This is a preprint version of the following article, published by Elsevier:
Oscar Urra, Sergio Ilarri, Raquel Trillo-Lado, "*An approach driven by mobile agents for data management in vehicular networks*", Information Sciences, ISSN 0020-0255, Volume 381, Pages 55-77, Elsevier, March 2017. (DOI: 10.1016/j.ins.2016.11.007).

An Approach Driven by Mobile Agents for Data Management in Vehicular Networks

Oscar Urra[a], Sergio Ilarri[a,b], Raquel Trillo-Lado[a,b]

[a] *Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza (Spain)*
[b] *I3A, University of Zaragoza, Zaragoza (Spain)*

Abstract

In the last years, and thanks to the improvements on computing and communications technologies, wireless networks formed by vehicles (called vehicular networks) have risen as a key topic of interest. In these networks, the vehicles can exchange data with others by using short-range radio signals in order to get useful information related to traffic conditions, road safety, and others. The availability of different types of sensors can be exploited by the vehicles to measure many parameters from their surroundings. These data can then be shared with other drivers who, on the other side, could also explicitly submit queries to retrieve information available in the network. This can be a challenging task, since the data is scattered among the vehicles belonging to the network and the communication links among them have usually a short life due to their constant movement.

In this paper, we use the technology of mobile software agents to help to accomplish these tasks, since they have a number of features that are very well suited for their use in mobile environments, such as their autonomy, mobility, and intelligence. Specifically, we analyze the benefits that mobile agents can bring to vehicular networks and the potential difficulties for their adoption. Moreover, we describe a query processing approach based on the use of mobile agent technology. We focus on range queries that retrieve interesting information from the vehicles located within a geographic area and present an extensive experimental evaluation that shows the feasibility and the interest of the proposal.

Key words: vehicular ad hoc networks, mobile agents, data management, query processing

Email addresses: ourra@itainnova.es (Oscar Urra), silarri@unizar.es (Sergio Ilarri), raqueltl@unizar.es (Raquel Trillo-Lado)

1. Introduction

Nowadays, it has become easy to acquire computer and communication devices at a low cost. These devices exhibit noteworthy features such as a small size, low power consumption, ease of programming, and affordable price (e.g., the Raspberry Pi [37] or Arduino [14] devices). These devices can be used in mobile scenarios to build systems onto which a number of interesting services and applications can be developed.

Thanks to these features, these devices can be carried on vehicles and they can communicate directly among them (i.e., without needing a pre-existent infrastructure) by using short-range wireless connections that are free to use and follow standard protocols such as Wi-Fi or the more specific 802.11p standard. Thus, the involved vehicles form a network where they can exchange data for short periods of time with other nearby vehicles using direct connections, which is known as a *vehicular ad hoc network* or *VANET* [30]. The data exchanged by the vehicles in the network can be obtained from different sources, such as the vehicle itself (e.g., its current and average speed, its geographic location, etc.) but also from other sensors or devices specifically installed for different purposes, such as environmental sensors (that can measure the air quality, temperature, humidity, etc.) or video-cameras capable of taking images from the surroundings and extracting useful information (e.g., recognizing traffic signals) by using artificial vision techniques. All these types of data can be gathered by vehicles in the VANET in different places and at different times of the day, and stored in local databases to be used later. Then, any driver can ask for specific information (e.g., the traffic conditions in the city center, the number of parking slots in an area, etc.) by issuing a query to the network.

There are many interesting applications that could be developed in a scenario like this, but there exist a number of challenges that must be addressed first. Many of them are a consequence of the short-range communications established among the vehicles. These communication links can be active only for a few seconds, since the involved vehicles are constantly moving, sometimes at high speeds. In fact, some of the vehicles can even disappear completely from the network for unpredictable periods of time, for example if they travel to areas with low density of vehicles or are parked inside a garage. The process of finding the relevant data to process a query in such a chaotic and unstable scenario makes it necessary the use of data management techniques especially suited for vehicular networks [19].

These issues can be addressed by using mobile agent technology [28]. A mobile agent is a special kind of software that runs on an execution environment (called *place*) and is able to transfer itself autonomously from one place to another. Once the transfer is completed correctly, it resumes its execution at the destination place [28]. To run mobile agents in a computer, a middleware called the *mobile agent platform* [36] is necessary, which creates the environment where agents can execute as well as get the platform's support needed to move to other execution environments. The mobile agent platform also offers life cycle functionalities (e.g., to create new agents), communication services to exchange data with other existing agents, security, etc. This platform must be executed on every computer or device in the distributed system that needs the capability to receive and execute agents, and it must offer some interface to communicate with other platforms present in other devices or computers.

Thanks to the mobility capability of mobile agents, it is easy to build complex distributed applications that are at the same time flexible. 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 (i.e., 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 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), which make them very attractive for distributed computing. Particularly, mobile agents can be useful in mobile environments [33, 42, 43]. For example, mobile agents can move to the place where the information is stored and process the data locally, discarding the data that are not relevant and therefore need not be communicated through the wireless network. As another example, they can transport themselves and their data through networks following complicated and changing paths, by continuously evaluating alternatives based on information about their environment. As a final example, they could move to a fixed powerful computer with the required resources to perform a complex task that otherwise could overload the mobile device. This will increase, in turn, its battery life, an important limitation on these devices.

In this paper, we focus on the use of mobile agents in vehicular networks, as mobile agents can provide key advantages thanks to their adaptability and mobility features:

- *They can bring a processing task wherever it is needed*, and *the algorithm that follows 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 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 [17]). In such situations, other protocols can be used, such as carry-and-forward [50], in which 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 adapt dynamically the route to the conditions of the network at any moment.

Mobile agents have been shown to be very useful in wireless environments, but the application of this technology in the specific case of vehicular networks has not been extensively explored yet. Therefore, in this paper, 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 useful information to the drivers or other interested parties, 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 of 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 rest of this paper is organized as follows. In Section 2, we present some related work. In Section 3, we explain our approach based on mobile agents. In Section 4, we present an extensive experimental evaluation that shows the interest and feasibility of our approach. Finally, in Section 5, we present our conclusions and future work.

2. Related Work

In this section, we describe some work that provides the background for our research and motivates it. First, in Section 2.1, we focus on the problem of mobile query processing in vehicular networks. Secondly, in Section 2.2, we describe a few proposals to apply mobile agent technology for intelligent vehicles and transportation. The use of mobile agents in wireless environments is an interesting option that has already been studied in several works [33]. However, when it comes to the specific case of vehicular networks, their potential application has not been explored in depth yet, and therefore we cannot fairly compare our proposal with others that involve the processing of queries in VANETs using mobile agents. For this reason, in Section 2.3, we also describe some works regarding related problems addressed using different approaches.

2.1. Mobile Query Processing in Vehicular Networks

We first present different types of queries that can be interesting in VANETs, and then we describe some possible approaches to the problem of query processing in vehicular networks, along with their limitations.

2.1.1. Queries in a Vehicular Network and Basic Concepts: Area of Interest and Relevant Area

A variety of queries can be interesting in the context of vehicular networks. 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* [46] (which retrieve objects within a certain range/region), *nearest-neighbor queries* [34] (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 [22] (although not in the context of vehicular networks). In this paper we focus on range queries, as nearest-neighbor queries could be processed based on range queries, as suggested in [20].

In range queries and in other types of queries, such as in *constrained nearest-neighbor queries* [15], 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* [16, 20], 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.

2.1.2. Pull-Based Approach: 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 [5]. 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 [19]. 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.

2.1.3. Push-Based Approach: 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 by a query processor in charge of retrieving data relevant for 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 [7, 11, 47].

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 for many vehicles.

2.2. Works Using Mobile Agents for Intelligent Vehicles and Transportation

In [10], mobile agents travel to areas where different environmental sensors are used to collect data using the vehicles from a taxi fleet. This is probably the most similar work to ours, 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 the one presented in this paper, since the authors do not use real cities maps (instead, they use grid-like maps). That work is in the same spirit as a previous work that we had presented in [40]. 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 [38].

In [18], 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.

In [25], 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 [8], 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.

In [31], 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 provide a service), and therefore that work shows how useful they can be in a vehicular network scenario.

Overall, as described in [19], the use of mobile agents in Intelligent Transportation Systems (ITS) and vehicular networks has been quite unexplored so far. This claim can be extended even to standard agents in general, according to [12].

2.3. Other Works on Query Processing in VANETs

In [45], 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 [26], 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 [35], 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 that 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 originator, which will usually be a moving vehicle. The interesting method to locate where the moving originator is consists of what the authors call *breadcrumbs*. They 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 [29, 48, 49]. *PeopleNet* [29] is an infrastructure-based proposal for information exchange in a mobile environment. *Roadcast* [49] 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 [48] 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.

Finally, a survey collecting different methods for routing data in vehicular networks is presented in [4]. 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 both 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. 802.11) are allowed.

3. Proposed Approach Based on Mobile Agents

We argue that alternative and more flexible approaches are possible by using mobile agents. As explained in Section 1, mobile agents can bring interesting benefits for data management in VANETs. 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. Another advantage of mobile agents is that they can move to any place where the data are located, to process and collect only the most relevant data (filtering out unnecessary data). That is, they can perform data aggregation and filtering locally, and thus reduce the network load, which is a key issue in vehicular ad hoc networks where two vehicles may be within communication range of each other only for a short time.

These properties make 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.

Algorithm 1 summarizes the process performed by a mobile agent to process a query. This process involves four steps: 1) defining the query (see Section 3.1), 2) tracking the relevant area (see Section 3.2), 3) collecting the interesting data (see Section 3.3), and 4) returning the results to the query originator (see Section 3.4). In the algorithms that follow, we assume for simplicity that *strong mobility* is supported by the mobile agent platforms (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 [3]. After describing the four steps, we summarize the process in Section 3.5, 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.

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.}

3.1. 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., "the results must be ready in 10 minutes") by using the techniques proposed in [21]. 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 2.1.1, 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 the user intervention), if possible. For example, if the user asks with his/her query about a specific number of available parking spaces, then the *amountOfDataToCollect* is the number specified. As another example, a user may ask about certain information about vehicles within a certain area and there could be statistics about the traffic in that area, and so the *amountOfDataToCollect* could be set as the product of the amount of traffic expected and the number of results to collect in each vehicle. 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 [6], 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. 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 from 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.

3.2. 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 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 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 should 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 *hopping strategies* that the agent can follow to decide if is better to move to the other vehicle or stay in the current one.

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);
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
```

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 hopping strategy must be encoded within the *isBetter(...)* function shown in that algorithm (line 2).

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 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.

Algorithm 3 hopAmongVehiclesToReach(destination)

Require: *destination* is given 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
```

3.3. 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). 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 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, 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 the data collection step in more detail.

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 for the query).

Use of Clones for Data Collection

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. 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 their behavior, 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 function *badDataCollectionRate*() is used. This function 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 maximizes the probability of obtaining an answer.

In Section 4.6, the previous cloning strategy is compared with an alternative where the clones are created at the beginning of the query processing, rather than during the data collection phase.

3.4. 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 usually not the 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 [19], in our prototype we assume for simplicity that the agent has a suitable estimation of the trajectory of the query originator. 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 4 processQuery(dataNeed, relevantArea, deadline, amountOfDataToCollect, minPercentData)

Require: The mobile agent has reached the *relevantArea*. The 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: dataCollectionStartingTimeInstant ← 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.}
16:      end if
17:      if thisIsNotAClone() & badDataCollectionRate(dataCollectionStartingTimeInstant, 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
```

Algorithm 5 returnToQueryOriginator(deadline)

```
1: sucessfulReturn ← false;
2: repeat
3:    queryOriginatorLocation ← estimateQueryOriginatorLocation();
4:    areaOfTheQueryOriginatorLocation ← createSquare(queryOriginatorLocation, COM-
      MUNICATION_RANGE); {A square centered in the estimated location for the query
      originator and with side twice the communication range (e.g., 200 meters) is built.}
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)
```

3.5. 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 in Step 3 of the whole process, 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 new integrated results to the user 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; 4) the agents continuously re-evaluate the situation in order to adapt their behavior, which leads to a good reliability; and 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 useful information to the users (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 provided in [44]. 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.

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 3.1).

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. Experimental Evaluation

In this section, we present the tests that we have performed to study the impact of using mobile agents in vehicular networks. In Section 4.1, we describe the experimental settings. In Section 4.2, we present an experiment performed to measure the travel time of mobile agents in a wireless environment. In Section 4.3, we propose and compare experimentally different hopping strategies. In Section 4.4, we evaluate the influence of

simulating or not buildings that block the wireless signals. In Section 4.5, we analyze the impact of different factors that may affect the reliability of the proposal, such as the amount of relevant data in the vehicles, the vehicle traffic density, or the existence of communication failures, among others. Finally, in Section 4.6, we study the impact that the use of clones could have during the query processing. We have repeated each experiment fifty times with correlative random seeds (as generators of random decisions during each experiment), so that the initial positions of the vehicles and the relevance of their carried data are different for every experiment. We report the average values of the obtained results.

4.1. Experimental Settings

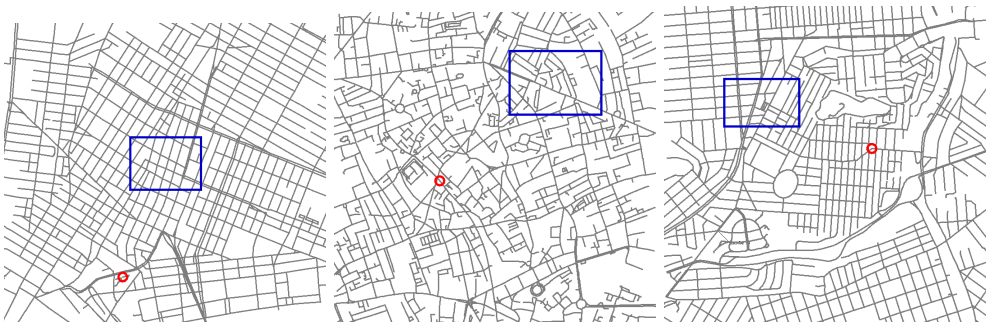
In this section, we present the general experimental settings (see Table 1), that have been used in the experiments performed unless specified otherwise in the individual descriptions of the experiments (several experiments evaluate the impact of variations of these parameters). For evaluation, we consider real road networks (extracted from *OpenStreetMap*, `http://www.openstreetmap.org`) 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* [1] (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 of 50 $Km/h \pm 10\%$.

Parameter	**Default value**
Map dimensions	4 $Km\times$ 4 Km
Size of the relevant area	0.25 Km^2
Distance to the relevant area	1000 m
Density of vehicles	Medium (100 vehicles / Km^2)
Speed of the vehicles	50 $Km/h \pm 10\%$
Mobility model	Pathway mobility model
Hop strategy	MAP (map distance)
Total agent size	200 KB
Min. % of data to retrieve	100%
Communication range	250 m
Latency of transmission of a mobile agent (hop delay)	1 second
Communication bandwidth	54 Mbps (IEEE 802.11g)
Buildings block communication signals	Yes
Relevance of information	50% of vehicles

Table 1: Configuration parameters

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 (see Figure 1): New York (squared layout, with long and straight streets and avenues), London (*old city* layout, with many short and curved streets) and San Francisco (mixed layout). In all cases the initial position of the vehicle (highlighted in the figures with a small circle) is at a distance of 1000 meters from the relevant area (highlighted with a rectangle), which has a surface of 0.25 square kilometers.

The wireless communication range between vehicles is 250 meters [27] with a bandwidth of 54 Mbps (nominal transfer rate of IEEE 802.11g). Moreover, buildings are



(a) New York (USA) (b) London (UK) (c) San Francisco (USA)

Figure 1: Map fragments used for evaluation

simulated appropriately, determining the impact they have on the signal propagation.

The simulator used to perform the experiments [39, 41] has been developed to test easily different configurations and data management approaches based on mobile agents. For example, we can easily change the query processing algorithms used by the agents, their hopping strategies, the mobility models used by the vehicles, the underlying road network infrastructure, etc., and obtain statistics about different performance parameters (such as the number of communications performed, or the partial times invested in each step of the query processing). For more information, see `http://sid.cps.unizar.es/MAVSIM`.

4.2. 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 first 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 [23], 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 2(a)) with the following features and roles: 1) a Samsung Galaxy Tab *tablet* with 512 MB of RAM and Android 2.3, that hosts the 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 the 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 the place *C3*. They communicate among them by connecting to the same wireless network through a IEEE 802.11g Wi-Fi router. Figure 2(b) shows an example of the graphical user interface to launch queries: the performance of the query processing will be evaluated in other tests of our experimental evaluation.

For this experiment, a mobile agent visits the three previously-described devices measuring the time taken to hop from one place to another. First, the mobile agent starts its execution on place C3, hops to place C1, and returns again to C3. This is repeated

(a) Devices used to measure the hop times (b) Graphical user interface to launch a query

Figure 2: Examples of mobile devices considered

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 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 3 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

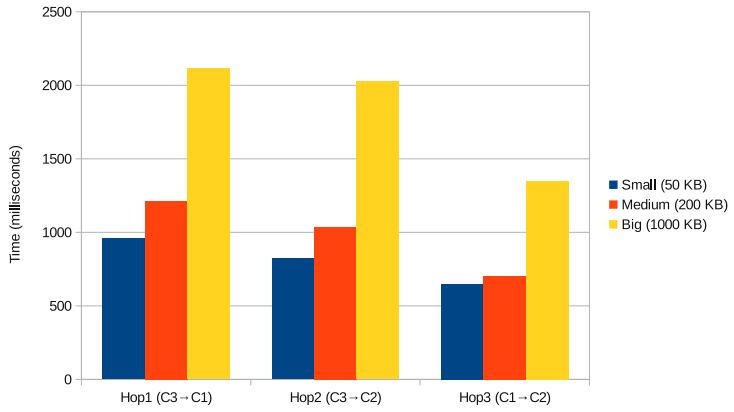


Figure 3: Hop times for a mobile agent hopping among different devices

to the ones described in this paper is not expected to exceed a size of 200 KB. We also assume that the vehicles carry 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 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 the rest of the experiments presented in this section.

4.3. Best Hopping Strategy

In this experiment, we compare different hopping 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. We evaluate six hopping strategies:

- *Frontal angle (ANG).* The angle of direction of the target vehicle regarding the target area is considered. The agent hops if this angle is less than 90° and the current vehicle where the agent is traveling does not satisfy this condition.
- *Euclidean distance (EUC).* This is a simple strategy. The agent hops to another vehicle within the coverage area if that vehicle is nearest to the target area than the current one. To compute the distance, the classic Euclidean distance is used.
- *Encounter Probability (EP).* This measure is presented in [11], 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. The agent hops if, by hopping, the EP increases.
- *Vehicle digital map (MAP).* With this strategy, the involved vehicles are assumed to have digital road maps of their surroundings. The route to the destination is

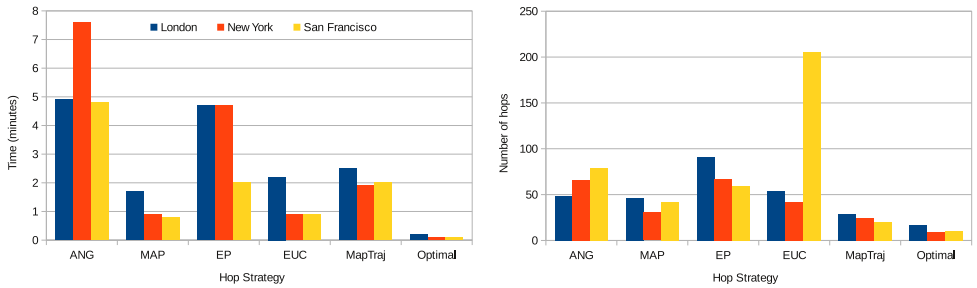
computed following the street layout and the total distance is obtained. The agent hops when another vehicle within the communication range has a route whose length is smaller than the current one.

- *Trajectories using maps (MapTraj).* With this strategy, the vehicles also have digital road maps of the area and they follow a pre-computed route (as they would do with a GPS navigator). The agent hops only when another vehicle follows a route that will travel through the target area. So, as opposed to MAP, this strategy assumes knowledge about the routes of the vehicles.

- *Optimal route (Optimal).* This 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. We use this *strategy* as an unreachable baseline to compare how efficient the other strategies are.

4.3.1. 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) with the different hopping strategies, for a medium vehicle density, can be seen in Figure 4(a). 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 optimum strategy) is MAP, taking less than two minutes for data collection in the worst case.



(a) Data collection time

(b) Number of hops

Figure 4: Performance of the hopping strategies

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 stated before, 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 all over the 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

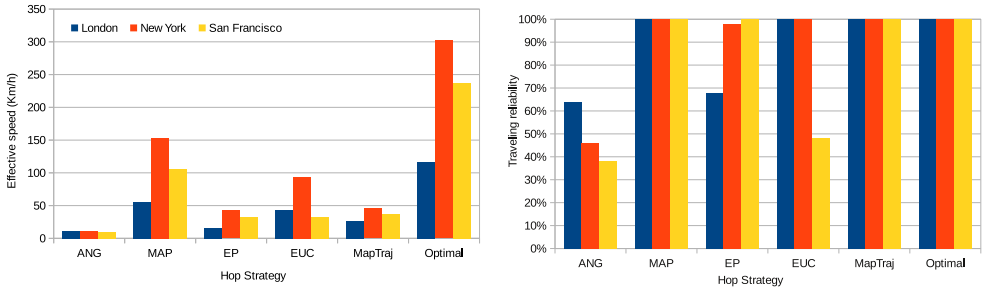
be accurate (otherwise, the mobile agent could take incorrect decisions). In the case of MapTraj, it also needs to know the planned trajectories that the vehicles will follow, which is not very realistic since it is unlikely that all the drivers in the area will use GPS navigators. Moreover, privacy 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 4(b), it needs a higher number of hops than MapTraj.

4.3.2. 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 5(a) 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.



(a) Effective traveling speed

(b) Traveling reliability

Figure 5: Effective speed and reliability of the mobile agent traveling to the target area

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 5(b), 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%.

4.3.3. Conclusions

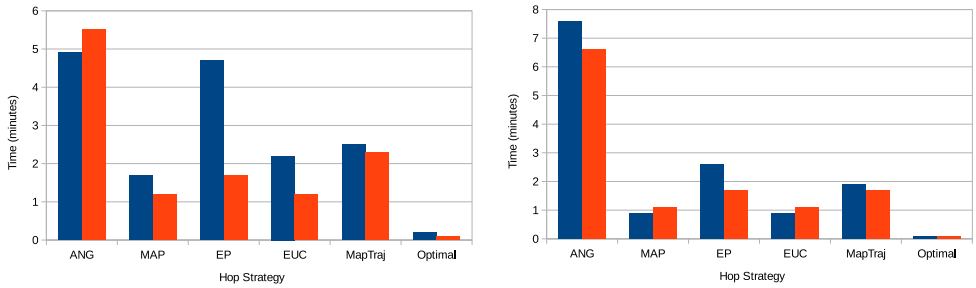
Since MAP is the best strategy when considering globally all the key features studied (data collection time, network overhead in terms of the number of hops, effective traveling speed, and traveling reliability) and besides it does not need to know in advance the routes followed by vehicles (as opposed 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 hopping 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 described 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.

4.4. Influence of Buildings in Urban Scenarios

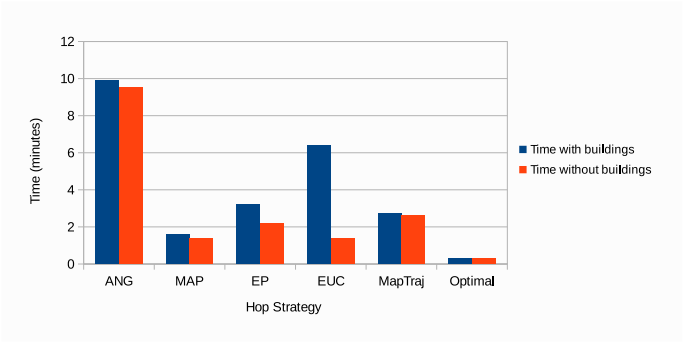
The short-range wireless communications used in a VANET operate with radio signals with a very low depth of penetration [32]. Therefore, buildings and other obstacles can block the signal propagation, making the communication among vehicles possible only if they have a direct line of sight. This means 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 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 [27] to compare their influence on the data collection time for scenarios with medium traffic density. As can be seen in Figure 6, 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 the 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 and the buildings (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: 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 in our experimental evaluation.

4.5. Analysis of Factors Affecting the Reliability of the Query Processing

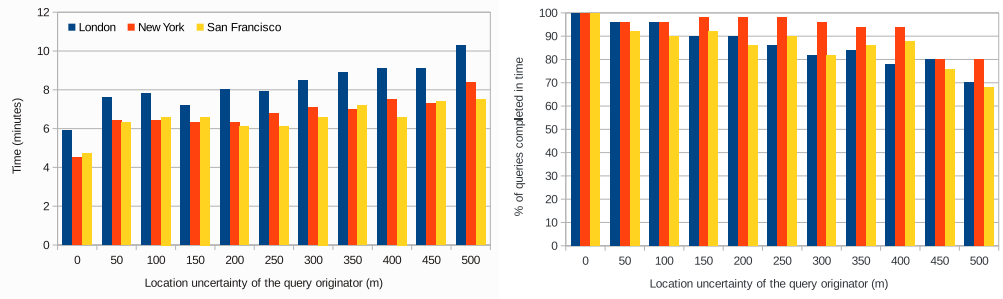
In this section, we perform some experiments to evaluate the impact of elements that can affect the reliability and performance of the proposal. Section 4.5.1 evaluates the impact of an uncertain query originator's location when the results have to be returned to a vehicle that submitted the query. Section 4.5.2 analyzes the impact of the amount of relevant information available in the vehicles. Section 4.5.3 studies the reliability and performance for several degrees of vehicle traffic density. Section 4.5.4 analyzes the reliability of the approach when there are communication failures. Finally, Section 4.5.5 includes experiments with scenarios where vehicles change their routes unexpectedly.

4.5.1. Influence of the Uncertainty of the Location of the Query Originator

As commented in Section 3.4, 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 paper 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). 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. The MAP hopping strategy is used.

Figure 7(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 circular area of radius 500 meters). The query processing time increases with the location uncertainty, as expected, as 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 7(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%.



(a) Query processing time

(b) Queries processed

Figure 7: Impact of the location uncertainty of the query originator on the query processing

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 Road Side Units) when they are available and the extra economic cost that their use implies is affordable [19], or by using mailboxes to store the query results at fixed locations to be retrieved by the query originator [13].

4.5.2. Influence of the Relevance of Information

In the use case 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. The hopping strategy used is MAP and a medium traffic density is considered. Figure 8 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.

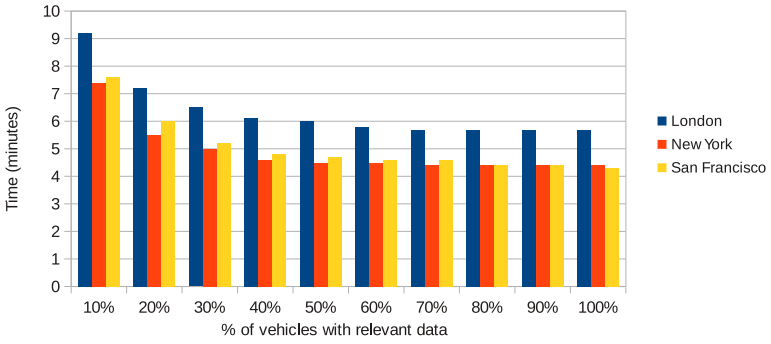


Figure 8: Impact of the amount of vehicles with relevant data on the query processing

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.

4.5.3. Influence of the Vehicle Density

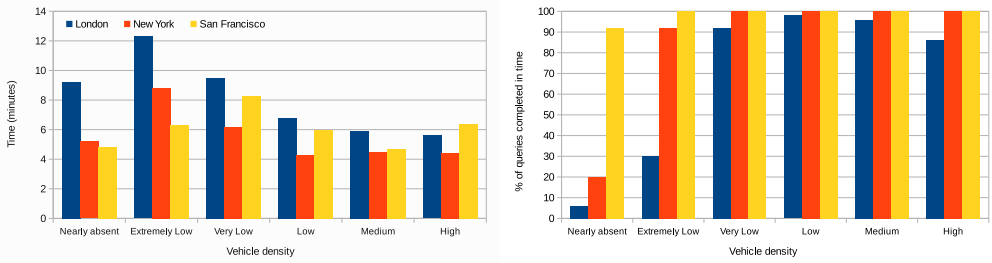
In this experiment, we evaluate the performance and reliability of different values of vehicles density. The vehicle density can be defined as the number of vehicles present per surface unit [9] and it can be measured in terms of vehicles per square kilometer. Another similar vehicle density measure unit is the number of vehicles per road length unit [24], 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. 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 squares or street junctions. For these reasons, we use the *vehicles per square kilometer* (or v/Km^2) unit, since it is more appropriate for urban scenarios like those

we are testing. The density values used in the simulation are inspired by those used in works such as [2, 27].

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*), 50 v/Km^2 (*low*), 100 v/Km^2 (*medium*), and 200 v/Km^2 (*high*). The hopping strategy used is MAP.

Figure 9(a) 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.

By looking only at Figure 9(a), 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 9(b) 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%). Even with *extremely low* traffic values the percentage of success is quite high (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.



(a) Query processing time

(b) Queries processed

Figure 9: Impact of the density of vehicles on the query processing

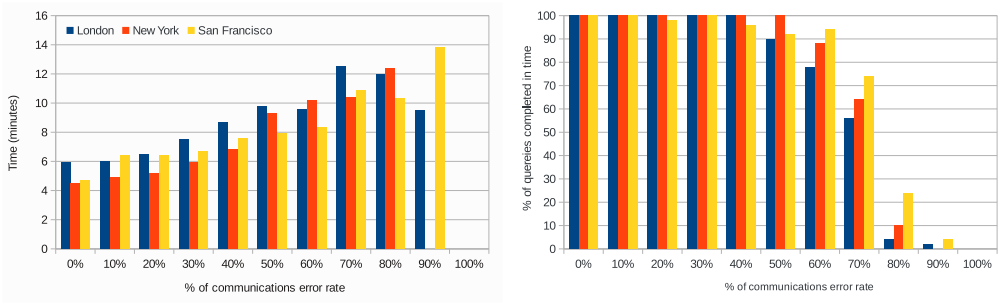
As all the queries are completed with *low* traffic (and higher traffic densities) and most of them are completed with *very low* traffic, we can conclude that the proposal performs well in most scenarios, even if the traffic density is low.

4.5.4. 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 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 4.4); so, a building may block a communication even if communication error rates are not simulated.

The hop strategy used is MAP and we vary the communication error rate from 0% (no extra failures simulated) to 100% (all the communications fail always). Figure 10(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 10(a) has to be analyzed by considering also Figure 10(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 10(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.



(a) Query processing time

(b) Queries processed

Figure 10: Impact of communication failures on the query processing

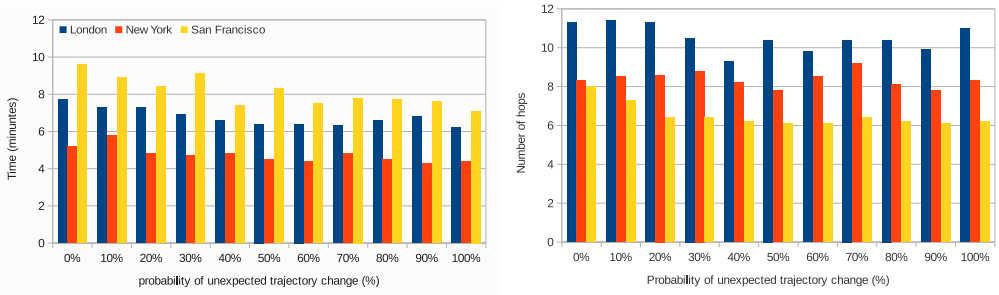
These experiments show that, as expected, the query processing approach performs better when the wireless communication medium is reliable, but that the approach is also flexible enough to adapt to communication failures. Mobile agents are able to react to communication failures and the communication failures need to be quite high to affect the reliability of the query processing.

4.5.5. Influence of Unexpected Changes in Routes

In this section, we analyze the reliability and performance of the query processing approach when vehicles change their route unexpectedly. We focus on the *MapTraj* hopping 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 worse 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 the 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 11(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 11(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 11(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* hopping strategy becomes similar in practice to the *MAP* strategy.



(a) Query processing time

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

Figure 11: Impact of trajectory changes on the query processing for the MapTraj strategy

4.6. Influence of the Use of Clones

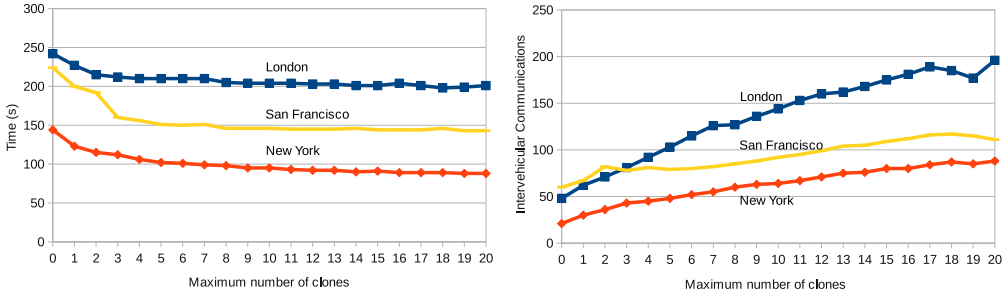
In this experiment, our goal is to evaluate the potential benefits of using clones and the impact of the specific number of clones created during the query processing (see Section 3.3). With that purpose, in the following experiments we set the value of *minPercentData* (minimum acceptable percentage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

4.6.1. 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, the clones are created only in the data collection phase if they are needed to increase the observed collection rate.

In Figure 12(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 contributes to a higher overhead in the network, as shown in Figure 12(b). Thus, above a certain number of clones there is little improvement and the overhead introduced does not pay off.



(a) Query processing time

(b) Network overhead

Figure 12: Use of clones for data collection

4.6.2. Cloning Strategy 2: Clones for the Whole Process

In the second experiment, the mobile agent creates copies of itself in the fist step of the process (all at once), and they hop randomly among the nearby vehicles for a certain interval of time. Afterwards, they start using the predefined hop strategy (MAP) to travel to the target area executing individually the algorithms described in Section 3, and gathering the collected data in the origin. 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, increasing in this way 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.

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 13, 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.

4.6.3. Comparison of Both Cloning Strategies

Figures 14(a) and 14(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

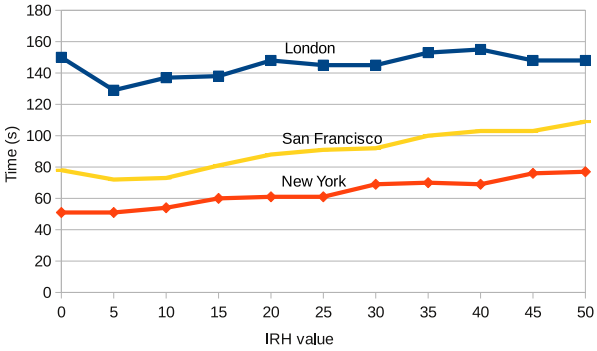
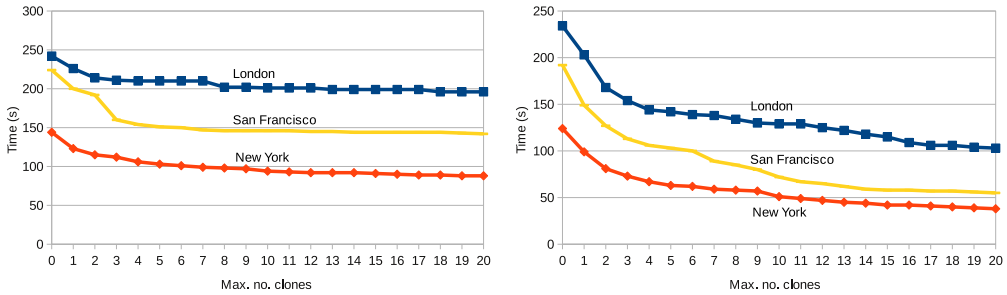


Figure 13: Query processing time for different values of IRH in different scenarios

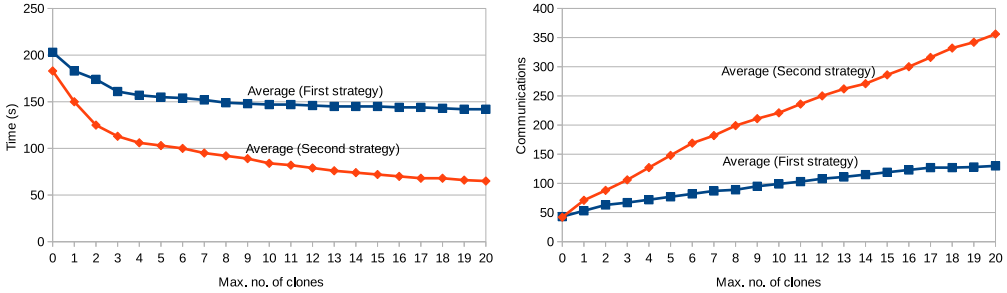
more clearly in Figure 15(a), that compares the average of the processing times. However, as shown in Figure 15(b), the network overhead for the second cloning strategy is higher than when using the first one, since all the agents are created at the beginning of the process.



(a) First cloning strategy

(b) Second cloning strategy with IRH=5

Figure 14: Time to solve the query with two cloning strategies in different scenarios

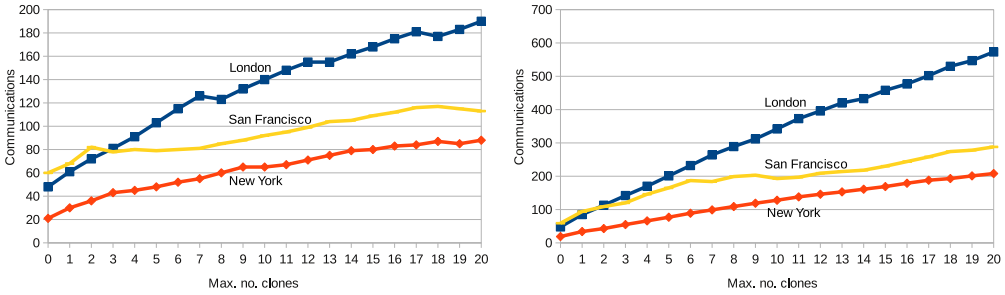


(a) Query processing time

(b) Network overhead

Figure 15: Comparison of two different cloning strategies

Regarding the city topology, as Figures 16(a) and 16(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 16: Network overhead using two cloning strategies in different scenarios

5. Conclusions and Future Work

Mobile agents have been proved to be an interesting technology for query processing in distributed and mobile environments. However, their use for data management in vehicular networks is a largely unexplored area that calls for new research efforts. In this paper, we have proposed the use of mobile agents for processing queries in vehicular ad hoc networks. Mobile agents have the ability to hop from vehicle to vehicle as needed, not only for collecting relevant data but also to reach a target area. Thus, the vehicles act not only as data sources but also as intermediate points and physical carriers of the agents, carrying with them both data and queries wherever they are needed. Moreover, they can process the relevant data in the vehicles, aggregating them if necessary or discarding unnecessary data. Besides, they can adapt dynamically to the environment and change their routing strategy if needed in order to arrive in a certain area or come back to the moving query originator. Finally, the flexibility of mobile agents allows to bring any required query processing to the relevant vehicles. Summing up, mobile agents offer a useful abstraction level and provide interesting advantages for vehicular networks.

We have presented an in-depth experimental evaluation that shows promising results that indicate the feasibility and interest of using mobile agents in vehicular ad hoc networks for data management. In the future, we plan to analyze the use of mobile agents in other application scenarios. For example, it could be interesting to study the benefits that they can provide to build vehicular social networks or for cooperative driving. We also plan to extend the mobile agent platform SPRINGS to adapt it better to the features and needs of mobile P2P environments and vehicular networks.

Acknowledgments

The authors acknowledge the support of the CICYT project TIN2013-46238-C4-4-R. We would like to thank Thierry Delot for his comments on a very preliminary and seminal version of our work. Last but not least, we are very grateful for the insightful and useful comments provided by the reviewers.

References

[1] F. Bai, A. Helmy, Wireless Ad Hoc and Sensor Networks, chap. "A Survey of Mobility Modeling and Analysis in Wireless Adhoc Networks", Springer, 2006.

[2] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, C. T. Calafate, P. Manzoni, Road side unit deployment: A density-based approach, IEEE Intelligent Transportation Systems Magazine 5 (3) (2013) 30–39.

[3] L. Bettini, R. D. Nicola, Translating strong mobility into weak mobility, in: Fifth Int. Conf. on Mobile Agents (MA'01), Springer, 2002, pp. 182–197.

[4] S. M. Bilal, C. J. Bernardos, C. Guerrero, Position-based routing in vehicular networks: A survey, Journal of Network and Computer Applications 36 (2) (2013) 685–697.

[5] S. Biswas, R. Morris, ExOR: opportunistic multi-hop routing for wireless networks, ACM SIGCOMM Computer Communication Review 35 (4) (2005) 133–144.

[6] A. Bonifati, P. K. Chrysanthis, A. M. Ouksel, K.-U. Sattler, Distributed databases and peer-to-peer databases: past and present, SIGMOD Record 37 (2008) 5–11.

[7] N. Cenerario, T. Delot, S. Ilarri, A content-based dissemination protocol for VANETs: Exploiting the encounter probability, IEEE Transactions on Intelligent Transportation Systems 12 (3) (2011) 771–782.

[8] B. Chen, H. Cheng, J. Palen, Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems, Transportation Research Part C 17 (1) (2009) 1–10.

[9] T. Darwish, K. A. Bakar, Traffic density estimation in vehicular ad hoc networks: A review, Ad Hoc Networks 24 (2015) 337–351.

[10] E. P. de Freitas, T. Heimfarth, L. A. G. Costa, A. M. Ferreira, C. E. Pereira, F. R. Wagner, T. Larsson, Analyzing different levels of geographic context awareness in agent ferrying over VANETs, in: 2011 ACM Symposium on Applied Computing (SAC 2011), ACM, 2011, pp. 413–418.

[11] T. Delot, N. Cenerario, S. Ilarri, Vehicular event sharing with a mobile peer-to-peer architecture, Transportation Research Part C: Emerging Technologies 18 (4) (2010) 584–598.

[12] T. Delot, S. Ilarri, M. del Carmen Rodríguez-Hernández, Intelligent transportation systems – maybe, but where are my agents?, in: Sixth Int. Conf. on Ad Hoc Networks (AdHocNets 2014), Rhodes (Greece), vol. 140 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST), Springer, 2014, pp. 39–50.

[13] T. Delot, N. Mitton, S. Ilarri, T. Hien, GeoVanet: A routing protocol for query processing in vehicular networks, Mobile Information Systems 7 (4) (2011) 329–359.

[14] B. Evans, Beginning Arduino Programming, Apress, 2011.

[15] H. Ferhatosmanoglu, I. Stanoi, D. Agrawal, A. E. Abbadi, Constrained nearest neighbor queries, in: Seventh Int. Symposium on Advances in Spatial and Temporal Databases (SSTD'01), vol. 2121 of Lecture Notes in Computer Science (LNCS), Springer, 2001, pp. 257–276.

[16] B. Gedik, L. Liu, MobiEyes: Distributed processing of continuously moving queries on moving objects in a mobile system, in: Ninth Int. Conf. on Extending Database Technology (EDBT'04), vol. 2992 of Lecture Notes in Computer Science (LNCS), Springer, 2004, pp. 67–87.

[17] W. R. Heinzelman, J. Kulik, H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks, in: Fifth Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom'99), 1999, pp. 174–185.

[18] X. Hu, J. Zhao, B.-C. Seet, V. Leung, T. Chu, H. Chan, S-Aframe: Agent-based multilayer framework with context-aware semantic service for vehicular social networks, IEEE Transactions on Emerging Topics in Computing 3 (1) (2015) 44–63.

[19] S. Ilarri, T. Delot, R. Trillo-Lado, A data management perspective on vehicular networks, IEEE Communications Surveys and Tutorials 17 (4) (2015) 2420–2460.

[20] S. Ilarri, E. Mena, A. Illarramendi, Location-dependent queries in mobile contexts: Distributed processing using mobile agents, IEEE Transactions on Mobile Computing 5 (8) (2006) 1029–1043.
[21] S. Ilarri, E. Mena, A. Illarramendi, Using cooperative mobile agents to monitor distributed and dynamic environments, Information Sciences 178 (9) (2008) 2105–2127.
[22] S. Ilarri, E. Mena, A. Illarramendi, Location-dependent query processing: Where we are and where we are heading, ACM Computing Surveys 42 (3) (2010) 1–73.
[23] S. Ilarri, R. Trillo, E. Mena, SPRINGS: A scalable platform for highly mobile agents in distributed computing environments, in: Fourth Int. WoWMoM 2006 workshop on Mobile Distributed Computing (MDC'06), IEEE, 2006, pp. 633–637.
[24] B. S. Kerner, Introduction to modern traffic flow theory and control, Springer, 2009.
[25] R. Kumar, D. M. Dave, Mobile agent as an approach to improve QoS in vehicular ad hoc network, IJCA Special Issue on MANETs (2010) 67–72.
[26] U. Lee, J. Lee, J.-S. Park, M. Gerla, Fleanet: A virtual market place on vehicular networks, IEEE Transactions on Vehicular Technology 59 (1) (2010) 344–355.
[27] F. J. Martinez, M. Fogue, C. K. Toh, J.-C. Cano, C. T. Calafate, P. Manzoni, Computer simulations of VANETs using realistic city topologies, Wireless Personal Communications 69 (2) (2012) 639–663.
[28] D. Milojicic, F. Douglis, R. Wheeler, Mobility: processes, computers, and agents, ACM, 1999.
[29] M. Motani, V. Srinivasan, P. S. Nuggehalli, PeopleNet: engineering a wireless virtual social network, in: 11th Annual Int. Conf. on Mobile Computing and Networking (MobiCom'05), ACM, 2005, pp. 243–257.
[30] S. Olariu, M. C. Weigle (eds.), Vehicular Networks: From Theory to Practice, Chapman & Hall/CRC, 2009.
[31] O. Riva, T. Nadeem, C. Borcea, L. Iftode, Context-aware migratory services in ad hoc networks, IEEE Transactions on Mobile Computing 6 (12) (2007) 1313–1328.
[32] C. Sommer, D. Eckhoff, R. German, F. Dressler, A computationally inexpensive empirical model of IEEE 802.11p radio shadowing in urban environments, in: Eighth Int. Conf. on Wireless On-Demand Network Systems and Services (WONS 2011), IEEE, 2011, pp. 84–90.
[33] C. Spyrou, G. Samaras, E. Pitoura, P. Evripidou, Mobile agents for wireless computing: the convergence of wireless computational models with mobile-agent technologies, Mobile Networks and Applications 9 (5) (2004) 517–528.
[34] Y. Tao, D. Papadias, Q. Shen, Continuous nearest neighbor search, in: 28th Int. Conf. on Very Large Data Bases (VLDB'02), VLDB Endowment, 2002, pp. 287–298.
[35] J. Timpner, M. Wozenilek, L. Wolf, Breadcrumb routing: Query-response geocast for mobile originators in vehicular networks, in: Fifth IEEE Vehicular Networking Conf. (VNC 2014), IEEE, 2014, pp. 45–52.
[36] R. Trillo, S. Ilarri, E. Mena, Comparison and performance evaluation of mobile agent platforms, in: Third Int. Conf. on Autonomic and Autonomous Systems (ICAS'07), IEEE, 2007, p. 41, 6 pages.
[37] E. Upton, G. Halfacree, Raspberry Pi user guide, John Wiley & Sons, 2014.
[38] O. Urra, S. Ilarri, Using mobile agents in vehicular networks for data processing, in: 14th Int. Conf. on Mobile Data Management (MDM 2013), Ph.D. Forum, vol. 2, IEEE, 2013, pp. 11–14.
[39] O. Urra, S. Ilarri, Cognitive Vehicular Networks, chap. "MAVSIM: Testing VANET Applications Based on Mobile Agents", CRC Taylor and Francis Group, 2016, pp. 199–224, print ISBN 978-1-4987-2191-2, eBook ISBN 978-1-4987-2192-9.
[40] O. Urra, S. Ilarri, T. Delot, E. Mena, Using hitchhiker mobile agents for environment monitoring, in: Seventh Int. Conf. on Practical Applications of Agents and Multi-Agent Systems (PAAMS'09), Springer, 2009, pp. 557–566.
[41] O. Urra, S. Ilarri, E. López, Simulating mobile agents in vehicular networks, in: XIX Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2014), Cádiz (Spain), ISBN-13: 978-84-697-1152-1, ISBN-10: 84-697-1152-0, 2014, pp. 71–84.
[42] O. Urra, S. Ilarri, E. Mena, Testing mobile agent platforms over the air, in: First ICDE Workshop on Data and Services Management in Mobile Environments (DS2ME'08), IEEE, 2008, pp. 152–159.
[43] O. Urra, S. Ilarri, R. Trillo, E. Mena, Mobile agents and mobile devices: Friendship or difficult relationship?, Journal of Physical Agents. Special Issue: Special Session on Practical Applications of Agents and Multiagent Systems 3 (2) (2009) 27–37.
[44] T. L. Willke, P. Tientrakool, N. F. Maxemchuk, A survey of inter-vehicle communication protocols and their applications, IEEE Communications Surveys and Tutorials 11 (2) (2009) 3–20.
[45] O. Wolfson, B. Xu, H. J. Cho, Multimedia traffic information in vehicular networks, in: 17th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (GIS'09), ACM, 2009, pp. 480–483.

[46] K.-L. Wu, S.-K. Chen, P. S. Yu, Incremental processing of continual range queries over moving objects, IEEE Transactions on Knowledge and Data Engineering 18 (11) (2006) 1560–1575.
[47] B. Xu, A. M. Ouksel, O. Wolfson, Opportunistic resource exchange in inter-vehicle ad-hoc networks, in: Fifth Int. Conf. on Mobile Data Management (MDM'04), IEEE, 2004, pp. 4–12.
[48] B. Xu, F. Vafaee, O. Wolfson, In-network query processing in mobile P2P databases, in: ACM Int. Conf. on Advances in Geographic Information Systems (GIS'09), ACM, 2009, pp. 207–216.
[49] Y. Zhang, J. Zhao, G. Cao, Roadcast: A popularity aware content sharing scheme in VANETs, in: 29th Int. Conf. on Distributed Computing Systems (ICDCS'09), IEEE, 2009, pp. 223–230.
[50] J. Zhao, G. Cao, VADD: Vehicle-assisted data delivery in vehicular ad hoc networks, IEEE Transactions on Vehicular Technology 57 (3) (2008) 1910–1922.