Simulating Mobile Agents in Vehicular Networks

Óscar Urra, Sergio Ilarri, and Eduardo López

Department of Computer Science and System Engineering
University of Zaragoza
ourra@ita.es, silarri@unizar.es, eduardologa@gmail.com

Abstract. In the next years, vehicular ad hoc networks (VANETs) are expected to become a reality and a great number of interesting applications for drivers and passengers will be developed (related to safety, comfort, and entertainment). In all these applications, the acquisition, management and an efficient and effective exchange of data will be key issues. Therefore, significant data management challenges arise in this context. We argue that mobile agent technology could play an important role as a middleware for the development of applications for vehicular networks, as they naturally support disconnected operations and distributed data management. Overall, a development of solutions based on agents that autonomously take decisions may be promising.

A significant problem arises when we need to evaluate a data management approach for vehicular networks. Deploying it in the real-world in order to evaluate the proposal with real cars is impractical and very expensive, and so field tests are limited to very small-scale and controlled scenarios, mainly as a proof of concept. Instead, simulators are usually used for experimental evaluation. However, existing simulators for vehicular networks do not directly support testing data management techniques based on the use of mobile agents. In this paper, we present a simulator that offers interesting functionalities for that context. The simulator has been developed in a quite generic and extensible way, in order to facilitate its use in a variety of scenarios to test different data management approaches.

Keywords: vehicular networks, mobile agents, simulations, mobile computing, mobile ad-hoc networks

1 Introduction

In the last years, continuous technological advances have lead to the popularity of small computing devices with improved processing and communication capabilities. These devices are portable and have a small size and a moderate price, which has encouraged their adoption and nowadays they are ubiquitous. By exploiting available communication networks (e.g., Wi-Fi hot-spots or mobile phone networks), these devices could access and share a huge amount of data. In particular, mobile peer-to-peer networks (mobile P2P networks) [1], where the
devices dynamically form an ad hoc network using short-range wireless communications without the need of any support infrastructure, can provide significant benefits over traditional centralized architectures.

When the participating nodes are vehicles traveling along the roads or streets of a certain area, then we have a special case of mobile P2P network, called a *vehicular ad hoc network* (VANET) [10]. Vehicular networks are expected to become a reality in the upcoming years and a great number of interesting applications for drivers and passengers will be developed. However, VANETs imply a number of significant challenges from the point of view of data management [3–5], related to the acquisition, processing and the efficient and effective exchange of data among vehicles. Beyond the traditional limitations of mobile computing scenarios (e.g., lower performance in comparison with more powerful desktop computers, changing environment, limited battery lifetime of mobile devices carried by users, etc.), many of the difficulties are a consequence of the use of short-range wireless communication technologies (e.g., about 200 meters), which are unstable, subject to frequent disconnections (network partitioning), have a limited communication range, and are subject to security attacks (since the transmitted data are broadcasted).

From our point of view, the use of mobile agent technology in this environment is a research avenue worth exploring. Mobile agents are autonomous software entities that have the ability to transfer themselves from one execution environment to another by using a communication network. We believe that mobile agents could play an important role as a middleware for the development of applications for vehicular networks, as they naturally support disconnected operations and distributed data management. Overall, they can be an interesting design abstraction in a variety of scenarios.

If we focus on the test phase of the development of applications and data management strategies for vehicular networks, it is clearly very expensive and inconvenient to perform the software testing using real devices in real cars. So, real-world tests are usually limited to very small and controlled scenarios, mainly as a proof of concept or to obtain measures that can be used to fine-tune some simulator. The most logical alternative is indeed to simulate the behavior of the developed software in a controlled and simplified environment that can be managed more easily and that supports a large-scale simulation with a high number of vehicles. There exists a variety of simulation software to test and analyze with the required precision different aspects of communication networks and vehicular traffic. However, as far as we know none of the most popular simulators can be easily used to simulate scenarios with the goal of evaluating data management solutions based on the use of mobile agents. Motivated by this, we have developed a vehicular network simulator in which mobile agents can also be simulated, as well as a set of tools intended to ease the analysis of the results obtained in a variety of scenarios.

The rest of this paper is structured as follows. In Section 2, we briefly describe the technological context of our research area. In Section 3, we focus on the
development of our own simulator, describing its architecture and functionalities. Finally, in Section 4, we summarize our conclusions and some lines of future work.

2 Technological Context

In this section, we present an overview of the most important elements present in the considered scenario: vehicular networks, mobile agents, and simulators.

2.1 VANETs

A *vehicular ad hoc network (VANET)* [10] is a highly mobile network whose nodes are vehicles traveling along roads or highways. They can establish connections with other nearby vehicles, and in this way they can exchange relevant information for drivers.

In a vehicular ad hoc network, the vehicles are equipped with short-range wireless communication devices (such as Wi-Fi or UWB) and can establish connections with other nearby vehicles (*vehicle-to-vehicle or –V2V– communications*) in a peer-to-peer way. Using this type of networks in vehicular applications has a number of advantages over a traditional client-server approach, such as: 1) there is no need of a dedicated centralized support infrastructure (expensive to deploy and maintain); 2) the users do not need to pay for the use of these networks; and 3) it allows a very quick and direct (i.e., without intermediate proxies or routers) exchange of information between two vehicles that are within range of each other, which may be critical for safety applications for vehicular networks. Moreover, many application scenarios do not need to communicate with a specific target vehicle but with all the vehicles within a certain area, and therefore broadcast (the typical communication mechanism in VANETs) is a suitable choice.

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

VANETs open up a wide range of opportunities to develop interesting systems for drivers. Although safety applications are usually emphasized, there are also interesting applications related to comfort, entertainment, and travel efficiency. However, a number of difficulties can also arise. Some of these difficulties are due to the fact that two vehicles can communicate directly only if they are
near each other (the range of the wireless communication devices may be limited in practice to about 100-200 meters), which may leave a small time window available for potential data exchange. As the vehicles are constantly moving, the duration during which a communication link is alive can be very short (a few seconds), especially when two nearby vehicles move quickly in opposite directions. Also due to the short communication range, it is possible that there exists no direct connection between two vehicles in the network, in which case the use of some multi-hop communication protocol is necessary. These protocols are usually complex and it is difficult to guarantee an upper bound on the amount of time needed to deliver a message to a recipient, due to the fact that the existing links change constantly.

The development of applications for vehicular networks requires taking these constraints into account. Besides, they may need to consider also other special features of this dynamic context: the geographic area of interest could spread over a large extension, users (usually, the drivers) can move constantly at different speeds and directions (sometimes according to predefined routes, but not necessarily), the density of the network nodes can vary depending on the place or the time of the day, etc.

2.2 Mobile Agents

Mobile agents are software entities that run on an execution environment (traditionally called place), and can autonomously travel from place to place (within the same computer or between different computers) and resume their execution at the target [9]. Thus, they are not bound to the computer/device where they are initially created and they can move freely among computers/devices. To be able to use mobile agents it is necessary to execute a middleware known as a mobile agent platform [11], which provides agents an environment where they can execute as well as the ability to move to other execution environments and other services (e.g., communication, security, persistence, etc.). This platform must be executed on every computer or device that could host agents in the distributed system, and it must offer some interface to communicate with other platforms present in other devices or computers. Key functions provided by a mobile agent platform is its ability to locate other available platforms and transmit the code of an agent from the origin to the destination of the agent’s movement.

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

The autonomy, intelligence, and movement capabilities of mobile agents render them a powerful and flexible tool to build distributed systems, especially in mobile environments. For example, a mobile agent could be programmed to visit certain devices in a complex network whose nodes are mobile devices and (once it reaches devices storing relevant data) to process the local data available there in order to collect new interesting data, and finally to return to the origin with the final relevant data collected. So, mobile agents can move the processing to the data source instead of bringing all the data to the node that will perform the processing (thus reducing the amount of data communicated). Other interesting advantages of mobile agents include their ability to support disconnected operations (an agent can live outside its home device, which can be turned off while the agent is performing its tasks elsewhere), to exploit the most suitable resources available (e.g., using powerful fixed computers instead of the limited resources of a mobile device, when appropriate), and to minimize network connections (as opposed to a traditional client/server approach, that requires a connection open and alive while the request is being performed, an equivalent request processing using mobile agents would only require the connection active during the movements of the agents).

2.3 Using Mobile Agents in VANETs

As we have seen, VANETs offer interesting opportunities for the development of applications for vehicles. However, they also introduce some difficulties. For example, in VANET applications, data may need to be transported from vehicle to vehicle in order to reach locations that are not directly accessible due to the short range of the wireless communications used. Then, two major problems arise. First, as the propagation of the data can be slow, the information can be outdated when it reaches its destination. Second, it can be difficult to determine the destination itself and how to reach it. Given the highly mobile nature of a VANET, the destination could be a single specific vehicle, every vehicle present in a geographic area, all those vehicles matching a certain condition, etc.

To deal with these drawbacks, mobile agents can be very useful because of their adaptability and mobility features. Thus, they can bring a processing task wherever it is needed, and the algorithm or agent’s logic can be changed at any time by deploying new versions of the agent code. This flexibility is
quite 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.

Another important advantage of mobile agents in vehicular networks is that 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 their size smaller (irrelevant data are discarded), and making it easier to transmit them in a scenario where communications could be constrained.

Finally, we argue that mobile agents can be very useful for data dissemination in vehicular networks. Thus, they can adapt easily to changing environmental conditions in order to improve the dissemination. For example, a basic flooding dissemination protocol will fail if the traffic density of the vehicles is low and there are not enough vehicles to re-diffuse the data towards the target, as well as other problems such as the storm broadcast (network overloading). Other dissemination protocols, such as carry-and-forward [15], where the vehicles may hold the data to be transmitted until these data can be relayed to other vehicles, can be used in the case of low traffic density. However, considering the variety of existing dissemination protocols (and others that could be developed in the future), mobile agents seem an ideal technology to implement flexible and dynamic dissemination approaches and take suitable routing decisions. 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. With this approach, the routing decisions lie with the data themselves (encapsulated in the mobile agents) and different dissemination protocols (dynamic and adaptive to the current conditions) can be implemented.

2.4 Simulators for VANETs

A significant number of network and traffic simulators have been developed, both commercial and free or open source. In this section we briefly present some of the most popular ones.

Network Simulators. Network simulators allow the configuration and simulation of detailed parameters of the devices and the communication process. For example, the technology used for data transmission (wireless, copper wire, fiber optic, etc.), the data loss ratio, latencies, shadowing effects that make wireless communications more difficult, distance attenuation, etc. Some of the most used simulators of this type are NS-3 and Qualnet (Quality Network). In our simulation approach we simulate communication networks from a high-level perspective, so we will not describe these simulators in more detail.
Traffic Simulators. Traffic simulators are specialized in the movement of the vehicles and allow to generate traces of their movement, following different patterns and behaviors in different scenarios. Some examples of such simulators are SUMO and VanetMobiSim.

SUMO (Simulation of Urban MObility)\(^1\) is an open source microscopic road traffic simulator. It allows the simulation of vehicles as single entities, with the ability of traveling through specific routes, changing the road lane, and following the traffic rules. It can handle scenarios with large road networks and a high number of vehicles. It can be enhanced with plugins and can interoperate with other software both by importing and exporting data using different file formats.

VanetMobiSim\(^2\) focuses on vehicular mobility, and features realistic automotive motion models at both macroscopic and microscopic levels. At macroscopic level, it can import maps from the US Census Bureau database, or randomly generate them using a Voronoi tessellation. It has also support for multi-lane roads, separate directional flows, differentiated speed constraints, and traffic signs at intersections. At microscopic level, it implements different mobility models, providing realistic car-to-car and car-to-infrastructure interaction. According to these models, vehicles regulate their speed depending on nearby cars, overtake each other, and act according to traffic signs in the presence of intersections.

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

EstiNet\(^3\) is a commercial product consisting on an extensible network simulator and emulator capable of simulating various protocols used in both wired and wireless IP networks, as well as wireless vehicular networks (including V2V and V2I communications), among others. Regarding its traffic simulation capabilities, it can simulate multi-lane road networks, it incorporates different microscopic vehicle mobility models, and the behavior of any vehicle can be changed as it receives messages from the vehicular network. It has been used for modeling VANETs and other ad-hoc networks as well as for the evaluation of real-life P2P applications and traffic signal control algorithms.

VEINS (Vehicles in Network Simulator)\(^4\) is an open source software that supports online re-configuration and re-routing of vehicles in reaction to network packets, it supports different vehicular mobility models, and relies on detailed models of IEEE 802.11p and IEEE 1609.4 DSRC/WAVE network layers (including multi-channel operation, QoS channel access, noise and interference effects). It can import scenarios from OpenStreetMap\(^5\), including buildings, speed limits, lane counts, traffic lights, and access and turn restrictions. It can also employ validated and computationally inexpensive models of shadowing effects caused

---

\(^1\) http://sumo-sim.org/
\(^2\) http://vanet.eurecom.fr/
\(^3\) http://www.estinet.com
\(^4\) http://veins.car2x.org
\(^5\) http://www.openstreetmap.org
by buildings as well as by vehicles. Finally, it supplies data sources for a wide range of metrics, including travel time and vehicle emissions.

It is also interesting to mention that the use of videogames to facilitate the evaluation of data management approaches for vehicular networks has also been proposed recently (see the VANET-X videogame⁶).

3 A Simulator for VANETs with Mobile Agents

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

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

In the rest of this section, we describe the main features of the simulator developed, its architecture, and a use case. The web site of the simulator (http://sid.cps.unizar.es/MAVSIM/) offers additional information, screenshots, videos, and the possibility to download it.

3.1 Features

The main features of MAVSIM are the following:

- It is written in Java, which makes it portable among different architectures and operating systems. It has been successfully tested in Microsoft Windows 7 and 8 (32 and 64 bits), Linux (32 and 64 bits), and Solaris (64 bits).
- It can run both in graphic interactive mode with a Graphical User Interface (GUI), or in batch mode (useful to execute a large number of simulations or experiments in an easy way).
- Simulations can be recorded and replayed later (with step by step, pause, and rewind and forward functionalities), which facilitates a careful analysis of the whole process. For this purpose, the replay tool shown in Figure 1 is used; in that screenshot, we also show in the screen some information that is relevant for the monitoring task scenario described earlier.
- Any road map can be downloaded from OpenStreetMap. There is no limitation in the type of layout imported (cities, highways, rural areas, etc.).

⁶http://sid.cps.unizar.es/Vanet-X/
The simulated mobile agents can be programmed in a similar way as they would be programmed using real platforms such as SPRINGS [7] (e.g., we provide methods such as `moveTo(targetDevice)` to simulate the movement actions performed by the agents). A built-in generic mobile agent platform is included in the simulator with the most common methods and primitives necessary for this.

- It can import traces generated by real vehicles or by other traffic simulators, and use them for the experiments with mobile agents.
- Road side units (or fixed communication devices) can be simulated.
- It includes a variety of movement algorithms for the simulated vehicles, such as *Random way-point*, *Gauss-Markov*, and others.
- In urban areas it can simulate the presence of buildings that block the wireless signal constraining the communications. For this purpose, the geometric method described in [8] is used.
- The initial scenario conditions for the experiments can be set randomly or using a known *seed*, which allows to reproduce the same conditions and repeat exactly the same experiment (with the same trajectories for the vehicles and other random conditions) if necessary.

A number of parameters can be set to configure in detail the simulation scenario. Table 1 shows a summary of some of the general parameters that can be configured, as well as parameters that are applicable for the specific case of the evaluation of monitoring approaches based on mobile agents. Most parameters, when omitted, take a default value; others are optional and do not have a default value.
Table 1. Summary of configuration parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcr</td>
<td>Mobile communication range</td>
<td>250 m</td>
</tr>
<tr>
<td>vld</td>
<td>Vehicle Linear Density (vehicles/km), overrides -v</td>
<td>100</td>
</tr>
<tr>
<td>s</td>
<td>Average speed of vehicles (km/h)</td>
<td>50</td>
</tr>
<tr>
<td>lat</td>
<td>Latency for a mobile agent's trip</td>
<td>1.5</td>
</tr>
<tr>
<td>errRate</td>
<td>Error rate for communications</td>
<td>0%</td>
</tr>
<tr>
<td>batch</td>
<td>Execute in batch mode</td>
<td>false</td>
</tr>
<tr>
<td>map</td>
<td>Load a scenario map</td>
<td>Last used</td>
</tr>
<tr>
<td>mbx</td>
<td>Number of mailboxes connected through a wired network (static nodes with storage capacity and the capability to provide wide-area network coverage)</td>
<td>0</td>
</tr>
<tr>
<td>f</td>
<td>Number of fixed (non-moving) wireless devices</td>
<td>1</td>
</tr>
<tr>
<td>nobldg</td>
<td>Do not simulate buildings in the map as communication obstacles</td>
<td>false</td>
</tr>
<tr>
<td>rec</td>
<td>Record the experiment to a file for later replay and analysis</td>
<td>true</td>
</tr>
<tr>
<td>seed</td>
<td>Seed for random numbers (0 for random seed)</td>
<td>0</td>
</tr>
</tbody>
</table>

Configuration parameters to evaluate monitoring approaches

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>Initial distance (in meters) to the interest area</td>
<td>1000</td>
</tr>
<tr>
<td>km</td>
<td>Mobile agent hop strategy (1 = RND, 2 = BEP, 3 = Approach, 4 = ANG, 5 = MAP, 6 = MAP-Trace, 7 = Map-Traj, 8 = Optimal)</td>
<td>8</td>
</tr>
<tr>
<td>m</td>
<td>Maximum number of simulation iterations during which to collect data</td>
<td>999</td>
</tr>
<tr>
<td>mih</td>
<td>Minimum expected improvement to enable an agent’s hop</td>
<td>0%</td>
</tr>
<tr>
<td>grid</td>
<td>Probability that a device contains relevant data / appropriate sensors</td>
<td>50%</td>
</tr>
<tr>
<td>rph</td>
<td>Max distance to hop (as a percentage of the communication range)</td>
<td>100%</td>
</tr>
<tr>
<td>snru</td>
<td>Selection of nearest Roadside Unit (0 = none, 1 = once, 2 = continuous)</td>
<td>0</td>
</tr>
</tbody>
</table>

3.2 Structure and Classes

The simulator is designed to be extensible. It contains a number of classes with different functionalities (see Figure 2), and the classes are organized in modules (packages): one package contains everything related to graphics, other the functionalities related to mobility strategies, etc. In this way it is easier to maintain the code and perform changes. Moreover, to facilitate its extensibility, some abstract classes and interfaces have been defined, that allow to add new functionalities in a quick and easy way. For example, if it is necessary to add a new mobility strategy for vehicles, there is an abstract class Mobility available that includes the necessary methods and attributes for all the mobility strategies and makes adding a new one easier: it would be enough to define a new subclass implementing the methods of Mobility and extend the configuration files to add support to select and use that new mobility strategy in the simulations. In the same way, there also exists an abstract class (TrackReader) to read different file formats of mobility traces, so if a new format is needed only the appropriate subclass must be implemented. Finally, one particularly important abstract class and interface are Agent and IAgent, respectively; they are intended to allow the programming of new different types of simulated mobile agents.

In this way, the simulator developed can be easily extended in a number of ways, such as: to incorporate new behaviors for the vehicles (e.g., new mobility strategies), to define new types of mobile agents with different behavior (e.g., different agent mobility strategies), to define different phases in the data management approach for easy analysis of each of them with the replay tool (e.g., in the monitoring task scenario phases such as “going to the area”, “measuring environment data”, and “coming back to the query originator” are defined, as shown in the upper-left part of Figure 1), or to add new simulation parameters.
We could also simulate data management approaches that do not use mobile agents, as we can simulate that there is a single static agent (i.e., a mobile agent with no mobility) in each equipped vehicle, that implements the required data management functionalities that would be available in the equipped vehicles. Most extensions require defining appropriate subclasses and configuration parameters. Of course, for a comfortable use of the extensions in an interactive simulation mode, changes to the graphical user interface may also be required.

### 3.3 Example of Simulation Scenario

As an example, we have used this simulator to evaluate environment monitoring strategies using mobile agents. The idea is to exploit sensors available in normal vehicles to monitor the environment, using mobile agents to find the relevant cars [13]. In these scenarios, there is a target area (or interest area) that has to be monitored (e.g., see Figure 3, which shows the target area as a rectangle near the center of the map, the node requesting the monitoring task as a point, and its communication range as a circle around the query point). One or more mobile agents can be sent to that area so that they can use sensors available in the vehicles therein to take measures of some environmental parameter (e.g., the CO₂) in the area. So, conventional vehicles with the appropriate sensors are exploited in a dynamic way by using mobile agents, leading to a more flexible and economic solution than an alternative and more traditional approach where
static sensors are manually deployed within the area of interest. However, the approach based on mobile agents faces some challenges, as the agents need to reach the target area by hopping from car to car (transportation via communication, using short-range wireless devices) and/or benefiting from the physical mobility of the vehicles (transportation via locomotion), keep themselves in vehicles within the area (to take the required measures), and finally return the results to the query originator. So, a key issue is to develop appropriate hopping strategies for the agents.

![Diagram of a monitoring task with a target area and query originator](image)

**Fig. 3.** Simulating agents in a monitoring task: target area and query originator

We want to evaluate the performance of these strategies to know which one is the best, under what circumstances, and how long takes the entire monitoring process to complete. The use of MAVSIM eases the process of designing and testing the behavior of the different strategies and the agents themselves, thanks to the tight integration of all the simulated objects, their interactions (the communicates with in the class diagram of Figure 2), and the facilities available for recording their status at every iteration of the process (e.g. the device and vehicle where the agent is, if an agent hop ends successfully or fails, etc.). In this way, they can be analyzed later and/or reviewed with the replay tool, searching for any anomaly or pattern. If it were necessary to correct the behavior of the simulated agents, given the modularity of the simulator, the only part necessary to be modified would be the agent code. The rest of the elements (vehicles, devices, etc.) would remain the same.

Moreover, when a simulation ends different data are provided: the total number of iterations completed; if the simulated process ended successfully or not;
the number of iterations taken at every phase of the process; the number of times that every mobile agent hopped; their effective speed, etc. All these data can be stored in a database or a spreadsheet, and if the simulation is repeated many times varying the initial conditions, all the data collected can be analyzed to extract conclusions about the whole process performance. Since all the simulation parameters can be set from the command line and be executed unattended in batch mode, it is easy to automatize the execution of a high number of simulations varying only a few initial parameters, making the experimental data collection task very convenient.

4 Conclusions and Future Work

Vehicular networks bring important challenges for the data management community. For practical reasons, the proposed communication and data management strategies are usually evaluated using simulators, which provides flexibility in defining the required test conditions and enables quick results considering large-scale simulated scenarios with an affordable cost. However, although the use of mobile agents in the context of VANETs could be interesting, existing simulators cannot be easily adapted to evaluate approaches based on the use of mobile agents. For that reason, we have developed our own simulator.

We think that the simulator developed is quite well suited for the simulation of mobile agents in vehicular networks, taking into account different elements such as the network communications, the moving vehicles, the presence of buildings that block the signal, and other parameters of interest for testing different scenarios. The simulator is written in a modular and extensible way, so it can be enhanced with new features with little effort. We are currently using the simulator in our research, and we are constantly adding new features when we need them to evaluate different data management strategies. Thanks to the use of this simulator, we have obtained promising results regarding the potential interest of using mobile agents in vehicular networks [12–14].

There are however a number of features and enhancements that have not been implemented so far. Among these, we are especially interested in the development of some extensions to simulate events (e.g., accidents, traffic jams, etc.) and parking spaces, as they will enable testing data management strategies for the exchange of interesting events for drivers [2] and applications to help drivers in the search of appropriate parking spaces [6]. Moreover, we also plan to study the possibility of mixing the simulator developed with functionalities provided by existing general-purpose traffic simulators (in particular, SUMO with TraCI).

Acknowledgments This work has been supported by the CICYT project TIN2010-21387-C02-02 and DGA-FSE.
References


