Testing Agent-based Mobile Computing Applications Using Distributed Simulations[**]

S. Ilarri[††]
IIS Depart., Univ. of Zaragoza
Maria de Luna 1
50018 Zaragoza, Spain
Email: silarri@unizar.es

E. Mena
IIS Depart., Univ. of Zaragoza
Maria de Luna 1
50018 Zaragoza, Spain
Email: emena@unizar.es

A. Illarramendi
LSI Depart., Basque Country Univ.
Manuel Lardiazabal 1
San Sebastián, Spain
Email: jipileca@si.ehu.es

Abstract

The attractiveness of computing services that are available anywhere and anytime has given rise to an intense research and development of mobile computing applications. However, there is a great pressure over providers of wireless data services to be the first in providing a new service. This leads to the need of a mechanism that allows developers to test their prototypes in a timely and non-expensive fashion, what excludes the possibility of performing real tests.

In this paper we present a simulation framework for evaluating data services designed for wireless environments. Our approach, based on mobile agents, allows to perform distributed simulations of mobile networks composed of moving objects and proxies. We focus on the problems that arise to simulate moving objects that execute multiagent applications. Finally, we show the usefulness of our solution by testing a sample application.

1. Introduction

Testing mobile computing applications is an overwhelming task, mainly due to the distributed and dynamic nature of the underlying network infrastructure. Thus, testing a service is sometimes as time-consuming as developing it. Besides, deploying a data service in a real network with testing purposes is not usually an option: 1) we could need *many* users with *real* wireless devices following certain behavior to test how the new data service behave in that situation, 2) some extensions to the current network functionality could be required (e.g., we could want to simulate base stations that store the GPS location of moving objects under their coverage in order to provide location-based services with high-precision location data), 3) bugs in the new data service could affect the regular functionality of the real network, and 4) mobile users should not experience the overload of testing new applications.

As the context of our work, we consider a mobile environment architecture composed of two elements:

- *Proxies*: they are computers that provide mobile users under their area with data services. In a cellular network, a *proxy area* can be seen as the union of the coverage areas of one or more base stations. Notice that more specific terms could be used to designate a proxy depending on the service it provides; for example, a proxy that provides location information is usually known as a *Location Server*.
- *Moving Objects*: they are entities provided with a wireless device that allows them to access data services; for example, a car equipped with a portable computer connected to the Internet. Moving objects can be attached to devices such as GPS receivers or sensors that feeds them with context information. Similarly, they can include elements capable of executing different software, such as a word processor, a web browser or a route guiding system that shows on a map the GPS location of the moving object.

In [5] we presented our first approach for evaluating data services designed for wireless environments. We simulated a proxy by creating a *place* [2] in a fixed computer with a *proxy agent* which provided the *wanted* functionality of a proxy. Similarly, mobile agents were used as the natural software representations of their hardware counterparts (moving objects), and handoffs between proxies were naturally mapped to trips between the involved proxies by the corresponding mo-

[*] [*]Supported by the CICYT project TIC2001-0660, the DGA project P084/2001, and the research institute I3A.

[†] [†]Work supported by the Aragón Government and the European Social Fund (ref. B132/2002).

bile agents. In general, the simulation of a moving object consists of three tasks:

1. *Simulation of the object's movements.* We compute the current location of the object by using a trajectory function defined for the moving object. This location could be used for a variety of purposes; e.g., to simulate that the moving object communicates its location to its proxy.

2. *Simulation of handoffs.* We take advantage of the parallelism between moving objects and mobile agents by simulating a handoff (change of proxy area) of a moving object as a trip of the mobile agent that simulates that moving object (from the old to the new proxy). Thus, mobile agents will always interact locally with their proxies in the same way that moving objects interact with theirs. This is a natural and convenient way of distributing the simulation load across several computers, while avoiding the introduction of artificial delays in the simulation.

3. *Simulation of the functionalities* of the object that are provided to real objects by hardware appliances or software applications. Each interesting behavior will be implemented as a thread in the mobile agent. For example, we could use a thread to simulate requests of web pages by the user, and another thread to simulate the periodic communication of the user's location to its proxy.

However, the simulation of moving objects become a daunting task if we are interested in simulating that they are executing complex applications. One of the most complicated applications that moving objects can execute are, in our opinion, those based on mobile agents [2]. Although mobile agents have been proved to be very useful in mobile computing, complex distributed coordination mechanisms are usually needed for them to cooperate to attain their goal; in addition, mobile agents can travel across computers (they can decide to pause their execution on its current moving object and travel to another computer or moving object where the execution will be resumed). Instead of simulating the mobile agents that should be executed on the moving object (e.g., by using threads), we propose a simulation framework where the code of the real application can be reused in the simulations. With this approach, the time required to prepare a simulation is greatly reduced. The main focus of this paper is to show how to realize this *plug-and-simulate* proposal while keeping the original application's mobile agents unaware of the real or virtual nature of the mobile device where they execute.

With respect to our previous work, although we keep the basics of our approach, we have shifted our goal of getting a simulator to providing a simulation framework along with a graphical user interface that eases the tasks of definition and execution of a scenario using classes implemented with the framework. This change eases the testing of applications other than the one that motivated its initial development: a location-dependent query processor [4]. Thus, we found that the simulator-based approach was difficult to adapt to different simulation needs. On the contrary, the simulation framework allows developers of wireless applications to test their services in an easy way, with minimal changes in their code, and easily extend such a framework to implement moving objects and proxies adapted to their needs. Besides, we allow now the simulation of moving objects that execute multiagent applications.

The rest of the paper is as follows. In Section 2, we describe our approach to simulate moving objects that execute multiagent applications. In Section 3 we evaluate our approach by simulating and testing a sample wireless data service based on mobile agents. In Section 4 we analyze some related works. Conclusions and future work are presented in Section 5.

2. Simulating Multiagent Applications Running on a Moving Object

If we want that a simulated moving object executes a multiagent application, because it is interesting or needed for our tests, then such a multiagent application should theoretically execute "inside" the mobile agent that simulates the moving object, which is obviously not possible.

We advocate a solution based on reusing the original code of the multiagent application to perform a simulation as close to reality as possible and with a minimum effort. Thus, agents that should execute on the simulated moving object will be reused directly; they will be extended with the mechanism needed to make them *go together* with the mobile agent that simulates the moving object that should contain the multiagent application. One additional advantage of this approach is that, in the simulated scenario, multiagent applications "running" on simulated moving objects will behave exactly as expected: some of those agents could even move to real computers outside the simulation scenario, access a real database, and come back to the simulation scenario with the result.

2.1. Real and Simulated Scenario

As shown in Figure 1, while in a real scenario there exist moving objects containing agents, in our simulation we would have several mobile agents: the mobile agent that we use to simulate the moving object, that we call *Moving Object Agent (MOA)*, and the agents used in the multiagent application, that we call *Internal Agents (IAs)*. We summarize the equivalence between real and simulated scenarios in the following table:

Real	Simulated
Moving object (e.g., a PDA)	Mobile agent MOA
Place in moving object	Place in proxy
Agents in moving object	Mobile agents IAs

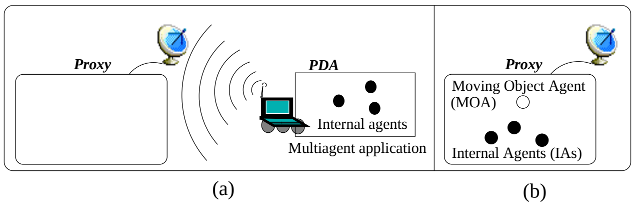

Figure 1. Moving objects executing multiagent applications: (a) real scenario and (b) simulated scenario

When an MOA moves to another proxy (to simulate the handoff of the simulated moving object), its associated IAs must move too, in the same way that agents executing on a real PDA will remain there although such a PDA moves. Thus, the MOA has to coordinate itself with its IAs when a travel is needed, making sure that its IAs move with it as well.

To achieve this goal, we need to extend the functionality of the original agents in order to allow the MOA to synchronize them in the simulation scenario. For that purpose we have defined a set of classes that provide such agents with the required features, in such a way that their original functionality will not be disturbed (they will be unaware of being running in a simulation scenario).

2.2. Classes in the Framework

In Figure 2 we show the classes that comprise our framework. The class *MobileAgentForSimulation* is the root of the hierarchy and extends the class that contains the basic functionality of a mobile agent (provided by the mobile agent platform used) by adding mechanisms to delay trips when required for agent synchronization (needed for both MOAs and IAs). Two classes extend MobileAgentForSimulation: *InternalAgent* is the implementation of the functionality of an IA, while *MovingObjectAgent* implements the basic functionality of a moving object (an MOA); i.e., it simulates that: 1) it follows a given trajectory, and 2) it is able to interact with a proxy. Finally, the class *Proxy* provides the functionality needed by any simulated proxy (e.g., the capability of broadcasting messages to all the simulated moving objects under its area).

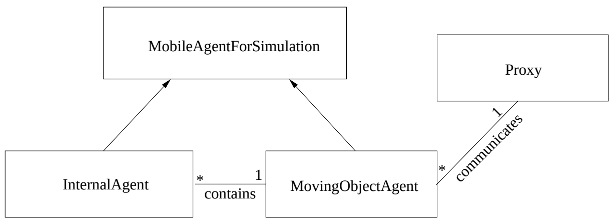

Figure 2. Class Diagram

The tester of a data service should extend classes MovingObjectAgent and Proxy to define the moving objects and proxies needed for his/her specific simulation tests (see Section 3). For example, a subclass of MovingObjectAgent could add the required code to communicate periodically the GPS location to its proxy, and a subclass of Proxy could receive this location data and store them in a database. In this way, developers of wireless data services can test features that are not even available in a real network. Similarly, for multiagent applications interesting for the simulation, each agent that should execute on a simulated moving object must extend the class InternalAgent to be movable.

2.3. Coordination Mechanism

We describe here the different stages of the coordination mechanism, implemented in our simulation framework:

1. *Detection of the need to travel.* The MOA gets the next location for the trajectory of the moving object that it represents and realizes that it implies a change of proxy area.
2. *Request of movement.* The MOA requests its IAs to travel to the destination proxy.
3. *Packaging for the trip.* The IAs get ready for the trip by waiting until the current *transaction* (sequence of operations that cannot be interrupted by a trip) finishes.

4. *Traveling.* The MOA and IAs move to the new proxy. The order in which they travel is not important, they can travel in parallel.
5. *Notification of arrival.* The IAs send an acknowledge to its MOA to notify it of their arrival. The notification of arrival can fail because the MOA is still traveling to the destination, in which case the notification of arrival will be retried. After that, the IAs will wait until the MOA tells them to resume their execution.
6. *End of Travel.* When the MOA arrives at the destination, it waits until it receives notification of arrival from all its IAs, and then the travel is considered finished.
7. *Resume Execution.* The MOA sends a notification to its IAs for them to resume the execution, and it will also continue its own execution.

3. Evaluation: Testing a Locker Rental Service (LRS)

In this section, we evaluate the suitability of the developed simulation framework by using it to test the Locker Rental Service [10]. As we will show with this example, we can use our simulation framework to test an agent-based service before it has been deployed in a real network.

The Locker Rental Service offers users the possibility of keeping their data in a secure and safe space called *locker* (located on a proxy). It uses mobile agents to provide mobile users with a reliable, efficient and convenient access to additional hard disk space. One of the main elements of the system is a mobile agent called *Locker* that is in charge of carrying and accessing the data in the locker. This agent can move from proxy to proxy to stay "close" to the user, with the goal of optimizing the cost of communications. There also exists a majordomo agent called *Alfred* running on the user device whose goal is to help the user to interact with the rest of the system, isolating the user from the network availability and the specific features of the mobile device.

To test this multiagent service, we have defined a simulated scenario where a user in the US travels to Spain, concretely to San Sebastian and then to Zaragoza. We have simulated that he/she requests files of about 190 KB from its PDA. To be able to obtain results in a short time, we use a temporal scale factor in the simulations.

To test the Locker Rental Service, we have used three fixed computers to represent the proxy in the USA, San Sebastian and Zaragoza, respectively. From the point of view of implementation, we have subclassed: MovingObjectAgent (class *PDA*) to extend it with the simulation of the interactions of a user asking for files periodically; InternalAgent (class *Alfred*) to act as an interface between the user and the rest of the service; Proxy (class of *LRS_Proxy*) to implement the capability of serving files; and MobileAgentForSimulation (class *LockerAgent*), that returns the requested files to Alfred (on behalf of the user) and decides when to move the locker to another computer (to keep it close to the user). Classes Alfred and LockerAgent are the original ones with the exception of some changes needed to use the simulation framework and, in the case of Alfred, to coordinate itself with the PDA class which represents its moving object.

We show in Figure 3 some performance results of the Locker Rental Service obtained using our simulation framework. The x-axis shows the user requests and the y-axis shows the accumulated time of interaction to obtain the requested files. We show results for three different strategies for the Locker movement: 1) it always moves, whenever the user device changes to a new proxy (handoff), 2) it never moves, and 3) it decides if it should move or not (by taking into account, for example, estimations regarding the network speed and file requests). We can use this simulation to easily evaluate under which circumstances it is better to use one strategy or another, or to change the rules used to decide when to move the locker and see the results immediately and at no cost.

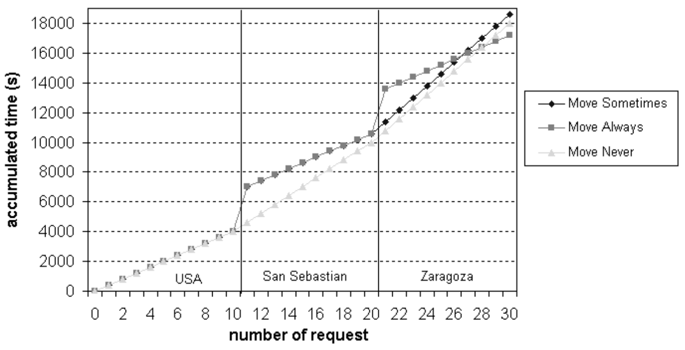

Figure 3. Testing the Locker Rental Service

4. Related Work

The goal of our simulation framework is to allow developers to test high-level data services and not to evaluate network protocols. Consequently, our work is orthogonal to those dealing with the simulation of wireless networks at a protocol level (for example, [8, 3]), and we could use them to improve our simulations at a low level.

There are some works about the problem of generating location data for benchmarking spatio-temporal databases [7, 6, 1]. These works focus mainly on the simulation of moving objects as generators of spatio-temporal data for performance evaluation, and thus their goal and features differ from ours.

In ABELS [11] they use mobile agents in a distributed environment where simulations are performed. However, their purpose is different: mobile agents are used to efficiently link independent simulations that can be executing in different computers, and not as elements of the simulations themselves.

There are some approaches to model and simulate mobile agents in distributed environments like [9]. By contrast, we propose not to simulate, but to reuse existing multiagent applications with the goal of testing them in simulation scenarios.

As far as we know, no other approach takes advantage of the mobile agent technology to simulate a mobile computing infrastructure. Similarly, they do not allow the testing of agent-based applications that are executed by the moving objects.

5. Conclusions and Future Work

In this paper we have presented a framework for the distributed simulation of mobile environments with the purpose of testing wireless data services. We have used our framework to prove how service designers can leverage it to test their prototypes without the inconveniences and expenses that real tests imply. We perform distributed simulations, which allows us to increase the parallelism of simulation (the simulation itself is not aware of the number of computers used). The use of mobile agents avoids the appearing of extra remote communications and their delays, which would decrease the accuracy of the simulation. Besides, we have studied the coordination issues that arise when we want to "directly" test applications based on mobile agents in our simulation environment.

As future work, we plan to study how to minimize even more the number of changes needed to reuse the existing code for the simulation. For example, in some situations many changes must be incorporated to the preexisting agents to allow a reasonable availability to be moved by their MOA (e.g., the agent must avoid to perform long operations that cannot be interrupted).

Acknowledgements

We would like to thank Stephane BUI-PHUONG for his contribution to the implementation work behind this paper.

References

[1] T. Brinkhoff. A framework for generating network-based moving objects. *Geoinformatica*, 6(2):153–180, June 2002.

[2] D. Milojicic et al. MASIF, the OMG mobile agent system interoperability facility. In *Proceedings of Mobile Agents'98*, September 1998.

[3] K. Fall. Network emulation in the vint/ns simulator. In *Proceedings of the 4th IEEE Symposium on Computers and Communications*, July 1999.

[4] S. Ilarri, E. Mena, and A. Illarramendi. A system based on mobile agents for tracking objects in a location-dependent query processing environment. In *Fourth International Workshop Mobility in Databases and Distributed Systems (MDSS'2001), Munich (Germany)*, pages 577–581. IEEE Computer Society, ISBN 0-7695-1230-5, September 2001.

[5] E. Mena, S. Ilarri, and A. Illarramendi. Distributed simulation of wireless environments using mobile agents. In *2002 International Conference on Wireless Networks (ICWN'02), Las Vegas, USA*, pages 103–110. CSREA Press, ISBN 1-892512-30-0, June 2002.

[6] D. Pfoser and Y. Theodoridis. Generating semantics-based trajectories of moving objects. In *International Workshop on Emerging Technologies for Geo-Based Applications, Ascona, Switzerland*, 2000.

[7] Jean-Marc Saglio and Jose Moreira. Oporto: A realistic scenario generator for moving objects. In *DEXA Workshop*, pages 426–432, 1999.

[8] Didier Samfat and Refik Molva. Idamn: An intrusion detection architecture for mobile networks. *IEEE Journal of Selected Areas in Communications*, 15(7):1373–1380, October 1997.

[9] Adelinde M. Uhrmacher, Petra Tyschler, and Dirk Tyschler. Modeling and simulation of mobile agents. *Future Generation Computer Systems*, 17(2):107–118, 2000.

[10] Y. Villate, A. Illarramendi, and E. Pitoura. Keep your data safe and available while roaming. *International Journal of Mobile Networks and Application (MONET), Special Issue on Pervasive Computing*, 7(4):315–328, August 2002.

[11] Linda F. Wilson, Daniel J. Burroughs, Anush Kumar, and Jeanne Sucharitaves. A framework for linking distributed simulations using software agents. In *Proceedings of the IEEE, Special issue on agents in modeling and simulation: exploiting the metaphor*, volume 89, pages 174–185, february 2001.