With this thesis, we have contributed to cover this research gap.

8.2 Query Processing in VANETs

As described in Section 2.6, query processing in vehicular networks involves the retrieval of data that meet the required conditions and that are distributively stored among the vehicles that travel through the roads of an area. The data can be transmitted using direct P2P connections or cellular phone-based communications, such as 3G/4G. In the first case, a temporary ad hoc network is established, which has the advantage of not requiring a previously-existing communication infrastructure, but has the drawback of its limited range and duration, which makes the transmission of data to other vehicles beyond that wireless communication range difficult. On the other hand, communications using 3G/4G are much easier, but they need the existence of a cellular communication infrastructure and the payment of a fee to use it.

The processing of queries in a vehicular network is usually performed following one out of two possible approaches. In *push-based approaches*, the data are transmitted by their originators to their nearby neighbors, who can retransmit them again (or not), and in this way the data reach distant places, with the hope that they will arrive at some receiver interested in them; then, the query processing is performed locally by each vehicle, by executing the query over the local knowledge base or local database that stores the data received opportunistically from other vehicles. On the other hand, with *pull-based approaches* the users search actively for the data they are interested in, by disseminating the query through the vehicular network and, after that, retrieving the answer using different techniques.

Most research on query processing in vehicular networks has focused on push-based approaches (e.g., [CDI11]) and only a few works consider pull-based approaches instead (e.g., [DMIH11]). The main reason is higher simplicity: push-based approaches

avoid some challenges that appear when a query is disseminated in a VANET and the results need to be collected and communicated to the query originator vehicle. For example, in order to facilitate the return of a query answer to the query originator vehicle, the usage of mailboxes is proposed in [DMIH11]. Mailboxes are static locations where the data of the query answers can be stored (e.g., roadside units), in such a way that the problem of routing a query answer to a moving query originator disappears; instead, when the driver needs to retrieve an answer, he/she will need to drive towards the mailbox assigned to his/her query or contact it remotely. Another option is the application of mobile agents for query processing in vehicular networks, but it has not been studied in depth, except for the work performed in this thesis. In the following, we present some existing works for the processing of queries in vehicular networks.

In [WXC09], a system called *TrafficMedia* is proposed for querying multimedia clips recorded by vehicles, which store visual and audio information about the traffic conditions. In that system, the vehicles exchange information directly using P2P communications (there is no central node for query processing). However, not only short-range communications such as Wi-Fi, but also cellular communications, are considered. The problem of routing the answer of a query to a mobile query originator is not addressed explicitly, although the authors of that paper are probably assuming that the cellular network is used to communicate the answer. Of course, using a cellular network it is possible to return the results to the query originator more efficiently.

In [LLPG10b], the authors propose a system called *FleaNet*, which is a flea market where sell and buy requests are made by vehicles, pedestrians, and also passengers of public transportation. In that system, the messages are disseminated by sending them only to neighbors that are at k-hop distance, so the dissemination depends heavily on the mobility and density of the nodes. It does not include the possibility of hopping from one node to another to reach other specific places faster, as in our proposal.

In [TWW14], an application scenario similar to ours is presented, although it does not take advantage of mobile agents. In the proposed system, queries are launched on a vehicular network and are routed to a destination area, where the query is solved using data from sensors present in that area. Then, the response is routed back to the query originator, which will usually be a moving vehicle. A method to locate the mobile query originator is proposed, based on what the authors call *breadcrumbs*. These are small pieces of information, stored at certain nodes along the route followed by the originator, that will help to track back the position of the query originator when the response has to be returned.

Other interesting works are [MSN05, ZZC09, XVW09]. *PeopleNet* [MSN05] is an infrastructure-based proposal for information exchange in a mobile environment. *Roadcast* [ZZC09] is a content sharing scheme for VANETs, such that a vehicle can only query other vehicles that it encounters on the way, and the scheme proposed tries to return the most popular content relevant to the query. In [XVW09], a combination of pull and push is considered for *in-network query processing in mobile P2P databases*. In these works, the queries and results are communicated only to the neighboring

vehicles, and therefore the query/result routing problem does not appear (the query originator is always at one-hop distance). Besides, these proposals are not able to process some kinds of queries, since the query originator must move near the nodes which store the information needed.

A survey collecting different methods for routing data in vehicular networks is presented in [BBG13]. This is a very important and also challenging issue in our application scenario, since the success of the query processing depends entirely on how efficiently the query is routed to the interest area and the results returned to the originator. There exist different routing algorithms that can be used depending on a number of scenario conditions, such as the presence or not of *roadside units*, if these roadside units are static or mobile, and if GPRS-like communications are available or only short-range communications (e.g., IEEE 802.11) are allowed.

As opposed to these approaches, in our proposal the routing decisions are encapsulated by mobile agents, which may apply different hop strategies based on the data they receive from the environment.

8.3 Spatial Crowdsourcing and Crowdsensing

The topic of *mobile crowdsensing* (also called *collaborative sensing*, *mobile sensing*, or *participatory sensing*) has attracted significant research attention (e.g., see [GYL11, STZ12, CCC+14, CFZ+14, TGS14, GWY+15, TCS17]). One common problem of spatial crowdsourcing is known as the *Maximum Task Assignment (MTA)* problem, which focuses on how to allocate the available *workers* (in our context, collaborating vehicles) to spatial tasks, to maximize the global performance. There exist many proposals to tackle this problem (e.g., see [TSD+16, TSK15, CLC+15, uHC16, To16, WLL17]) under different circumstances and with different constraints. However, most of them need a central server, which centralizes all the tasks that need to be assigned and which constantly receives the updated locations of the potential collaborators, in order to allocate them the most appropriate task according to several factors (e.g., their positions, travel times to the destinations, etc.). These solutions have the advantage of optimizing the performance of the crowdsourcing process, but they also have some drawbacks, such as privacy concerns (the central server needs to know the positions of all the collaborators) and the need of direct communications with the central server, which can have an economic cost (in case of using 3G/4G technologies) or may need a previously-existing infrastructure (e.g., cellular phone towers or Wi-Fi access points). In our approach, we consider a pure ad hoc solution where a mobile agent directly asks for help to the neighbor vehicles. Thus, we avoid the disadvantages of the centralized approaches. However, the assignments performed in our approach are not necessarily globally optimal, as they are based on local decisions taken without having a global vision of the scenario.

In [WYW+18], the authors propose a method for recruiting collaborators for crowdsensing tasks, by using semi-Markov models to predict the trajectory that the potential collaborators will follow and to estimate if recruiting them may be interesting or not according to the existing sensing needs. In our approach, there is no explicit

recruiting process based on the trajectories of the vehicles, which we assume that can be unknown (e.g., due to privacy reasons): only vehicles in the neighborhood (within the communication range) of the vehicle carrying the mobile agent can participate in the spatial crowdsourcing, and they may accept or not a collaboration request when the agent asks for their help. Nevertheless, the proposal in [WYW+18] could be used as a complementary technique if access to those trajectories is enabled.

In [CCC+14], the authors present a sensing engine called *MoST (Mobile Sensing Technology)*, that can be used as a framework for the development of location-aware applications for Android mobile phones. The data read by the mobile phone sensors are transferred to a central server by using the connection available on the phone (Wi-Fi, 3G/4G, etc.) and processed in the server. This work is not related specifically to VANETs, but it illustrates the interest of sensing and crowdsourcing applications.

In [AHT18], the authors tackle the problem of recruiting vehicles for public sensing in such a way that the selected vehicles cover an area as large as possible at the lowest cost. To achieve this, they present a *reputation-aware, trajectory-based recruitment (RTR)* framework that handles the recruitment of vehicles for public sensing. The framework considers the spatiotemporal availability of participants along with their reputation to select vehicles that should achieve the desired coverage of an area of interest with a cost within the budget limitations. The framework consists of a reputation assessment scheme, a pricing model, and a selection scheme, that are combined to accomplish the objective of maximizing the coverage with the minimum cost. The authors propose greedy heuristic solutions targeting the selection problem in real-time. The RTR framework generalizes the basic selection problem to handle some practical scenarios, including vehicles that leave the area that needs to be monitored, and varying redundancy requirements. An extensive performance evaluation shows that the proposed greedy heuristics are able to achieve results close to those previously obtained by optimal benchmarks under different scenarios, and that the framework succeeds in achieving high levels of coverage even when the vehicles do not stick to their announced trajectories. In our spatial crowdsourcing approach, a mobile agent takes decisions locally based on information regarding the neighbor vehicles, and therefore the collaboration takes place opportunistically rather than through an explicit recruitment process.

It might also be possible to consider more sophisticated crowdsourcing strategies with the goal to minimize the social cost while at the same time ensuring a minimum compensation for the driver. For example, in the context of ride-sharing for passengers, [TLZ+18] provides a method to select a driver to maximize the *overall shared route percentage (SRP)* subject to a minimum required value of this parameter for each driver (*expectation rate of the driver*). This technique could be applied to locate vehicles that, even if they are not at that moment within communication range, could pick up the mobile agent and carry it to the intended destination. However, applying this method in a vehicular ad hoc network, where there is no global view of the environment and the vehicles are usually constantly moving, is challenging.

Other works related to crowdsensing, but further from the specifics of spatial crowdsourcing in this thesis, are [FBG17, QCJ+18]. In [FBG17], the authors pro-

pose GENIUS-C, a framework to support the development of spatial crowdsourcing platforms. It is based on a generic architecture to reduce the gap between academy and industry, and is meant to decrease the development cost and effort and increase the overall quality of spatial crowdsourcing platforms. A case study is created using GENIUS-C to demonstrate its benefits and how it can be used in the development of spatial crowdsourcing platforms. In [QCJ+18], the authors describe an algorithm to detect the accuracy of data that come from crowdsensing sources such as sensors aboard vehicles (car sensors), sensors built into mobile devices (phone sensors) carried by one or more occupants, or both. They demonstrate, through evaluation, that their detection algorithms can achieve high accuracy for some tasks related to the driver's behavior and the environment (e.g., higher than 90% for lane change determination) and that crowdsensing plays an indispensable role in improving the detection performance (e.g., improving recall by 35% for lane change determination on curves).

Besides, although it does not specifically propose a crowdsensing method, another relevant related work to mention is [YLOJ12], where the authors describe a hierarchical Bayesian non-parametric approach for efficient and scalable route prediction, that can harness *the wisdom of crowds* of route planning agents by aggregating their sequential routes of possibly-varying lengths and origin-destination pairs. This approach has the advantages that it does not require a Markov assumption and that it generalizes well with sparse data, thus resulting in an improved prediction accuracy, as the authors demonstrate empirically using real-world data about trajectories of taxis.

Finally, regarding the use of incentives to encourage cooperation, *OPPay* [SQM17] is a payment system for opportunistic data services (such as Wi-Fi sharing, content-based file sharing, and opportunistic networking), which implements a micropayment communication protocol for mobile devices to perform data transactions and make payments using bitcoins. In our approach, we propose the use of virtual money to pay collaborating vehicles, but the specific details about how those payments can be made is out of the scope of this paper. OPPay is intended to operate using incremental payments that are resilient to interruptions in the communications, and therefore it may be of interest in the context of vehicular networks. Other economic and incentive models have been proposed for mobile P2P networks (e.g., [WXS04, YL14]) and collaborative sensing in general (e.g., [JVLR15, GLW+15, ZYS+16, DWSH17]).

8.4 Monitoring Tasks with Vehicles

The monitoring of certain parameters (e.g., polluting gases and particles, pollen, or solar radiation) has been performed traditionally using fixed sensing stations, that have a number disadvantages, such as their lack of flexibility to perform a different monitoring task (e.g., monitor a different area), and the requirement of a certain infrastructure to provide the energy and communication support required, which also make them costly to build and maintain. As an alternative, vehicles can be used as mobile monitoring platforms, since they can be equipped with the necessary sensors

and can travel to any place (as long as a nearby road exists) to monitor the required parameters. The sensors installed can be replaced by others if it is required to monitor different parameters, and the sensors do not need a dedicated power or communication infrastructure, since they can use the power generated by the vehicle's engine and the wireless communication devices of the on-board computer. For these reasons, the use of vehicles for performing monitoring tasks have attracted the attention of the research community, and the term *vehicular sensor network* has emerged [LG10, Vac15]. In the following, we present some works related to this topic.

In [LWZM15], the authors assume the existence of a vehicular network in which the vehicles carry sensors, and they pose the problem of determining how much information should be received from the sensors and where the vehicles should be located to achieve an optimum monitoring process. If the sensors send information very frequently, then their readings would likely be similar, which implies that bandwidth would be wasted. On the other hand, if the sensors send a little amount of information, then the measurements may be not accurate enough. That work is centered on the mathematical aspect of such a method. It is considered that sensors send their data to a central server by using cellular connections instead of ad hoc communications. As stated before, and also in studies such as [IDTL15], this has some disadvantages, such as the existence of economic cost for the user.

In [WJM13], the authors use information about the traffic status to compute the best route to recharging points for electrical vehicles, taking also into account the amount of energy available on board. The traffic status is received from the surrounding cars, that send information about their positions and speeds to estimate if the traffic is light or dense. Ad hoc communications and a simple flooding model for data dissemination are used. However, the proposal focuses only on sharing traffic data. In our proposal, the data sent by the vehicles can be of any type (not only the speeds and positions of vehicles) and the sensed data are carried by mobile agents, which can behave more efficiently than flooding algorithms that continuously send all the data to all the neighbors. So, a mobile agent can encapsulate any desired dissemination approach and even commute from one approach to another depending on context conditions (e.g., the density of vehicles in the area). With the approach presented in this thesis, only a limited amount of data is sent, every time a mobile agent hops from one vehicle to another, instead of sending multiple copies of the data to multiple receivers, as it occurs with flooding algorithms. Besides, the proposal in [WJM13] does not exploit any spatial crowdsourcing approach.

A working prototype is presented in [SCGa+15] that consists of a monitoring VANET where the vehicles are equipped with sensors and wireless communication devices that constantly look for Wi-Fi access points to opportunistically send the collected data to a central server. By contrast, in our approach the data are stored in the vehicles instead of being all sent to a central server, and they are processed distributively by mobile agents looking for the most relevant data to perform the monitoring task. The system presented in [SCGa+15] was deployed in a real city and the vehicles participating in the VANET belonged to the local authority, instead of being open to any citizen, which has both advantages and drawbacks.

In [CDMP16], the authors propose a method for route planning in a city by using information collected opportunistically by the moving vehicles. The system operates in a decentralized way and it exploits a technique based on the ant optimization problem, which leads to routes that are not strictly optimal (as opposed to classic shortest-path search algorithms) but good enough to both reach the intended destination and distribute the traffic flow along different streets so that they do not become congested. While the approach is interesting, its application is limited to the problem of route planning and its performance in low-traffic density scenarios is not clear.

In [GPZA13], a smart parking system for locating available parking slots is proposed. In this system, end users are not the only subjects that collaborate by sharing information. Additionally, both the parking controllers and the city administrators also participate by providing information in a fast and integrated way. However, the system operates in a centralized way, and a mobile 3G/4G connection is required to operate it.

In [ZMLT18], the authors present a unified delay analysis framework for opportunistic data collection, that integrates the *sensing and transmission delays*, that are usually analyzed separately. A third delay metric (that they call *data collection delay*) is added to the analysis that can be performed by the framework, and thus the QoS of opportunistic data collection applications can be measured more comprehensively. The theoretical analysis of the framework is validated by the authors by performing a number of simulations. This work is complementary to ours, as it focuses on the analysis of performance metrics.

Also related to delay aspects, in [LCX+17], the authors tackle the problem of the sensor's reading delay, and they propose a solution to mitigate it. Certain types of sensors (for example, some gas sensors) take a relatively long time (about 20 to 50 seconds) to reach a stable state and obtain a valid sample measurement. If these sensors are carried by moving vehicles, they may have moved several hundred meters in that lapse, thus obtaining data from a place which is distant from the intended target. If this effect is not taken into account, the measured parameters may be treated incorrectly and they could lead to wrong conclusions. To avoid this, the authors propose a method to calibrate the sensor readings by using a filter on the raw data provided by the sensors, which also has as a benefit the reduction of the time needed to obtain valid sensors readings.

Finally, a recent work [WZZ+16] analyzes crowdsourcing in ITS, but it does not focus on vehicular networks nor spatial crowdsourcing. As another example, [ZLJ+17] proposes a privacy-preserving vehicular urban sensing platform; however, it relies on a cellular network, rather than using ad hoc communications, and focuses on privacy issues and not on the use of spatial crowdsourcing.

8.5 Searching for Parking Spaces

Searching for an available parking space is a time-consuming task for drivers and it also has a high environmental cost due to the fuel that is wasted. Therefore, as a representative case study for the work performed in this thesis, in Chapter 7, we have

chosen a scenario where a driver asks in advance for information about one or more available parking spaces in a city area. The driver receives a list of such spaces that he/she can try to reach, instead of having to look around searching for them, and thus potentially saving time and fuel. It is likely that some of the suggested parking spaces are occupied by other drivers while the user reaches them, but although the proposed system is certainly not perfect, it can still be helpful to their users.

Helping drivers to find parking spaces is a topic that has received considerable research attention. So, *smart parking systems* can be designed and deployed to provide information about parking spaces in specific areas (e.g., see [KcSH17] for a recent review). However, these infrastructure-based solutions are quite expensive and not available globally. On the contrary, it would be interesting to have solutions that are flexible enough to obtain information about any available on-street parking. This motivated the development of proposals that exploit data dissemination in VANETs, such as [CGM06, DILC13]. Information about the availability of parking spaces can be quite volatile, and therefore some proposals try to take this into account. For example, the work presented in [CGM06] emphasizes the interest of guiding drivers towards areas where the probability to find an available parking space is high, rather than towards a specific parking space that may become occupied in a short time. In [DILC13], an allocation protocol is proposed for sharing information about available parking space using only ad hoc communications. This proposal guarantees that the information about a parking space is provided only to a single interested driver, which avoids competition problems that arise if several drivers receive an alert about the same parking space. Several approaches related to searching parking spaces are described in [DILC13, IDTL15]. In the following, we describe a few related works related to the problem of finding parking spaces using a VANET; it should be noted that a couple of related approaches have already been cited previously: [VS11] in Section 8.1 and [ZMLT18] in Section 8.4.

In [KLW14], the authors address the problem of predicting the number of available parking spaces in a parking lot. The parking lot is modeled by a continuous-time Markov chain, and it regularly communicates the number of occupied spaces, capacity, arrival, and parking rate, by disseminating that information through a vehicular network. The vehicles that receive those data compute the probability of an available parking space upon arrival, by applying the method proposed by the authors. Thus, the driver can choose, among different parking lots, the one which will most probably have more available spaces. This work only considers parking lots, and therefore it cannot be used for on-street parking spaces.

In [SJLF17], a smart parking system is proposed, in which a network of sensors monitors the availability of parking spaces in several parking lots, and sends those data using Narrowband Internet of Things (NB-IoT) modules, which is a new cellular technology introduced for Low-Power Wide-Area (LPWA) applications. The basic information management, sensor node surveillance, charge management, task management and business intelligence modules are implemented in a cloud server. With an integrated third-party payment platform and parking guide service, the mobile application developed for drivers is easy and convenient to use. The proposed system

has been deployed in two cities to improve the utilization of existing parking facilities effectively. This work uses a communication technology that requires the installation of an infrastructure, although it is optimized for its use in mobile devices, thanks to its low power requirements. Besides, it is focused on the management of parking lots and the payment of their services, which also differs from our proposal, which is more generic.

In [MDLW18], the authors propose a parking allocation model where parking lots are assigned to vehicles. The vehicles are assumed to be connected and can exchange information with a central management system. The vehicle's arrival times can be provided by a GPS device, and the estimated number of available parking slots, at each future time moment and for each parking lot, is used as an input. The author's initial model is static and may be viewed as a variant of the generalized assignment problem. However, the model can be re-run, and the algorithm can handle dynamic changes by frequently solving the static model, each time producing an updated solution. In practice, this approach is feasible only if reliable quality solutions of the static model are obtained within a few seconds, since the GPS can continuously provide new input regarding the vehicle's positioning and its destinations. The authors propose a 0–1 programming model to compute exact solutions, together with a variable neighborhood search-based heuristic to obtain approximate solutions for larger instances. This work proposes a centralized solution where the vehicles send their updated positions to a server (using a 3G/4G connection), that allocates specific parking places from lots to vehicles, and can adapt dynamically to changes in their positions. On the other hand, our proposed solution is decentralized and exploits the information exchanged among nearby vehicles. The communications in our solution are P2P and multi-hop, and the proposal can be used for both parking lots and street parking.

In [AWD+18], the authors present a search problem where mobile users are searching for static resources. Each user wants to obtain exactly one resource, and both users and resources are spatially located on a road network, which also constrains the movements of the users. This problem can be applied to various transportation applications; for example, vehicles searching for available parking spaces or taxicabs searching for clients to pick up. The authors model the problem in different scenarios that vary based on the level of information that is available to the users. They range from cases where the users have complete information about other users and resources to those where the users only have access to a fraction of the data about the availability of resources (i.e., there are uncertain data). The authors also propose pricing schemes that incentivize vehicles to search for resources in a way that benefits the system and the environment. The proposed algorithms are tested in a simulation environment that uses real-world data, and they are able to attain up to 40% of improvements over other approaches. In this work, some communication mechanism (e.g., cellular-based or P2P communications) is assumed to exist between the users and the information sources, and the work is focused on the resource allocation problem itself. In our approach, we focus on how to obtain the available parking information, and we do not tackle the problem of resource allocation among several users.

Up to the authors' knowledge, existing proposals to provide information about available parking spaces are push-based, and therefore they cannot be used in scenarios like the one presented as a sample case study in this thesis, where the driver needs to retrieve information about parking spaces in areas that are not nearby. Moreover, the use of mobile agents to search available on-street parking spaces is also new. Finally, the experimental evaluation of approaches for VANETs usually focus either on urban scenarios or highways, whereas in the evaluation of the case study presented in Section 7 we have considered a mixed scenario that includes both an urban area and an inter-urban region.

8.6 Summary of the Chapter

In this chapter, we presented relevant works in areas related to our data management approach for VANETs. Firstly, we reviewed some works that use mobile agents in vehicular networks for a variety of tasks. Secondly, we described works related to query processing in VANETs, where information about the traffic or other topics are transmitted to other users in the area, whether they are traveling in vehicles or not. Thirdly, we presented some other works about the topic of spatial crowdsourcing and crowdsensing, where collaborating users help by working on small portions of bigger tasks related to visiting certain places or taking measures of certain parameters in those places. After that, we reviewed works where vehicles are used as mobile sensing platforms to perform monitoring tasks in relatively wide areas. Finally, we described some works related to the parking space use case scenario.

Chapter 9

Conclusions

In this chapter, we present some conclusions about the work presented in this thesis. Firstly, we summarize our contributions to the field of data management in vehicular networks. After that, we evaluate the results obtained. Finally, we point out some open lines of work for future research.

In this thesis, we studied the use of mobile agents in vehicular networks and we developed an approach for distributed data management in such type of networks. To accomplish this, we used mobile agents that can move to any location in the network by using different techniques, such as different hop strategies or spatial crowdsourcing. Thanks to their mobility, different actions can be performed on the existing data, such as their processing, filtering, and/or transportation to other places. As part of this research work, we performed the following tasks:

- We developed and evaluated experimentally a method that allows a mobile agent to move anywhere in a VANET by using different hop strategies.
- Motivated by the limitations of existing simulation software regarding the simulation of mobile agents, we developed our own simulator (MAVSIM) to help us in the development, evaluation and testing of different data management strategies involving mobile agents and vehicular networks.
- We developed a complete distributed data management approach based in a four-step process for monitoring and retrieving data located in a certain geographic area in the context of a vehicular network. This approach uses mobile agents for all the main tasks, such as locating, filtering, processing and moving the data from/to any static or moving location.
- We studied and evaluated the enhancement of this approach by applying spatial crowdsourcing techniques in such a way that mobile agents use the help of collaborators to perform their task more efficiently and faster.

As far as we know, no existing work addresses the problem of data management in vehicular networks in such a generic and flexible way as we propose, since the

proposed approach can be used for the management of any type of data (e.g., sensor data, data coming from other vehicles or roadside units, etc.) and in any type of vehicular scenario (i.e., highways, interurban roads, and different layouts of urban streets). Moreover, this work is also novel in exploring the possibility of building such a system using the technology of mobile agents, which eventually proved to be highly flexible and adequate enough for accomplishing the required tasks.

9.1 Main Contributions

In the following sections, we summarize in more detail the main contributions of this work.

9.1.1 A Data Management Approach Using Mobile Agents

The proposed data management approach allows the exploitation of the data that are present in a VANET. These data are usually scattered, unorganized and with intermittent availability due to the movements of their holders (i.e., the vehicles) and the limited-range wireless communications. Thanks to the method developed, it is possible to perform queries about those data, which involves a number of steps:

- Firstly, it is necessary to define what data the users want to query.
- Secondly, it is necessary to go where the data are stored, which in a vehicular network are the vehicles traveling within an area. For this step, we use mobile agents, thanks to their abilities to move wirelessly in a changing and dynamic environment such as a VANET.
- Thirdly, the relevant data must be located and aggregated to compose a complete solution. These data are distributed among the vehicles in the interest area and, again, mobile agents are used for this task, not only due to their moving abilities, but also because they are able to process the data stored locally in the vehicles, filter out the data that are irrelevant to the query, and carry only the relevant data.
- Finally, once the requested data have been located and gathered, they are returned to the user who initiated the query, by means of mobile agents.

This approach has the advantage of being very generic, and therefore it can be used for the development of a variety of applications and services on top of it. Regarding its generality, we could highlight the following features:

- It can use any communication technology, both wireless (e.g., Wi-Fi, Bluetooth, Ultra Wide-Band, etc.) or wired, as long as it allows the transfer of mobile agents.
- It can operate in any kind of device with computing and communication capabilities, as long as it can execute the mobile agent platform. This includes devices such as smartphones, tablets, laptops, etc.

- Any data acquired and stored in the vehicles can be queried, whatever their nature. For example, it could be data from sensor readings (temperature, air pressure, noise level, etc.), data obtained from the environment by on-board cameras, data received from other vehicles via Wi-Fi, etc. Moreover, these data can be stored in the vehicles' computer devices in either a structured format (e.g., relational or NOSQL databases) or an unstructured way (e.g., plain files). If there exists some program or algorithm that can access those data, then the mobile agent moves to where the data are stored and processes them in situ without the need of transferring or carrying all of them.

- The data to be queried can be present in any place or area of any size, and the position, size and even the shape of that area can be static or change as time passes. For example, the data queried can be limited to the streets of the city center, to the vehicles traveling along a certain road in a certain direction, to a circle with a radius of 1 km centered around a truck carrying dangerous merchandise, etc. Thanks to the intelligence encoded in a mobile agent, it is able to know at every moment if the place where it is currently located belongs or not to an arbitrarily-complex interest area.

- Similarly, and regarding the temporal dimension, the queries can refer to data stored in a certain past date or interval (as long as those data have not been deleted and are still stored in some nodes of the network), or to present data. In fact, it could even refer to future data, in which case the mobile agent could be *wandering* near the interest area in advance, waiting for a certain date or hour to start the task of locating, processing and querying the interest data.

- This data management strategy was developed to be effective in a VANET, which is a very challenging environment, but it could also be used in other scenarios with moving nodes (such as a MANET) or static nodes (such as a sensor network), and with different kinds of communications and data. The principles of the data management approach would be very similar and mobile agent technology could be used in the same way.

As a conclusion, we believe that, thanks to its generality, the proposed approach (which is not tied to any specific technology or hardware platform) can be used to develop interesting applications in the context of data management in mobile and dynamic scenarios.

9.1.2 Hop Strategies

The transfer of data between two distant places (i.e., farther than what is directly reachable) or vehicles in a VANET is problematic, since multi-hop protocols are necessary, that use intermediate vehicles as relays to resend the data from one to another until the destination is reached. Therefore, we proposed the use of mobile agents to accomplish this task, due to their ability to move wirelessly from one execution device to another in such a constantly-changing environment.

However, in order to complete the routing of the transmitted data to the desired destination, it is necessary that the mobile agents be programmed to follow some algorithm or heuristic function that helps them to estimate if a certain neighbor vehicle will bring them closer or farther from their destinations. We call these heuristics *hop strategies* and we proposed a number of them that assume the availability of a different amount of information to help a mobile agent to take a decision.

These hop strategies were extensively evaluated using a simulator, under different conditions of traffic density, network reliability, geometry of the streets, presence or absence of signal-blocking obstacles, distance from the origin to the destination, availability and relevance of data, etc. Additionally, along with the evaluation of every hop strategy, their working details were fully disclosed and explained, including the previous necessary knowledge needed by them, that can be as simple as the position of the vehicles in relation to the destination, or as complex as the intended route to be followed by the vehicle's driver.

As a result, for every evaluated hop strategy we obtained measures such as the total time spent by a mobile agent to finish its travel, the number of hops (and therefore, the bandwidth usage) needed, the effective speed of the process, etc.

These results, as well as the amount of information needed by each hop strategy, can help to decide which of them would be the best one to use according to the conditions and resources available.

9.1.3 Spatial Crowdsourcing

After evaluating the hop strategies previously described, we found that they performed acceptably well in most conditions. However, there exist some cases where their behavior could be enhanced.

With the aforementioned hop strategies, the mobile agent can easily reach the places crossed by frequently traveled roads, since the agent can hop more frequently to different vehicles and the probability of finding one that travels towards the destination increases. However, it would be more difficult for the mobile agent to visit places whose roads are not frequently traveled. Note that, in this case, we are not talking about low vehicle density, but instead about infrequently-traveled roads. In the first case, a road can have a small number of vehicles crossing it (e.g., one vehicle per kilometer), but sooner or later it will be traversed. In the second case, a certain street or a stretch of road can be barely traveled by any vehicle, independently of the traffic density in the area.

In such cases, it is necessary to use methods different from the basic hop strategies. Specifically, in order to find a solution to this problem, we explored the use of spatial crowdsourcing techniques, which use the help of collaborating drivers so that the mobile agent can be explicitly carried to the required place, or at least near it. Since helping the mobile agent takes some cost for the collaborators, the collaboration is assumed to be compensated in order to encourage the drivers to help agents. We performed many experiments exploring the application of these techniques to the movements of mobile agents through a VANET.

One of the conclusions drawn is that a crowdsourcing-only strategy is not adequate, since despite its high reliability to take the mobile agent nearer its destination, it also has some drawbacks. Firstly, the mobile agent tends to travel physically in the vehicles, which is much slower than using the wireless communications to transfer itself from one vehicle to another. Secondly, the compensating costs for the collaborators may grow a lot, since the amount to pay is proportional to the duration of the trip in the collaborator. Therefore, it is necessary to use a mixed solution, where a mobile agent uses by default the initial hop strategies and switches to using spatial crowdsourcing when they perform poorly. This solution proved, after being extensively evaluated, to be more efficient than using the basic hop strategies or only the spatial crowdsourcing strategies independently.

9.1.4 Simulation of VANETs for the Evaluation of Data Management Strategies Using Mobile Agents

When we started to explore the data management techniques described above, and particularly the hop strategies used by the mobile agents, we realized that the existing simulation software did not meet some special needs that were essential to properly test and evaluate these strategies.

The main drawback of existing simulators is that they are not capable of simulating at the same time the movement of the vehicles and the mobile agents. The closest functionality that can be obtained is by means of combining different types of simulators, by exporting and importing among them the data created in the simulations following a multi-step process. Moreover, even following this procedure, it is not possible to define scenarios where the behavior of the vehicles (e.g., the routes they follow) could be influenced by the information provided by the mobile agents, which would certainly be an interesting feature for the development of applications related, for example, to traffic security, searching of parking spaces. This is also essential for the evaluation of spatial crowdsourcing strategies, where the collaborating drivers can alter their routes to help a mobile agent.

Due to these limitations, we developed our own software, that meets our especial requirements regarding the simulation of VANETs and mobile agents, and also contributes to cover the existing gap in this topic. The main features of MAVSIM are:

- It is portable (written in Java), so it can be executed on a variety of computers and architectures, from a Mac laptop running macOS, to an x86-64 workstation or a SPARC-based server.

- It can execute simulations in both graphic and batch mode. This last feature is especially useful for launching a high number of simulations, that can also use any existing job scheduler, such as HTCondor [HTC], Torque [TOR], or Slurm [SLU], which allows a more efficient use of computing resources.

- The whole simulations can be recorded to data files, that can be replayed later

using a graphic tool that allows to follow step by step the status of the simulated objects (i.e., the vehicles and the mobile agents).

- It is highly configurable, so it can execute simulations in a high variety of scenarios and under different conditions.
- It supports using real world maps (of any city or interurban area) from the OpenStreetMap site, so the simulations can take place in realistic places.
- It can simulate a high number of both mobile and static elements: vehicles that follow different mobility models, public transport vehicles that follow fixed routes, roadside units or antennas at fixed places that can communicate among them instantly using simulated wired communications and, of course, mobile agents that can transfer themselves to any simulated device or be physically transported by any vehicle.
- It can also simulate some limiting factors, such as the blocking of wireless signals by buildings or the interruption of communications by unexpected errors. This helps the simulation to be more realistic and challenging, just like real VANETs.
- It can simulate many instances and types of mobile agents simultaneously, that can be programmed in a similar way as they would be developed using a conventional mobile agent platform such as JADE [BPR01] or SPRINGS [ITLM06].

As a result, thanks to the use of this simulator, the development and testing of the proposed data management approaches was possible in all their extent. The simulations performed provided a high amount of data and details about all the processes followed by the mobile agents and the other simulated objects. For example, the simulator provides information about the amount of data gathered by an agent along the simulated time in the query solving scenario, the speed and heading of every vehicle in range just before evaluating them as potential carriers, the distance traveled by the vehicles, the mobile agent's remaining distance to reach its destination at every time instant, etc.

We think that the simulator developed is quite well suited for testing applications, protocols and systems that require the use of mobile agents in vehicular networks, taking into account different elements that can have an influence on their performance. The simulator is written in a modular and extensible way, so it can be enhanced with new features with little effort. We continue using the simulator in our research, and we are constantly adding new features when we need them to evaluate different data management strategies. Thanks to the use of this simulator, we have obtained promising results regarding the potential interest of using mobile agents in vehicular networks.

9.1.5 Experimental Evaluation

Our contributions have been tested through an extensive set of experiments, that have shown the following:

- It is possible to build a data management strategy in a VANET, despite the difficulties caused by ad hoc communications.
- The use of mobile agents helps to solve many of the problems that occur in a highly-dynamic and changing environment like a VANET.
- Mobile agents are real practical software and not only a theoretical computational concept. They can actually be executed in real and heterogeneous devices that use currently available technology, and transfer themselves among them by using wireless communications.
- The application of the proposed data management strategy for the distributed processing of queries works well even in scenarios with a high number of limitations, such as low traffic density, communication interruptions or changes in the route of the vehicles, etc. Despite these difficulties, mobile agents can, in general, end the process in reasonable times, with a high reliability and an efficient usage of the available bandwidth.
- New forms of joining efforts in a collaborative way, such as using crowdsourcing (or, more specifically, spatial crowdsourcing methods) can be applied successfully in order to enhance the process.

Summing up, the extensive experimentation performed shows that the proposed data management has a good performance even in challenging scenarios, thanks to the use of mobile agents, whose special features are suitable to face the difficulties of using a VANET.

9.2 Evaluation of Results

The results of this work have been published, along its development, in different international journals and conferences. These are the main publications obtained in relation to this thesis, in chronological order:

- In [UIM08], we presented results of evaluating the execution of mobile agent platforms in real mobile devices. We measured the performance of mobile agent movements and calls using different communication technologies, both wired and wireless.
- In [UIM09], we discussed the ideal properties that mobile agent platforms for mobile devices should have, and performed a comparative analysis of some existing platforms to know to what extent they met the requirements or not.
- In [UITLM09], we compared the features of different mobile agent platforms capable of being executed on a mobile device, and evaluated the performance of a proof of concept demo that involved a mobile agent searching data in different devices.

- In [UIDM09], we presented the novel idea of using mobile agents in a VANET for monitoring the environment, as an alternative (or complement) to traditional and costly fixed and dedicated monitoring stations.

- In [UIDM10], we discussed the advantages and difficulties of the development of applications in VANETs, and advocated the use of mobile agents as an adequate option to deal with such difficulties.

- In [UI13], we presented a generalization of the idea of environment monitoring, leading to the processing of data in general (not only related to the monitoring of the environment) in VANETs using mobile agents.

- In [UI16b], we presented for the first time the idea of applying spatial crowdsourcing techniques for data management in VANETs.

- In [UITL17a], we presented our query processing approach in detail, and we evaluated exhaustively all the aspects of the solution in different scenarios. This work was one of the three already-published papers selected, due to its quality and impact, as examples of relevant research performed by the JISBD (Software Engineering and Databases) community in Spain, for presentation in the hall of fame of JISBD 2017 [UITL17b].

- In [UI17a], we presented a use case of the data management approach, focused on an application to find available parking places in a city area, with the novelty of considering a mixed urban/interurban scenario and using mobile agents.

- In [UI17b], we described how the distributed query processing in a VANET using mobile agents could be modeled with Petri nets.

- In [UI], we described in detail the spatial crowdsourcing approach and evaluated it extensively, extending the initial work presented in [UI16b].

During the development of this thesis, we also presented some intermediate results in national conferences and events, such as [UIL14], [UI14] and [UITL17b], and we were also invited to write some book chapters about topics related to mobile agents and the simulator developed [UITLM12], [UITLM15], [UI16a].

Concerning other parameters that measure our contributions, our papers were referenced in other works, such as [FÁO+10], [EAAD12], [Rom12], [CN11], [NCN11], [Hua11], [SW12], [NCN12], [SSTK13], [SL13], [MCN14], [MS14], [KM14], [CV15], [Hu15], [Kha15], [TW16], [GLL16], [HSY17], [CCA+17], [BH17], [BHS17], [Lep18], where we can highlight references from some journals such as Vehicular Communications [KM14] and Ad Hoc Networks [TW16]. As an example, the authors of [CCA+17] refer to our work in the following terms: "*To this aim we followed the strategy proposed by Urra et al. [1], who provide a five-step process for general-purpose monitoring using vehicular networks*".

9.3 Future Work

Developing applications for VANETs is an interesting topic with a high potential, and our data management approach using mobile agents has shown to be an adequate option to support such applications. In order to enhance the approach, a number of improvements could be performed:

- Explore the use of mobile agents that can collaborate and communicate among them, so that they help each other to perform their tasks earlier.
- Also related to the former, explore the opposite situation, where many mobile agents have the same goal and compete for being the first to achieve it. The problem would be how to deal with these situations automatically in a fair and efficient way.
- Create new advanced hop strategies that, for example, make use of historical or statistical data about patterns of the driving style of the drivers, or about the traffic in a certain place according to the day of the week or the time of the day, to cite some examples.
- Explore the use of the data management approach in vehicular networks that also include *drones* or UAVs (*unmanned aerial vehicles*), that could be used not only as intermediate nodes for the mobile agents but also to transport them physically through the air.
- Build a working prototype using real devices aboard vehicles as well as personal mobile phones and desktop computers. The mobile agents could then accompany the user, or travel freely to other devices to perform some tasks, making the creation of entirely new types of applications possible. Defining suitable proof-of-concept scenarios and adjusting the developments for their use in real environments is not an easy task.
- Regarding the spatial crowdsourcing topic, design a compensation mechanism based on a cryptocurrency. Instead of the usual *proof of work* typical of most of these currencies, the new one would be based on *proof of collaboration*.
- Enhance the MAVSIM simulator with new types of scenarios and elements that would make the task performed by the mobile agents more interesting. For example, the addition of traffic jams could have benefits and drawbacks, since the movements among the vehicles could be easier if there are a great number of them very close to each other and moving at very low speeds. It would also be very interesting to simulate that the vehicles, from time to time, park at some places and remain stopped for hours or days. This could be used, for example, to temporarily convert such vehicles in fixed places where they can offer some services that require some spatial stability (i.e., that the node does not keep moving), for example, to act as a *mailbox* where mobile agents can leave messages or small amounts of data, so that other agents operating nearby could read or use them.

- Other operative enhancements for MAVSIM, such as improving the replay tool, so that it can show more information about a recorded simulation. Another enhancement would be the usage of GPUs to parallelize and accelerate some mathematical calculations, and thus increase the simulation speed.

As a conclusion, the goal of our future lines of research will be to continue exploring the challenges related to the development and use of data management strategies for VANETs. As a part of this future research, specific novel applications could also be considered.

Relevant Publications Related to this Thesis

International Journals

[UITL17a] O. Urra, S. Ilarri, and R. Trillo-Lado. An approach driven by mobile agents for data management in vehicular networks. *Information Sciences*, 381:55–77, March 2017.

[UI] O. Urra and S. Ilarri. Spatial crowdsourcing with mobile agents in vehicular networks. 37 pages according to the original authors' submission file. Manuscript submitted for publication, 2018.

[UITLM09] O. Urra, S. Ilarri, R. Trillo-Lado, and E. Mena. Mobile Agents and Mobile Devices: Friendship or Difficult Relationship? *Journal of Physical Agents. Special Session on Practical Applications of Agents and Multiagent Systems*, 3(2):27–37, May 2009.

International Conferences

[UI17a] O. Urra and S. Ilarri. Dear Mobile Agent, Could You Please Find Me a Parking Space? In *21st East-European Conference on Advances in Databases and Information Systems (ADBIS)*, volume 767 of *Communications in Computer and Information Science (CCIS), New Trends in Databases and Information Systems*, pp. 82–90. Springer, September 2017.

[UI16b] O. Urra and S. Ilarri. Towards Spatial Crowdsourcing in Vehicular Networks Using Mobile Agents. In *20th East-European Conference on Advances in Databases and Information Systems (ADBIS)*, volume 637 of *Communications in Computer and Information Science (CCIS), New Trends in Databases and Information Systems*, pp. 88–95. Springer, August 2016.

[UIDM10] O. Urra, S. Ilarri, T. Delot, and E. Mena. Mobile Agents in Vehicular Networks: Taking a First Ride. In *Eighth International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, volume 70 of *Advances in Intelligent and Soft Computing*, pp. 118–124. Springer, April 2010.

[UIM09] O. Urra, S. Ilarri, and E. Mena. Agents Jumping in the Air: Dream or Reality? In *10th International Work-Conference on Artificial Neural Networks (IWANN), Special Session on Practical Applications of Agents and Multi-Agent Systems*, volume 5517 of *Lecture Notes in Computer Science (LNCS)*, pp. 627–634. Springer, June 2009.

[UIDM09] O. Urra, S. Ilarri, T. Delot, and E. Mena. Using Hitchhiker Mobile Agents for Environment Monitoring. In *Seventh International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, volume 55 of *Advances in Intelligent and Soft Computing*, pp. 557–566. Springer, March 2009.

International Workshops

[UI17b] O. Urra and S. Ilarri. Modeling Mobile Agents in Vehicular Networks. In *International Workshop on Modeling and Software Engineering in Business and Industry (MoSeBin) at the International Workshop on Petri Nets and Software Engineering (PNSE), co-located with the 38th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets (PN) and the 17th International Conference on Application of Concurrency to System Design (ACSD)*, volume 1846, pp. 233–238. CEUR Workshop Proceedings, June 2017.

[UI13] O. Urra and S. Ilarri. Using Mobile Agents in Vehicular Networks for Data Processing. In *14th International Conference on Mobile Data Management (MDM), PhD Forum*, volume 2, pp. 11–14. IEEE Computer Society, June 2013.

[UIM08] O. Urra, S. Ilarri, and E. Mena. Testing Mobile Agent Platforms Over the Air. In *First Workshop on Data and Services Management in Mobile Environments (DS2ME) in conjunction with the 24th International Conference on Data Engineering (ICDE)*, pp. 152–159. IEEE Computer Society, April 2008.

Book Chapters

[UI16a] O. Urra and S. Ilarri. MAVSIM: Testing VANET Applications Based on Mobile Agents. In *Cognitive Vehicular Networks*, pp. 199–224. CRC Taylor and Francis Group, 2016.

[UITLM15] O. Urra, S. Ilarri, R. Trillo-Lado, and E. Mena. Mobile Agents for a Mobile World. In *Handbook of Research on Innovations in Systems and Software Engineering*, pp. 664–681. IGI Global, 2015.

[UITLM12] O. Urra, S. Ilarri, R. Trillo-Lado, and E. Mena. Mobile Software Agents for Mobile Applications. In *Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications*, pp. 725–740. IGI Global, May 2012.

National Conferences

[UITL17b] O. Urra, S. Ilarri, and R. Trillo-Lado. Una Aproximación Basada en Agentes Móviles para la Gestión de Datos en Redes de Vehículos ("An Approach Driven by Mobile Agents for Data Management in Vehicular Networks"). In *XXII Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, p. 1. September 2017. Relevant Work (artículo relevante ya publicado / Salón de la Fama - "Hall of Fame"); Handle: 11705/JISBD/2017/082.

[UIL14] O. Urra, S. Ilarri, and E. López. Simulating Mobile Agents in Vehicular Networks. In *XIX Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, pp. 71–84. September 2014. Handle: 11705/JISBD/2014/010.

Other Publications and Events

[UI14] O. Urra and S. Ilarri. Distributed data processing in vehicular networks using mobile agents. *Actas de las III Jornadas de Jóvenes Investigadores del I3A, ISSN 2341-4790*, 2:61–62, June 2014.

Bibliography

[Adm16] N. H. T. S. Administration. 2015 motor vehicle crashes: Overview. Traffic safety facts research note, August 2016.

[AFF16] M. Alam, J. Ferreira, and J. Fonseca. Introduction to Intelligent Transportation Systems. In *Intelligent Transportation Systems*, pp. 1–17. Springer, January 2016.

[AFJ+06] C. Andersson, D. Freeman, I. James, A. Johnston, and S. Ljung. *Mobile media and applications, from concept to cash: Successful service creation and launch*. John Wiley & Sons, 2006.

[AGM07] J. G. Andrews, A. Ghosh, and R. Muhamed. *Fundamentals of WiMAX: Understanding broadband wireless networking*. Pearson Education, 2007.

[AHT18] S. Abdelhamid, H. S. Hassanein, and G. Takahara. Reputation-Aware, Trajectory-Based Recruitment of Smart Vehicles for Public Sensing. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1387–1400, May 2018.

[All17] Allied Business Intelligence, Inc. Smart Home & Car Integration Brings Opportunity but Complexity for Car Manufacturers. `https://www.abiresearch.com/press/smart-home-car-integration-brings-opportunity-comp/`, November 2017. [Accessed May 14, 2018].

[Appa] Apple Inc. macOS Official Site. `https://www.apple.com/macos`. [Accessed May 14, 2018].

[Appb] Apple Inc. Siri. `https://www.apple.com/ios/siri/`. [Accessed May 14, 2018].

[Ara] Arada Systems. 802.11p Wireless Systems Products. `http://www.aradasystems.com/products/`. [Accessed May 14, 2018].

[Aso] Asociación de Constructoras y Concesionarias de Infraestructuras. Via-T Sistema de Telepeaje. `http://www.viat.es/`. [Accessed May 14, 2018].

[ASSC02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114, November 2002.

[AWD+18] D. Ayala, O. Wolfson, B. Dasgupta, J. Lin, and B. Xu. Spatio-Temporal Matching for Urban Transportation Applications. *ACM Transactions on Spatial Algorithms and Systems*, 3(4):11:1–11:39, May 2018.

[Axe15] J. Axelson. *USB complete: The developer's guide.* Lakeview research LLC, 2015.

[AYC18] S. Allani, T. Yeferny, and R. Chbeir. A scalable data dissemination protocol based on vehicles trajectories analysis. *Ad Hoc Networks*, 71:31–44, March 2018.

[Bar15] W. Barfield. *Fundamentals of wearable computers and augmented reality.* CRC Press, 2015.

[BBG13] S. M. Bilal, C. J. Bernardos, and C. Guerrero. Position-based routing in vehicular networks: A survey. *Journal of Network and Computer Applications*, 36(2):685–697, March 2013.

[BCOS08] A. Bonifati, P. K. Chrysanthis, A. M. Ouksel, and K.-U. Sattler. Distributed databases and peer-to-peer databases: Past and present. *SIGMOD Record*, 37:5–11, March 2008.

[BDG+00] J. Benedicto, S. Dinwiddy, G. Gatti, R. Lucas, and M. Lugert. GALILEO: Satellite system design. *European Space Agency*, November 2000.

[BDN01] L. Bettini and R. De Nicola. Translating strong mobility into weak mobility. In *International Conference on Mobile Agents*, pp. 182–197. Springer, December 2001.

[BEH04] J. J. Blum, A. Eskandarian, and L. J. Hoffman. Challenges of Intervehicle Ad Hoc Networks. *IEEE Transactions on Intelligent Transportation Systems*, 5(4):347–351, December 2004.

[BFS+02] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang. The case for cyber foraging. In *10th workshop on ACM SIGOPS (EW)*, pp. 87–92. ACM, July 2002.

[BG16] R. Balasubramonian and B. Grot. Near-data processing. *IEEE Micro*, 36(1):4–5, January-February 2016.

[BGF+13] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, et al. Road side unit deployment: A density-based approach. *IEEE Intelligent Transportation Systems Magazine*, 5(3):30–39, July 2013.

[BH17] P. Bagga and R. Hans. Mobile agents system security: A systematic survey. *ACM Computer Surveys*, 50(5):65:1–65:45, September 2017.

[BHS17] P. Bagga, R. Hans, and V. Sharma. A Biological Immune System (BIS) inspired Mobile Agent Platform (MAP) security architecture. *Expert Systems with Applications*, 72:269–282, April 2017.

[Bit] BitTorrent Inc. BitTorrent. `https://www.bittorrent.com/`. [Accessed May 14, 2018].

[BM05] S. Biswas and R. Morris. ExOR: Opportunistic multi-hop routing for wireless networks. *ACM SIGCOMM Computer Communication Review*, 35(4):133–144, August 2005.

[BMS00] E. Bicho, P. Mallet, and G. Schöner. Target representation on an autonomous vehicle with low-level sensors. *The International Journal of Robotics Research*, 19(5):424–447, May 2000.

[BMT+05] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: Collaborative P2P search. In *31st International Conference on Very Large Data Bases (VLDB)*, pp. 1263–1266. VLDB Endowment, October 2005.

[BMW17] Y. Bai, Y. Mai, and N. Wang. Performance comparison and evaluation of the proactive and reactive routing protocols for MANETs. In *2017 Wireless Telecommunications Symposium (WTS)*, pp. 1–5. IEEE Computer Society, April 2017.

[BMZA12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *First edition of the MCC workshop on Mobile cloud computing*, pp. 13–16. ACM, August 2012.

[BN02] L. Bettini and R. D. Nicola. Translating strong mobility into weak mobility. In *Fifth International Conference on Mobile Agents (MA)*, pp. 182–197. Springer, May 2002.

[BPR01] F. Bellifemine, A. Poggi, and G. Rimassa. JADE: A FIPA2000 compliant agent development environment. In *Fifth International Conference on Autonomous Agents*, pp. 216–217. ACM, June 2001.

[BS] I. Bluetooth SIG. Bluetooth technology website. `http://www.bluetooth.com`. [Accessed May 14, 2018].

[BWFS14] M. T. Beck, M. Werner, S. Feld, and S. Schimper. Mobile edge computing: A taxonomy. In *Sixth International Conference on Advances in Future Internet (AFIN)*, pp. 48–55. International Academy, Research, and Industry Association, November 2014.

[CCA+17] C. T. Calafate, K. Cicenia, O. Alvear, J. C. Cano, and P. Manzoni. Estimating rainfall intensity by using vehicles as sensors. In *Wireless Days*, pp. 21–26. IEEE Computer Society, March 2017.

[CCC+14] G. Cardone, A. Cirri, A. Corradi, L. Foschini, R. Ianniello, et al. Crowdsensing in urban areas for city-scale mass gathering management: Geofencing and activity recognition. *IEEE Sensors Journal*, 14(12):4185–4195, December 2014.

[CCL03] I. Chlamtac, M. Conti, and J. J.-N. Liu. Mobile ad hoc networking: Imperatives and challenges. *Ad Hoc Networks*, 1(1):13–64, July 2003.

[CCP09] B. Chen, H. Cheng, and J. Palen. Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems. *Transportation Research Part C*, 17(1):1–10, February 2009.

[CDI11] N. Cenerario, T. Delot, and S. Ilarri. A content-based dissemination protocol for VANETs: Exploiting the encounter probability. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):771–782, September 2011.

[CDMP16] D. Cerotti, S. Distefano, G. Merlino, and A. Puliafito. A Crowd-Cooperative Approach for Intelligent Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1529–1539, June 2016.

[CDW98] S. Clark and H. Durrant-Whyte. Autonomous land vehicle navigation using millimeter wave radar. In *International Conference on Robotics and Automation (ICRA)*, volume 4, pp. 3697–3702. IEEE Computer Society, May 1998.

[Cel01] M. Cellario. Human-centered intelligent vehicles: Toward multimodal interface integration. *IEEE intelligent systems*, 16(4):78–81, July 2001.

[CFZ+14] Z. Chen, R. Fu, Z. Zhao, Z. Liu, L. Xia, et al. gMission: A general spatial crowdsourcing platform. *VLDB Endowment*, 7(13):1629–1632, August 2014.

[CG14] M. Conti and S. Giordano. Mobile Ad Hoc Networking: Milestones, Challenges, and New Research Directions. *IEEE Communications Magazine*, 52(1):85–96, January 2014.

[CGM06] M. Caliskan, D. Graupner, and M. Mauve. Decentralized Discovery of Free Parking Places. In *Third International Workshop on Vehicular Ad Hoc Networks (VANET)*, pp. 30–39. ACM, September 2006.

[Cis17] Cisco Inc. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021 White Paper. `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html`, March 2017. [Accessed May 14, 2018].

[CLC+15] P. Cheng, X. Lian, Z. Chen, R. Fu, L. Chen, et al. Reliable diversity-based spatial crowdsourcing by moving workers. *VLDB Endowment*, 8(10):1022–1033, June 2015.

[CLF01] R. S. Cost, Y. Labrou, and T. Finin. Coordinating agents using agent communication languages conversations. In *Coordination of Internet agents*, pp. 183–196. Springer, 2001.

[CLZ00] G. Cabri, L. Leonardi, and F. Zambonelli. Weak and strong mobility in mobile agent applications. In *Second International Conference and Exhibition on The Practical Application of Java (PA JAVA)*. April 2000.

[CLZ+14] N. Cheng, N. Lu, N. Zhang, X. S. Shen, and J. W. Mark. Vehicular WiFi offloading: Challenges and solutions. *Vehicular Communications*, 1(1):13–21, January 2014.

[CM99] S. Corson and J. Macker. Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. RFC 2501, RFC Editor, January 1999.

[CMGSS13] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. Device-to-device communications with Wi-Fi Direct: Overview and experimentation. *IEEE Wireless Communications*, 20(3):96–104, June 2013.

[CMSM+18] D. Calabuig, D. Martn-Sacristn, J. F. Monserrat, M. Botsov, and D. Gozlvez. Distribution of Road Hazard Warning Messages to Distant Vehicles in Intelligent Transport Systems. *IEEE Transactions on Intelligent Transportation Systems*, 19(4):1152–1165, April 2018.

[CN11] C. Chowdhury and S. Neogy. Reliability Estimation of Delay Tolerant QoS Mobile Agent System in MANET. In N. Chaki and A. Cortesi, editors, *10th International Conference on Computer Information Systems Analysis and Technologies (CISIM)*, pp. 38–47. Springer, December 2011.

[Coo17] G. Cookson. Smart Parking – Silver Bullet for Parking Pain. `http://www2.inrix.com/research-parking-2017`, July 2017. [Accessed May 14, 2018].

[CV15] M. B. Channappagoudar and P. Venkataram. Performance analysis of a node monitoring protocol for mobile ad-hoc networks: An agent-based approach. *Internationl Journal of Communication Networks and Distributed Systems*, 14(3):303–338, April 2015.

[CYS15] X. Cheng, L. Yang, and X. Shen. D2D for Intelligent Transportation Systems: A feasibility study. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1784–1793, January 2015.

[DB15] T. Darwish and K. A. Bakar. Traffic density estimation in vehicular ad hoc networks: A review. *Ad Hoc Networks*, 24:337–351, January 2015.

[DCI10] T. Delot, N. Cenerario, and S. Ilarri. Vehicular event sharing with a mobile peer-to-peer architecture. *Transportation Research Part C: Emerging Technologies*, 18(4):584–598, August 2010.

[DD10] G. Dimitrakopoulos and P. Demestichas. Intelligent Transportation Systems. *IEEE Vehicular Technology Magazine*, 5(1):77–84, March 2010.

[dFHC+11] E. P. de Freitas, T. Heimfarth, L. A. G. Costa, A. M. Ferreira, C. E. Pereira, et al. Analyzing different levels of geographic context awareness in agent ferrying over VANETs. In *ACM Symposium on Applied Computing (SAC)*, pp. 413–418. ACM, March 2011.

[DHdLD10] A. Dey, J. Hightower, E. de Lara, and N. Davies. Location-based services. *IEEE Pervasive Computing*, 9(1):11–12, January 2010.

[DI11] T. Delot and S. Ilarri. Data gathering in vehicular networks: The VESPA experience (invited paper). In *Fifth IEEE Workshop On User MObility and VEhicular Networks (LCN ON-MOVE), in conjunction with the 36th IEEE Conference on Local Computer Networks (LCN)*, pp. 801–808. IEEE Computer Society, October 2011.

[DI13] T. Delot and S. Ilarri. The VESPA Project: Driving advances in data management for vehicular networks. *ERCIM News*, (94):17–18, July 2013.

[DIdCRH14] T. Delot, S. Ilarri, and M. del Carmen Rodríguez-Hernández. Intelligent Transportation Systems – Maybe, But Where Are My Agents? In *Sixth International Conference on Ad Hoc Networks (AdHocNets)*, volume 140 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST)*, pp. 39–50. Springer, August 2014.

[DILC13] T. Delot, S. Ilarri, S. Lecomte, and N. Cenerario. Sharing with caution: Managing parking spaces in vehicular networks. *Mobile Information Systems*, 9(1):69–98, February 2013.

[DLBB+15] M. Da Lio, F. Biral, E. Bertolazzi, M. Galvani, P. Bosetti, et al. Artificial co-drivers as a universal enabling technology for future intelligent vehicles and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):244–263, February 2015.

[DMIH11] T. Delot, N. Mitton, S. Ilarri, and T. Hien. GeoVanet: A routing protocol for query processing in vehicular networks. *Mobile Information Systems*, 7(4):329–359, October 2011.

[DPH05] S. M. Das, H. Pucha, and Y. C. Hu. Performance comparison of scalable location services for geographic ad hoc routing. In *24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pp. 1228–1239. IEEE Computer Society, March 2005.

[DPS13] E. Dahlman, S. Parkvall, and J. Skold. *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013.

[DWSH17] B. Di, T. Wang, L. Song, and Z. Han. Collaborative smartphone sensing using overlapping coalition formation games. *IEEE Transactions on Mobile Computing*, 16(1):30–43, January 2017.

[E-Z] E-ZPass Interagency Group. E-ZPass Homepage. `http://e-zpassiag.com/`. [Accessed May 14, 2018].

[EAAD12] M. EL Amine Ameur and H. Drias. A Cooperative Multi-Agent System for Traffic Congestion Management in VANET. In *Advances in Computer Science, Engineering & Applications (ICCSEA)*, pp. 499–508. Springer, May 2012.

[ECVL15] C. Englund, L. Chen, A. Vinel, and S. Y. Lin. *Future applications of VANETs*, pp. 525–544. Springer, 2015.

[eMu] eMule Project. Official eMule Homepage. `http://www.emule-project.net`. [Accessed May 14, 2018].

[Eri17] Ericsson. Mobile data traffic growth outlook. `https://www.ericsson.com/en/mobility-report/reports/november-2017/mobile-data-traffic-growth-outlook`, November 2017. [Accessed May 14, 2018].

[Est] EstiNet simulator. `http://www.estinet.com/ns`. [Accessed May 14, 2018].

[Eth16] IEEE Standard for Ethernet. *IEEE Std 802.3-2015 (Revision of IEEE Standard 802.3-2012)*, pp. 1–4017, March 2016.

[Eur08] European Parliament, Council of the European Union. *Comission Decission 2008/671/EC of 5 August 2008 on the harmonised use of radio spectrum in the 5875 - 5905 MHz frequency band for safety-related applications of Intelligent Transport Systems (ITS)*. The European Commission, 2008.

[Eur10] European Parliament, Council of the European Union. *Directive 2010/40/EU of the European Parliament and of the Council of 7 July 2010 on the framework for the deployment of Intelligent Transport Systems in the field of road transport and for interfaces with other modes of transport*. The European Commission, 2010.

[Eur15] European Parliament, Council of the European Union. *Regulation (EU) 2015/758 of the European Parliament and of the Council of 29 April 2015 concerning type-approval requirements for the deployment of the eCall in-vehicle system based on the 112 service and amending Directive 2007/46/EC*. The European Commission, 2015.

[Faca] Facebook Inc. Nearby Friends. `https://www.facebook.com/help/629537553762715`. [Accessed May 14, 2018].

[Facb] Facebook Inc. React Native. `https://facebook.github.io/react-native/`. [Accessed May 14, 2018].

[FÁO+10] D. Fuentes, J. A. Álvarez, J. A. Ortega, L. González-Abril, and F. Velasco. Event-Based Method for Detecting Trojan Horses in Mobile Devices. In *Second International Conference on Security and Privacy in Mobile Information and Communication Systems (MobiSec)*, pp. 153–162. Springer, May 2010.

[FAR] FAROO Limited. FAROO Search. `http://www.faroo.com/`. [Accessed May 14, 2018].

[FBG17] A. S. Fonteles, S. Bouveret, and J. Gensel. A programming framework for spatial crowdsourcing. In *15th International Conference on Advances in Mobile Computing & Multimedia (MoMM)*, pp. 131–140. ACM, December 2017.

[FCM+14] M. D. Felice, E. Cerqueira, A. Melo, M. Gerla, F. Cuomo, et al. A distributed backbone-based framework for live video sharing in VANETs. In *11th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks (PE-WASUN)*, pp. 33–40. ACM, September 2014.

[FG96] S. Franklin and A. Graesser. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *International Workshop on Agent Theories, Architectures, and Languages*, pp. 21–35. Springer, August 1996.

[FGM+14] M. Fogue, P. Garrido, F. J. Martinez, J.-C. Cano, C. T. Calafate, et al. A system for automatic notification and severity estimation of automotive accidents. *IEEE Transactions on Mobile Computing*, 13(5):948–963, May 2014.

[FIP] FIPA. The Foundation of Intelligent Physical Agents. `http://www.fipa.org/`. [Accessed May 14, 2018].

[FK15] D. J. Fagnant and K. Kockelman. Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, July 2015.

[Fle13] B. Fleming. Sensors – A forecast [automotive electronics]. *IEEE Vehicular Technology Magazine*, 8(3):4–12, August 2013.

[Fli09] B. Fling. *Mobile design and development: Practical concepts and techniques for creating mobile sites and Web apps*, chapter 1. O'Reilly Media, 2009.

[FRSS13] P. Fazio, F. D. Rango, C. Sottile, and A. F. Santamaria. Routing Optimization in Vehicular Networks: A New Approach Based on Multiobjective Metrics and Minimum Spanning Tree. *International Journal of Distributed Sensor Networks*, pp. 1–13, November 2013.

[FSAA01] H. Ferhatosmanoglu, I. Stanoi, D. Agrawal, and A. E. Abbadi. Constrained Nearest Neighbor Queries. In *Seventh Internationl Symposium on Advances in Spatial and Temporal Databases (SSTD)*, volume 2121 of *Lecture Notes in Computer Science (LNCS)*, pp. 257–276. Springer, July 2001.

[FYN04] K. Fujinami, T. Yamabe, and T. Nakajima. Take me with you!: A case study of context-aware application integrating cyber and physical spaces. In *ACM symposium on Applied computing*, pp. 1607–1614. ACM, March 2004.

[GAK15] M. Gao, G. Ayers, and C. Kozyrakis. Practical near-data processing for in-memory analytics frameworks. In *24th International Conference on Parallel Architecture and Compilation (PACT)*, pp. 113–124. IEEE Computer Society, October 2015.

[Gam18] Game Connect LTD. Play2Live is the first full-blown decentralized streaming platform for gamers and esports fans. `https://play2live.io/`, 2018. [Accessed May 14, 2018].

[Gar18] Gartner Press Release. Gartner Says Worldwide Sales of Smartphones Recorded First Ever Decline During the Fourth Quarter of 2017. `https://www.gartner.com/newsroom/id/3859963`, February 2018. [Accessed May 14, 2018].

[GBD15] T. Gindele, S. Brechtel, and R. Dillmann. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine*, 7(1):69–79, January 2015.

[GIZCC15] J. A. Guerrero-Ibanez, S. Zeadally, and J. Contreras-Castillo. Integration challenges of Intelligent Transportation Systems with connected vehicle, cloud computing, and Internet of Things technologies. *IEEE Wireless Communications*, 22(6):122–128, December 2015.

[GK94] M. Genesereth and S. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, 1994.

[GL04] B. Gedik and L. Liu. MobiEyes: Distributed Processing of Continuously Moving Queries on Moving Objects in a Mobile System. In *Ninth International Conference on Extending Database Technology (EDBT)*, volume 2992 of *Lecture Notes in Computer Science (LNCS)*, pp. 67–87. Springer, March 2004.

[GLL16] N. K. Giang, V. C. Leung, and R. Lea. On Developing Smart Transportation Applications in Fog Computing Paradigm. In *Sixth ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications (DIVANet)*, pp. 91–98. ACM, November 2016.

[GLW+15] H. Gao, C. H. Liu, W. Wang, J. Zhao, Z. Song, et al. A survey of incentive mechanisms for participatory sensing. *IEEE Communications Surveys and Tutorials*, 17(2):918–943, January 2015.

[Goo] Google Inc. Google Assistant. `https://assistant.google.com`. [Accessed May 14, 2018].

[GPZA13] A. Grazioli, M. Picone, F. Zanichelli, and M. Amoretti. Collaborative Mobile Application and Advanced Services for Smart Parking. In *14th International Conference on Mobile Data Management (MDM)*, volume 2, pp. 39–44. IEEE Computer Society, June 2013.

[GSB02] M. Gunes, U. Sorges, and I. Bouazizi. ARA-the ant-colony based routing algorithm for MANETs. In *International Conference on Parallel Processing Workshop (ICPP)*, pp. 79–85. IEEE Computer Society, December 2002.

[GWY+15] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, et al. Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm. *ACM Computing Surveys*, 48(1):7:1–7:31, August 2015.

[GYL11] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, November 2011.

[Hal10] A. Halsey III. D.C. tests new parking technology to help drivers find space, pay more easily. *The Washington Post*, June 29, 2010.

[Has09] C. Haseman. *Android Essentials*. Apress, 2009.

[HAY+05] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell. A survey of peer-to-peer storage techniques for distributed file systems. In *International Conference on Information Technology (ITCC)*, volume 2, pp. 205–213. IEEE Computer Society, April 2005.

[HBZ+06] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, et al. Cartel: A distributed mobile sensor computing system. In *International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 125–138. ACM, November 2006.

[HFB09] J. Harri, F. Filali, and C. Bonnet. Mobility models for vehicular ad hoc networks: A survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 11(4), December 2009.

[HH15] E. Hossain and M. Hasan. 5G cellular: Key enabling technologies and research challenges. *IEEE Instrumentation Measurement Magazine*, 18(3):11–21, May 2015.

[HKB99] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pp. 174–185. ACM, August 1999.

[HL08] H. Hartenstein and K. P. Laberteaux. A Tutorial Survey on Vehicular Ad Hoc Networks. *IEEE Communications Magazine*, 46(6):164–171, June 2008.

[HMMY16] S. P. Holland, E. T. Mansur, N. Z. Muller, and A. J. Yates. Are There Environmental Benefits from Driving Electric Vehicles? The Importance of Local Factors. *American Economic Review*, 106(12):3700–3729, December 2016.

[HMS14] A. M. Haidar, K. M. Muttaqi, and D. Sutanto. Technical challenges for electric power industries due to grid-integrated electric vehicles in low voltage distributions: A review. *Energy Conversion and Management*, 86:689–700, October 2014.

[HN06] X. Han and L. P. Naeher. A review of traffic-related air pollution exposure assessment studies in the developing world. *Environment International*, 32(1):106–120, January 2006.

[HPY+14] J. Harding, G. Powell, R. Yoon, J. Fikentscher, C. Doyle, et al. Vehicle-to-vehicle communications: Readiness of V2V technology for application. Technical report, National Highway Traffic Safety Administration, August 2014.

[HSY17] D. Hermelin, M. Segal, and H. Yedidsion. Coordination of Mobile Mules via Facility Location Strategies. In *15th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, pp. 107–119. Springer, June 2017.

[HTC] HTCondor - High Throughput Computing. `https://research.cs.wisc.edu/htcondor/`. [Accessed May 14, 2018].

[Hu15] X. Hu. *A platform for building context-aware mobile crowdsensing applications in vehicular social networks*. Ph.D. thesis, University of British Columbia, 2015.

[Hua11] D. Huang. The Future Use of Mobile Devices Served Psychology Applications. In Y. Zhang, editor, *International Conference on Future Wireless Networks and Information Systems (ICFWI)*, pp. 115–122. Springer, November 2011.

[HYY+15] C.-Y. Hsu, C.-S. Yang, L.-C. Yu, C.-F. Lin, H.-H. Yao, et al. Development of a cloud-based service framework for energy conservation in a sustainable intelligent transportation system. *International Journal of Production Economics*, 164:454–461, June 2015.

[HZS+15] X. Hu, J. Zhao, B.-C. Seet, V. Leung, T. Chu, et al. S-Aframe: Agent-based multilayer framework with context-aware semantic service for vehicular social networks. *IEEE Transactions on Emerging Topics in Computing*, 3(1):44–63, March 2015.

[I2P] I2P Project. I2P Anonymous network. `https://geti2p.net`. [Accessed May 14, 2018].

[IBHD16] M. D. Ingle, P. Bhor, G. Hargude, and S. Deshpande. Peer-to-peer video, audio streaming over wi-fi network on mobile device. *International Research Journal of Engineering and Technology (IRJET)*, 3(5):3279–2383, May 2016.

[IDTL15] S. Ilarri, T. Delot, and R. Trillo-Lado. A data management perspective on vehicular networks. *IEEE Communications Surveys and Tutorials*, 17(4):2420–2460, August 2015.

[IEE10] IEEE Computer Society. IEEE Standard for Information Technology – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments, July 2010.

[IEE16] IEEE Computer Society. IEEE Standard for Information Technology – Telecommunications and Information Exchange between Systems; Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, December 2016.

[IFT] IFTTT. IFTTT helps your apps and devices work together. `https://ifttt.com/`. Accessed May 14, 2018.

[IHTLdCRH15] S. Ilarri, R. Hermoso, R. Trillo-Lado, and M. del Carmen Rodríguez-Hernández. A Review of the Role of Sensors in Mobile Context-Aware Recommendation Systems. *International Journal of Distributed Sensor Networks, ISSN 1550-1329*, 2015:1–30, November 2015.

[IMI06] S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Transactions on Mobile Computing*, 5(8):1029–1043, August 2006.

[IMI08] S. Ilarri, E. Mena, and A. Illarramendi. Using cooperative mobile agents to monitor distributed and dynamic environments. *Information Sciences*, 178(9):2105–2127, May 2008.

[IMI10] S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Computing Surveys*, 42(3):1–73, March 2010.

[IMR13] S. Ilarri, E. Mena, and V. Rújula. Vanet-X: A videogame to evaluate information management in vehicular networks. 1075:28–35, January 2013.

[IMS+11] S. Ilarri, E. Mena, A. Sheth, et al. Semantics in location-based services. *IEEE Internet Computing*, 15(6):10, November 2011.

[IRTL18] S. Ilarri, P. Roig, and R. Trillo-Lado. GeoSPRINGS: Towards a Location-Aware Mobile Agent Platform. In *16th International Symposium on Web and Wireless Geographical Information Systems (W2GIS)*, volume 10819 of *Lecture Notes in Computer Science (LNCS)*, pp. 51–60. Springer, May 2018.

[ITLM06] S. Ilarri, R. Trillo-Lado, and E. Mena. SPRINGS: A Scalable Platform for Highly Mobile Agents in Distributed Computing Environments. In *Fourth International WoWMoM Workshop on Mobile Distributed Computing (MDC)*, pp. 633–637. IEEE Computer Society, June 2006.

[IWD14] S. Ilarri, O. Wolfson, and T. Delot. Collaborative sensing for urban transportation. *IEEE Data Engineering Bulletin*, 37(4):3–14, December 2014. Special Issue on Urban Informatics.

[Jen01] N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, April 2001.

[JLW+16] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen. A survey on platoon-based vehicular cyber-physical systems. *IEEE communications surveys & tutorials*, 18(1):263–284, March 2016.

[JT03] N. R. John and D. C. Tucker. 10 Myths About PDAs - Debunked! *Computers in Libraries*, 23(3):26–30, March 2003.

[JTM+06] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 GHz DSRC-based vehicular safety communication. *IEEE Wireless Communications*, 13(5), November 2006.

[JVLR15] L. G. Jaimes, I. J. Vergara-Laurens, and A. Raij. A survey of incentive techniques for mobile crowd sensing. *IEEE Internet of Things Journal*, 2(5):370–380, October 2015.

[KAA+11] F. Kelly, B. Armstrong, R. Atkinson, H. R. Anderson, B. Barratt, et al. The London low emission zone baseline study. *Research Report Health Effects Institute*, (163):3–79, November 2011.

[KAE+11] G. Karagiannis, O. Altintas, E. Ekici, G. J. Heijenk, B. Jarupan, et al. Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions. *IEEE Communications Surveys & Tutorials*, 13(4):584–616, July 2011.

[KcSH17] A. O. Kotb, Y. chun Shen, and Y. Huang. Smart Parking Guidance, Monitoring and Reservations: A Review. *IEEE Intelligent Transportation Systems Magazine*, 9(2):6–16, April 2017.

[KD10] R. Kumar and D. M. Dave. Mobile Agent as an Approach to Improve QoS in Vehicular Ad Hoc Network. *International Journal of Computer Applications Special Issue on "Mobile Ad-hoc Networks"*, pp. 67–72, August 2010.

[Ker09] B. S. Kerner. *Introduction to modern traffic flow theory and control.* Springer, 2009.

[KGM14] B. W. Kolosz and S. M. Grant-Muller. Appraisal and evaluation of interurban ITS: A European survey. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1070–1087, September 2014.

[Kha15] A. Khalifa. *Collaborative Computing Cloud: Architecture and Management Platform.* Ph.D. thesis, Virginia Tech, 2015.

[KLW14] A. Klappenecker, H. Lee, and J. L. Welch. Finding available parking spaces made easy. *Ad Hoc Networks*, 12:243–249, January 2014.

[KM12] M. Kakkasageri and S. Manvi. Multiagent driven dynamic clustering of vehicles in VANETs. *Journal of Network and Computer Applications*, 35(6):1771–1780, November 2012.

[KM14] M. Kakkasageri and S. Manvi. Regression based critical information aggregation and dissemination in VANETs: A cognitive agent approach. *Vehicular Communications*, 1(4):168–180, October 2014.

[KN12] S. King and S. Nadal. PPCoin: Peer-to-peer crypto-currency with proof-of-stake. `https://peercoin.net/assets/paper/peercoin-paper.pdf`, August 2012. [Accessed May 14, 2018].

[KOB+08] T. Kindberg, E. O'Neill, C. Bevan, V. Kostakos, D. Stanton Fraser, et al. Measuring trust in Wi-Fi hotspots. In *26th International Conference on Human Factors in Computing Systems (CHI)*, pp. 173–182. ACM, April 2008.

[Kor02] G. Kortuem. Proem: A middleware platform for mobile peer-to-peer computing. *ACM SIGMOBILE Mobile Computing and Communications Review*, 6(4):62–64, October 2002.

[KP11] V. Khairnar and S. Pradhan. Mobility models for Vehicular Ad-hoc Network simulation. In *IEEE Symposium on Computers Informatics (ISCI)*, pp. 460–465. IEEE Computer Society, March 2011.

[KSN00] M. T. Kone, A. Shimazu, and T. Nakajima. The state of the art in agent communication languages. *Knowledge and Information Systems*, 2(3):259–284, August 2000.

[Küp05] A. Küpper. *Location-Based Services.* John Willey & Sons, 2005.

[LAB+11] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, et al. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV)*, pp. 163–168. IEEE Computer Society, June 2011.

[LCX+17] X. Liu, X. Chen, X. Xu, E. Mai, H. Y. Noh, et al. Delay Effect in Mobile Sensing System for Urban Air Pollution Monitoring. In *15th ACM Conference on Embedded Network Sensor Systems (SenSys)*, pp. 73:1–73:2. ACM, November 2017.

[LDK+15] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, April 2015.

[Lep18] T. Leppänen. *Resource-Oriented Mobile Agent and Software Framework for the Internet of Things*. Ph.D. thesis, University of Oulu, March 2018.

[LFP99] Y. Labrou, T. Finin, and Y. Peng. Agent communication languages: The current landscape. *IEEE Intelligent Systems and their Applications*, 14(2):45–52, March 1999.

[LG10] U. Lee and M. Gerla. A survey of urban vehicular sensing platforms. *Computer Networks*, 54(4):527–544, March 2010.

[Lit17] T. Litman. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute, 2017.

[Lla] Llamalab. Automate, make your phone or tablet smarter with automation. `http://llamalab.com/automate/`. [Accessed May 14, 2018].

[LLC13] Z. Li, C. Liu, and C. Chigan. On secure VANET-based ad dissemination with pragmatic cost and effect control. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):124–135, March 2013.

[LLPG10a] U. Lee, J. Lee, J.-S. Park, and M. Gerla. FleaNet: A virtual market place on vehicular networks. *IEEE Transactions on Vehicular Technology*, 59(1):344–355, January 2010.

[LLPG10b] U. Lee, J. Lee, J.-S. Park, and M. Gerla. Fleanet: A virtual market place on vehicular networks. *IEEE Transactions on Vehicular Technology*, 59(1):344–355, January 2010.

[LMMSC14] I. M. Lombera, L. E. Moser, P. M. Melliar-Smith, and Y.-T. Chuang. Peer-to-peer publication, search and retrieval using the Android mobile platform. *Computer networks*, 65:56–72, June 2014.

[LMZ+06] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, et al. MobEyes: Smart mobs for urban monitoring with a vehicular sensor network. *IEEE Wireless Communications*, 13(5):52–57, November 2006.

[LO99] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.

[LT03] C. Lindemann and A. Thümmler. Performance analysis of the General Packet Radio Service. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 41(1):1–17, January 2003.

[LW07] F. Li and Y. Wang. Routing in Vehicular Ad Hoc Networks: A Survey. *IEEE Vehicular Technology Magazine*, 2(2):12–22, June 2007.

[LWZM15] L. Liu, W. Wei, D. Zhao, and H. Ma. Urban resolution: New metric for measuring the quality of urban sensing. *IEEE Transactions on Mobile Computing*, 14(12):2560–2575, December 2015.

[MCCF14] F. Malandrino, C. Casetti, C.-F. Chiasserini, and M. Fiore. Content download in vehicular networks in presence of noisy mobility prediction. *IEEE Transactions on Mobile Computing*, 13(5):1007–1021, May 2014.

[MCN14] G. Mitra, C. Chowdhury, and S. Neogy. Application of mobile agent in VANET for measuring environmental data. In *Applications and Innovations in Mobile Computing (AIMoC)*, pp. 48–53. February 2014.

[MDLW18] M. Mladenović, T. Delot, G. Laporte, and C. Wilbaut. The parking allocation problem for connected vehicles. *Journal of Heuristics*, pp. 1–23, January 2018.

[MDNB14] F. Mezghani, R. Dhaou, M. Nogueira, and A.-L. Beylot. Content dissemination in vehicular social networks: Taxonomy and user satisfaction. *IEEE Communications Magazine*, 52(12):34–40, December 2014.

[MDS+14] D. Micheli, A. Delfini, F. Santoni, F. Volpini, and M. Marchetti. Measurement of electromagnetic field attenuation by building walls in the mobile phone and satellite navigation frequency bands. *IEEE Antennas and Wireless Propagation Letters*, 14:698–702, December 2014.

[MDW99] D. Milojicic, F. Douglis, and R. Wheeler. *Mobility: Processes, computers, and agents*. ACM Press/Addison-Wesley Publishing Co., 1999.

[ME06] P. Misra and P. Enge. *Global Positioning System: Signals, measurements and performance*. Ganga-Jamuna Press, 2006.

[MFT+13] F. J. Martinez, M. Fogue, C. K. Toh, J.-C. Cano, C. T. Calafate, et al. Computer simulations of VANETs using realistic city topologies. *Wireless Personal Communications*, 69(2):639–663, March 2013.

[Mic] Microsoft Inc. Windows Official Site. `https://windows.microsoft.com`. [Accessed on May 14, 2018].

[Mor02] R. Morrow. *Bluetooth: Operation and Use*. McGraw-Hill, Inc., 2002.

[MS14] P. K. Mishra and R. Singh. A survey on reliability estimation techniques for mobile agent based systems. *International Journal of Advanced Computer Research*, 4(1):123, March 2014.

[MSN05] M. Motani, V. Srinivasan, and P. S. Nuggehalli. PeopleNet: Engineering a wireless virtual social network. In *11th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 243–257. ACM, September 2005.

[Mur89] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[MV11] S. Mizzaro and L. Vassena. A social approach to context-aware retrieval. *World Wide Web*, 14(4):377–405, July 2011.

[Nak08] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `http://bitcoin.org/bitcoin.pdf`, 2008. [Accessed May 14, 2018].

[NCN11] R. Neogy, C. Chowdhury, and S. Neogy. Reliability Estimation of Mobile Agents for Service Discovery in MANET. In *First International Conference on Parallel, Distributed Computing Technologies and Applications (PDCTA)*, pp. 148–157. Springer, September 2011.

[NCN12] R. Neogy, C. Chowdhury, and S. Neogy. A reliable service discovery protocol using mobile agents in MANET. In *Annual Reliability and Maintainability Symposium*, pp. 1–7. IEEE Computer Society, January 2012.

[NCT] NCTUns 6.0 Network Simulator and Emulator. `http://nsl.cs.nctu.edu.tw/NSL/nctuns.html`. [Accessed May 14, 2018].

[NLS+15] J. Nurmi, E. S. Lohan, S. Sand, H. Hurskainen, et al. *GALILEO positioning technology*, volume 176. Springer, 2015.

[Ns3] Network simulator ns-3. `https://www.nsnam.org/`. [Accessed May 14, 2018].

[Nwa96] H. S. Nwana. Software agents: An overview. *The Knowledge Engineering Review*, 11(3):205–244, September 1996.

[OHI05] I. Oppermann, M. Hämäläinen, and J. Iinatti. *UWB: Theory and applications*. John Wiley & Sons, 2005.

[OHY13] S. Olariu, T. Hristov, and G. Yan. *Mobile Ad Hoc Networking: Cutting Edge Directions, Second Edition*, chapter 19 "The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds", pp. 645–700. John Wiley & Sons, 2013.

[ON98] P. D. O'Brien and R. C. Nicol. FIPA - Towards a standard for software agents. *BT Technology Journal*, 16(3):51–59, July 1998.

[Ope] OpenStreetMap Foundation (OSMF). OpenStreetMap. `http://openstreetmap.org`. [Accessed May 14, 2018].

[OR03] F. Ohrtman and K. Roeder. *Wi-Fi handbook: Building 802.11b wireless networks*, volume 67. McGraw-Hill, 2003.

[Ora01] A. Oram. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[Ova14] S. Ovadia. Automate the internet with If This Then That(IFTTT). *Behavioral & Social Sciences Librarian*, 33(4):208–211, November 2014.

[OW09] S. Olariu and M. C. Weigle. *Vehicular Networks: From Theory to Practice*. Chapman & Hall/CRC, first edition, 2009.

[PBV05] B. Pourebrahimi, K. Bertels, and S. Vassiliadis. A survey of peer-to-peer networks. In *16th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC)*. Technology Foundation, November 2005.

[Pee] Peer5 Inc. Peer5, the Serverless CDN for Video Live Streaming. `https://www.peer5.com/`. [Accessed May 14, 2018].

[PPM17] F. Perez-Prada and A. Monzon. Ex-post environmental and traffic assessment of a speed reduction strategy in Madrid's inner ring-road. *Journal of Transport Geography*, 58:256–268, January 2017.

[PRF10] C. E. Palazzi, M. Roccetti, and S. Ferretti. An intervehicular communication architecture for safety and entertainment. *IEEE Transactions on Intelligent Transportation Systems*, 11(1):90–99, March 2010.

[PS96] B. Parkinson and J. Spilker. *Global Positioning System: Theory and Applications*. American Institute of Aeronautics & Astronautics, 1996.

[PUV95] M. Poloni, G. Ulivi, and M. Vendittelli. Fuzzy logic and autonomous vehicles: Experiments in ultrasonic vision. *Fuzzy Sets and Systems*, 69(1):15–27, January 1995.

[QCJ+18] H. Qiu, J. Chen, S. Jain, Y. Jiang, M. McCartney, et al. Towards robust vehicular context sensing. *IEEE Transactions on Vehicular Technology*, 67(3):1909–1922, March 2018.

[Qua] QualNet Network Simulator Software. `https://web.scalable-networks.com/qualnet-network-simulator-software`. [Accessed May 14, 2018].

[RA11] P. Ranjan and K. K. Ahirwar. Comparative Study of VANET and MANET Routing Protocols. In *International Conference on Advanced Computing and Communication Technologies (ACCT)*, pp. 517–523. RG Education Society, January 2011.

[RD00] P.-M. Ricordel and Y. Demazeau. From analysis to deployment: A multi-agent platform survey. In *International Workshop on Engineering Societies in the Agents World (ESAW)*, pp. 93–105. Springer, August 2000.

[Rec] Recursion Software Inc. Voyager: Middleware for Enterprise Mobile Applications. `http://www.recursionsw.com/voyager-intro/`. [Accessed on May 9, 2018].

[Red] Redpine Signals. 802.11p V2X Connectivity. `http://www.redpinesignals.com/Products/802.11p_V2X_Connectivity/`. [Accessed May 14, 2018].

[Rev12] S. Revnivykh. GLONASS status and modernization. *International GNSS Committee IGC-7*, pp. 4–9, November 2012.

[RN15] P. R. and R. N. An improved multipath MANET routing using link estimation and swarm intelligence. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):173, June 2015.

[RNBI07] O. Riva, T. Nadeem, C. Borcea, and L. Iftode. Context-aware migratory services in ad hoc networks. *IEEE Transactions on Mobile Computing*, 6(12):1313–1328, December 2007.

[Rom12] B. Romito. *Adaptive and decentralized storage: Data autonomy and mobility in peer-to-peer networks.* PhD thesis, Université de Caen, December 2012.

[RP01] V. Roth and J. Peters. A scalable and secure global tracking service for mobile agents. In *International Conference on Mobile Agents*, pp. 169–181. Springer, December 2001.

[Sad12] E. Sadun. *The Core iOS 6 Developer's Cookbook.* Pearson Education, 2012.

[SAQM17] N. Shah, S. A. Abid, D. Qian, and W. Mehmood. A survey of P2P content sharing in MANETs. *Computers & Electrical Engineering*, 57:55–68, January 2017.

[Sat17] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, January 2017.

[SB11] M. Seredynski and P. Bouvry. A survey of vehicular-based cooperative traffic information systems. In *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 163–168. IEEE Computer Society, October 2011.

[SCGa+15] P. M. Santos, T. Calcada, D. Guimarães, T. Condeixa, S. Sargento, et al. Demo: Platform for Collecting Data From Urban Sensors Using Vehicular Networking. In *21st Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 167–169. ACM, 2015.

[SDKM14] M. Sathiamoorthy, A. G. Dimakis, B. Krishnamachari, and F. B. Member. Distributed storage codes reduce latency in vehicular networks. *IEEE Transactions on Mobile Computing*, 13(9):2016–2027, September 2014.

[SEGD11] C. Sommer, D. Eckhoff, R. German, and F. Dressler. A computationally inexpensive empirical model of IEEE 802.11p radio shadowing in urban environments. In *Eighth International Conference on Wireless On-Demand Network Systems and Services (WONS)*, pp. 84–90. IEEE Computer Society, January 2011.

[SJLF17] J. Shi, L. Jin, J. Li, and Z. Fang. A smart parking system based on NB-IoT and third-party payment platform. In *17th International Symposium on Communications and Information Technologies (ISCIT)*, pp. 1–5. IEEE Computer Society, September 2017.

[SK14] M. Sood and S. Kanwar. Clustering in MANET and VANET: A survey. In *International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pp. 375–380. IEEE Computer Society, April 2014.

[SKK12] M. Sharifi, S. Kafaie, and O. Kashefi. A survey and taxonomy of cyber foraging of mobile devices. *IEEE Communications Surveys & Tutorials*, 14(4):1232–1243, November 2012.

[SKRM11] A. Singh, M. Kumar, R. Rishi, and D. Madan. A relative study of MANET and VANET: Its applications, broadcasting approaches and challenging issues. In *International Conference on Computer Science and Information Technology (CCIS)*, pp. 627–632. Springer, January 2011.

[SL13] K. Smarsly and K. H. Law. A migration-based approach towards resource-efficient wireless structural health monitoring. *Advanced Engineering Informatics*, 27(4):625–635, October 2013.

[SLU] Slurm workload manager. `https://slurm.schedmd.com/`. [Accessed May 14, 2018].

[SP16] A. Sładkowski and W. Pamuła. *Intelligent Transportation Systems. Problems and perspectives*, volume 303 of *Studies in Systems, Decision and Control*. Springer, 2016.

[SQM17] F. Shi, Z. Qin, and J. A. McCann. OPPay: Design and Implementation of a Payment System for Opportunistic Data Services. In *37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1618–1628. IEEE Computer Society, June 2017.

[SSPE04] C. Spyrou, G. Samaras, E. Pitoura, and P. Evripidou. Mobile agents for wireless computing: The convergence of wireless computational models with mobile-agent technologies. *Mobile Networks and Applications*, 9(5):517–528, October 2004.

[SSTK13] P. Sangulagi, M. Sarsamba, M. Talwar, and V. Katgi. Recognition and Elimination of Malicious Nodes in Vehicular Ad hoc Networks (VANETs). *Indian Journal of Computer Science and Engineering*, 4(1), February 2013.

[Sta17a] Statista. Number of mobile phone users worldwide from 2013 to 2019. `https://www.statista.com/statistics/274774`, 2017. [Accessed May 14, 2018].

[Sta17b] Statista. Number of smartphone users worldwide from 2014 to 2020. `https://www.statista.com/statistics/330695`, 2017. [Accessed May 14, 2018].

[Sta18] Statista. Number of apps available in leading app stores as of March 2017. https://www.statista.com/statistics/276623, 2018. [Accessed May 14, 2018].

[STYW04] H. Sawant, J. Tan, Q. Yang, and Q. Wang. Using Bluetooth and sensor networks for Intelligent Transportation Systems. In *Seventh Conference on International Intelligent Transportation Systems (ITS)*, pp. 767–772. IEEE Computer Society, October 2004.

[STZ12] X. Sheng, J. Tang, and W. Zhang. Energy-Efficient Collaborative Sensing with Mobile Phones. In *31st Annual IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1916–1924. March 2012.

[SUM] SUMO - Simulation of Urban MObility. http://sumo.dlr.de. [Accessed May 14, 2018].

[SvEK+10] R. S. Schwartz, M. van Eenennaam, G. Karagiannis, G. J. Heijenk, W. K. Wolterink, et al. Using V2V communication to create over-the-horizon awareness in multiple-lane highway scenarios. In *Intelligent Vehicles Symposium (IVS)*, pp. 998–1005. IEEE Computer Society, June 2010.

[SW12] A. A. Saifan and H. A. Wahsheh. Mutation Operators for JADE Mobile Agent Systems. In *Third International Conference on Information and Communication Systems (ICICS)*, pp. 16:1–16:5. ACM, May 2012.

[TC14] J. D. Townsend and R. J. Calantone. Evolution and transformation of innovation in the global automotive industry. *Journal of Product Innovation Management*, 31(1):4–7, January 2014.

[TCS17] Y. Tong, L. Chen, and C. Shahabi. Spatial crowdsourcing: Challenges, techniques, and applications. *VLDB Endowment*, 10(12):1988–1991, August 2017.

[Tel] Telecom Italia SpA. JAVA Agent DEvelopment Framework. http://jade.tilab.com/. [Accessed on May 14, 2018].

[TGS14] H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *VLDB Endowment*, 7(10):919–930, June 2014.

[Thea] The Apache Software Foundation. Apache Cordova. https://cordova.apache.org. [Accessed May 14, 2018].

[Theb] The Free Software Foundation. GNU Operating System. https://www.gnu.org/. Accessed May 14, 2018.

[The14] The European Comission. eCall: Time saved = lives saved. https://ec.europa.eu/digital-single-market/ecall-time-saved-lives-saved, June 2014. [Accessed May 14, 2018].

[TLIM07] R. Trillo-Lado, S. Ilarri, and E. Mena. Comparison and Performance Evaluation of Mobile Agent Platforms. In *Third International Conference on Autonomic and Autonomous Systems (ICAS)*, pp. 41–46. IEEE Computer Society, June 2007.

[TLZ+18] N. Ta, G. Li, T. Zhao, J. Feng, H. Ma, et al. An Efficient Ride-Sharing Framework for Maximizing Shared Route. *IEEE Transactions on Knowledge and Data Engineering*, 30(2):219–233, February 2018.

[TNCS02] Y.-C. Tseng, S.-Y. Ni, Y.-S. Chen, and J.-P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. *Wireless Networks*, 8(2/3):153–167, March 2002.

[To16] H. To. Task Assignment in Spatial Crowdsourcing: Challenges and Approaches. In *24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS), third ACM SIGSPATIAL PhD Symposium*, pp. 1:1–1:4. ACM, November 2016.

[TOR] Torque Resource Manager. `http://www.adaptivecomputing.com/products/open-source/torque/`. [Accessed May 14, 2018].

[TPS02] Y. Tao, D. Papadias, and Q. Shen. Continuous nearest neighbor search. In *28th International Conference on Very Large Data Bases (VLDB)*, pp. 287–298. VLDB Endowment, August 2002.

[TSD+16] Y. Tong, J. She, B. Ding, L. Wang, and L. Chen. Online Mobile Micro-Task Allocation in Spatial Crowdsourcing. In *32nd International Conference on Data Engineering (ICDE)*, pp. 49–60. IEEE Computer Society, May 2016.

[TSK15] H. To, C. Shahabi, and L. Kazemi. A server-assigned spatial crowdsourcing framework. *ACM Transactions on Spatial Algorithms and Systems*, 1(1):2:1–2:28, August 2015.

[TW16] J. Timpner and L. Wolf. Query-response geocast for vehicular crowd sensing. *Ad Hoc Networks*, 36:435–449, January 2016.

[TWW14] J. Timpner, M. Wozenilek, and L. Wolf. Breadcrumb Routing: Query-Response Geocast for Mobile Originators in Vehicular Networks. In *Fifth IEEE Vehicular Networking Conference (VNC)*, pp. 45–52. IEEE Computer Society, December 2014.

[TZQS09] M. Tubaishat, P. Zhuang, Q. Qi, and Y. Shang. Wireless sensor networks in Intelligent Transportation Systems. *Wireless Communications and Mobile Computing*, 9(3):287–302, March 2009.

[uHC16] U. ul Hassan and E. Curry. Efficient task assignment for spatial crowdsourcing: A combinatorial fractional optimization approach with semi-bandit learning. *Expert Systems with Applications*, 58:36–56, October 2016.

[UI] O. Urra and S. Ilarri. Spatial crowdsourcing with mobile agents in vehicular networks. 37 pages according to the original authors' submission file. Manuscript submitted for publication, 2018.

[UI13] O. Urra and S. Ilarri. Using Mobile Agents in Vehicular Networks for Data Processing. In *14th International Conference on Mobile Data Management (MDM), PhD Forum*, volume 2, pp. 11–14. IEEE Computer Society, June 2013.

[UI14] O. Urra and S. Ilarri. Distributed data processing in vehicular networks using mobile agents. *Actas de las III Jornadas de Jóvenes Investigadores del I3A, ISSN 2341-4790*, 2:61–62, June 2014.

[UI16a] O. Urra and S. Ilarri. MAVSIM: Testing VANET Applications Based on Mobile Agents. In *Cognitive Vehicular Networks*, pp. 199–224. CRC Taylor and Francis Group, 2016.

[UI16b] O. Urra and S. Ilarri. Towards Spatial Crowdsourcing in Vehicular Networks Using Mobile Agents. In *20th East-European Conference on Advances in Databases and Information Systems (ADBIS)*, volume 637 of *Communications in Computer and Information Science (CCIS), New Trends in Databases and Information Systems*, pp. 88–95. Springer, August 2016.

[UI17a] O. Urra and S. Ilarri. Dear Mobile Agent, Could You Please Find Me a Parking Space? In *21st East-European Conference on Advances in Databases and Information Systems (ADBIS)*, volume 767 of *Communications in Computer and Information Science (CCIS), New Trends in Databases and Information Systems*, pp. 82–90. Springer, September 2017.

[UI17b] O. Urra and S. Ilarri. Modeling Mobile Agents in Vehicular Networks. In *International Workshop on Modeling and Software Engineering in Business and Industry (MoSeBin) at the International Workshop on Petri Nets and Software Engineering (PNSE), co-located with the 38th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets (PN) and the 17th International Conference on Application of Concurrency to System Design (ACSD)*, volume 1846, pp. 233–238. CEUR Workshop Proceedings, June 2017.

[UIDM09] O. Urra, S. Ilarri, T. Delot, and E. Mena. Using Hitchhiker Mobile Agents for Environment Monitoring. In *Seventh International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, volume 55 of *Advances in Intelligent and Soft Computing*, pp. 557–566. Springer, March 2009.

[UIDM10] O. Urra, S. Ilarri, T. Delot, and E. Mena. Mobile Agents in Vehicular Networks: Taking a First Ride. In *Eighth International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, volume 70 of *Advances in Intelligent and Soft Computing*, pp. 118–124. Springer, April 2010.

[UIL14] O. Urra, S. Ilarri, and E. López. Simulating Mobile Agents in Vehicular Networks. In *XIX Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, pp. 71–84. September 2014. Handle: 11705/JISBD/2014/010.

[UIM08] O. Urra, S. Ilarri, and E. Mena. Testing Mobile Agent Platforms Over the Air. In *First Workshop on Data and Services Management in Mobile Environments (DS2ME) in conjunction with the 24th International Conference on Data Engineering (ICDE)*, pp. 152–159. IEEE Computer Society, April 2008.

[UIM09] O. Urra, S. Ilarri, and E. Mena. Agents Jumping in the Air: Dream or Reality? In *10th International Work-Conference on Artificial Neural Networks (IWANN), Special Session on Practical Applications of Agents and Multi-Agent Systems*, volume 5517 of *Lecture Notes in Computer Science (LNCS)*, pp. 627–634. Springer, June 2009.

[UITL17a] O. Urra, S. Ilarri, and R. Trillo-Lado. An approach driven by mobile agents for data management in vehicular networks. *Information Sciences*, 381:55–77, March 2017.

[UITL17b] O. Urra, S. Ilarri, and R. Trillo-Lado. Una Aproximación Basada en Agentes Móviles para la Gestión de Datos en Redes de Vehículos ("An Approach

Driven by Mobile Agents for Data Management in Vehicular Networks"). In *XXII Jornadas de Ingeniería del Software y Bases de Datos (JISBD)*, p. 1. September 2017. Relevant Work (artículo relevante ya publicado / Salón de la Fama - "Hall of Fame"); Handle: 11705/JISBD/2017/082.

[UITLM09] O. Urra, S. Ilarri, R. Trillo-Lado, and E. Mena. Mobile Agents and Mobile Devices: Friendship or Difficult Relationship? *Journal of Physical Agents. Special Session on Practical Applications of Agents and Multiagent Systems*, 3(2):27–37, May 2009.

[UITLM12] O. Urra, S. Ilarri, R. Trillo-Lado, and E. Mena. Mobile Software Agents for Mobile Applications. In *Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications*, pp. 725–740. IGI Global, May 2012.

[UITLM15] O. Urra, S. Ilarri, R. Trillo-Lado, and E. Mena. Mobile Agents for a Mobile World. In *Handbook of Research on Innovations in Systems and Software Engineering*, pp. 664–681. IGI Global, 2015.

[UJ16] G. A. Ubiergo and W.-L. Jin. Mobility and environment improvement of signalized networks through Vehicle-to-Infrastructure (V2I) communications. *Transportation Research Part C: Emerging Technologies*, 68:70–82, July 2016.

[USAM09] R. A. Uzcategui, A. J. D. Sucre, and G. Acosta-Marum. Wave: A tutorial. *IEEE Communications Magazine*, 47(5):126–133, May 2009.

[Vac15] J. R. Vacca. *Handbook of sensor networking: advanced technologies and applications*. CRC Press, 2015.

[Van] VanetMobiSim simulator. `http://vanet.eurecom.fr/`. [Accessed May 14, 2018].

[VCML13] A. M. Vegni, C. Campolo, A. Molinaro, and T. D. C. Little. *Routing in Opportunistic Networks*, chapter 7 "Modeling of Intermittent Connectivity in Opportunistic Networks: The Case of Vehicular Ad hoc Networks", pp. 179–207. Springer, 2013.

[VD09] F. S. T. Van Diggelen. *A-GPS: Assisted GPS, GNSS, and SBAS*. Artech House, 2009.

[VdHW08] W. Van der Hoek and M. Wooldridge. Multi-agent systems. *Foundations of Artificial Intelligence*, 3:887–928, January 2008.

[VEI] VEINS, the open source vehicular network simulation framework. `http://veins.car2x.org`. [Accessed May 14, 2018].

[VN15] A. M. Vegni and E. Natalizio. Forwarder smart selection protocol for limitation of broadcast storm problem. *Journal of Network and Computer Applications*, 47:61–71, January 2015.

[VRM14] L. M. Vaquero and L. Rodero-Merino. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, 44(5):27–32, October 2014.

[VS11] V. B. Vaghela and D. J. Shah. Vehicular Parking Space Discovery with Agent Approach. In *International Conference & Workshop on Emerging Trends in Technology (ICWET)*, pp. 613–617. ACM, February 2011.

[VVV+12] W. Vandenberghe, E. Vanhauwaert, S. Verbrugge, I. Moerman, and P. Demeester. Feasibility of expanding traffic monitoring systems with floating car data technology. *IET Intelligent Transport Systems*, 6(4):347–354, December 2012.

[WBS07] A. I. Wang, T. Bjornsgard, and K. Saxlund. Peer2Me-rapid application framework for mobile peer-to-peer applications. In *International Symposium on Collaborative Technologies and Systems(CTS)*, pp. 379–388. IEEE Computer Society, May 2007.

[WCY06] K.-L. Wu, S.-K. Chen, and P. S. Yu. Incremental processing of continual range queries over moving objects. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1560–1575, November 2006.

[WER+05] L. Wischhof, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. *InterVehicle-Communications Based on Ad Hoc Networking Principles: The FleetNet Project*, chapter 8 "Self-Organizing Traffic Information System", pp. 233–257. Universitätsverlag Karlsruhe, 2005.

[WJ95] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, June 1995.

[WJM13] Y. Wang, J. Jiang, and T. Mu. Context-aware and energy-driven route optimization for fully electric vehicles via crowdsourcing. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1331–1345, September 2013.

[WLL17] Y. Wang, H. Li, and T. Li. Participant selection for data collection through device-to-device communications in mobile sensing. *Personal and Ubiquitous Computing*, 21(1):31–41, February 2017.

[Woo09] M. Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, June 2009.

[Woo14] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151:1–32, 2014.

[WRH+12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, et al. *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.

[WTM09] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk. A survey of inter-vehicle communication protocols and their applications. *IEEE Communications Surveys and Tutorials*, 11(2):3–20, June 2009.

[WXC09] O. Wolfson, B. Xu, and H. J. Cho. Multimedia traffic information in vehicular networks. In *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, pp. 480–483. ACM, November 2009.

[WXS04] O. Wolfson, B. Xu, and P. Sistla. An Economic Model for Resource Exchange in Mobile Peer to Peer Networks. In *16th International Conference on Scientific and Statistical Database Management (SSDBM)*, pp. 235–244. IEEE Computer Society, June 2004.

[WYW+18] E. Wang, Y. Yang, J. Wu, W. Liu, and X. Wang. An efficient prediction-based user recruitment for mobile crowdsensing. *IEEE Transactions on Mobile Computing*, 17(1):16–28, January 2018.

[WZZ+16] X. Wang, X. Zheng, Q. Zhang, T. Wang, and D. Shen. Crowdsourcing in ITS: The state of the work and the networking. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1596–1605, June 2016.

[XOW04] B. Xu, A. M. Ouksel, and O. Wolfson. Opportunistic Resource Exchange in Inter-Vehicle Ad-Hoc Networks. In *Fifth International Conference on Mobile Data Management (MDM)*, pp. 4–12. IEEE Computer Society, January 2004.

[XVW09] B. Xu, F. Vafaee, and O. Wolfson. In-network query processing in mobile P2P databases. In *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS)*, pp. 207–216. ACM, November 2009.

[YL14] L.-Y. Yeh and Y.-C. Lin. A proxy-based authentication and billing scheme with incentive-aware multihop forwarding for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 15(4):1607–1621, August 2014.

[YLOJ12] J. Yu, K. H. Low, A. Oran, and P. Jaillet. Hierarchical Bayesian Nonparametric Approach to Modeling and Learning the Wisdom of Crowds of Urban Traffic Route Planning Agents. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, volume 2, pp. 478–485. December 2012.

[YMG08] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, August 2008.

[YMII14] R. Yus, E. Mena, S. Ilarri, and A. Illarramendi. SHERLOCK: Semantic Management of Location-Based Services in Wireless Environments. *Pervasive and Mobile Computing, ISSN 1574-1192*, 15(0):87–99, December 2014. Special Issue on Information Management in Mobile Applications.

[YS00] H. Yan and T. Selker. Context-aware office assistant. In *Fifth International Conference on Intelligent User Interfaces (IUI)*, pp. 276–279. January 2000.

[ZC08] J. Zhao and G. Cao. VADD: Vehicle-assisted data delivery in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 57(3):1910–1922, May 2008.

[ZH16] Y. Zhao and Q. Han. Spatial crowdsourcing: Current state and future directions. *IEEE Communications Magazine*, 54(7):102–107, July 2016.

[ZJ18] A. Zekri and W. Jia. Heterogeneous vehicular communications: A comprehensive study. *Ad Hoc Networks*, 75–76:52–79, June 2018.

[ZLJ+17] C. Zuo, K. Liang, Z. L. Jiang, J. Shao, and J. Fang. Cost-effective privacy-preserving vehicular urban sensing system. *Personal and Ubiquitous Computing*, 21(5):893–901, October 2017.

[ZMLT18] D. Zhao, H. Ma, Q. Li, and S. Tang. A Unified Delay Analysis Framework for Opportunistic Data Collection. *Wireless Networks*, 24(4):1313–1325, May 2018.

[ZWW+11] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, et al. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, July 2011.

[ZYS+16] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, et al. Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys and Tutorials*, 18(1):54–67, First quarter 2016.

[ZZC09] Y. Zhang, J. Zhao, and G. Cao. Roadcast: A Popularity Aware Content Sharing Scheme in VANETs. In *29th International Conference on Distributed Computing Systems (ICDCS)*, pp. 223–230. IEEE Computer Society, June 2009.